# Petri Nets: Applications

# Petri Nets: Applications

Edited by
Pawel Pawlewski

*In-Tech*
*intechweb.org*

Published by In-Teh

**In-Teh**
Olajnica 19/2, 32000 Vukovar, Croatia

Abstracting and non-profit use of the material is permitted with credit to the source. Statements and opinions expressed in the chapters are these of the individual contributors and not necessarily those of the editors or publisher. No responsibility is accepted for the accuracy of information contained in the published articles. Publisher assumes no responsibility liability for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained inside. After this work has been published by the In-Teh, authors have the right to republish it, in whole or part, in any publication of which they are an author or editor, and the make other personal use of the work.

© 2010 In-teh
www.intechweb.org
Additional copies can be obtained from:
publication@intechweb.org

First published February 2010
Printed in India

Technical Editor: Maja Jakobovic
Cover designed by Dino Smrekar

Petri Nets: Applications,
Edited by Pawel Pawlewski

p. cm.
ISBN 978-953-307-047-6

# Preface

Petri Nets are the graphical and mathematical tool used in many different science domains. Their characteristic features are the intuitive graphical modeling language and advanced formal analysis method. The concurrence of performed actions is the natural phenomenon due to which Petri Nets are perceived as mathematical tool for modeling concurrent systems. The nets whose model was extended with the time model can be applied in modeling real-time systems.

Petri Nets were introduced in the doctoral dissertation by K.A. Petri, titled "„Kommunikation mit Automaten" and published in 1962 by University of Bonn. During more than 40 years of development of this theory, many different classes were formed and the scope of applications was extended. Depending on particular needs, the net definition was changed and adjusted to the considered problem. The unusual "flexibility" of this theory makes it possible to introduce all these modifications. Owing to varied, currently known net classes, it is relatively easy to find a proper class for the specific application.

The present monograph shows the whole spectrum of Petri Nets applications, from classic applications (to which the theory is specially dedicated) like computer science and control systems, through fault diagnosis, manufacturing, power systems, traffic systems, transport and down to Web applications. At the same time, the publication describes the diversity of investigations performed with use Petri Nets in science centers all over the world.

Pawel Pawlewski

# Contents


Preface V

1. An Application of GSPN for Modeling and Evaluating Local Area Computer Networks 001
   Masahiro Tsunoyama and Hiroei Imai

2. Architecture of Computer Intrusion Detection Based on Partially Ordered Events 013
   Liberios Vokorokos and Anton Baláž

3. Particle Filter for Depth Evaluation of Networking Intrusion Detection Using Coloured Petri Nets 029
   Chien-Chuan Lin and Ming-Shi Wang

4. Modeling and Analyzing Software Architecture Using Object-Oriented Petri Nets and $\pi$-calculus 045
   Zhenhua Yu, Xiao Fu, Yu Liu, JingWang and Yuanli Cai

5. Systolic Petri Nets 061
   Alexandre Abellard and Patrick Abellard

6. Towards Rewriting Semantics of Software Architecture Specification 093
   Yujian Fu, Zhijiang Dong, Phil Bording and Xudong He

7. Transfer-Resource Graph and Petri-net for System-on-Chip Verification 115
   Xiaoxi Xu and Cheng-Chew Lim

8. Using Petri nets for modeling and verification of Hybrid Systems 137
   Ricardo Rodriguez, Otoniel Rodriguez, Gerardo Reyes and Vianey Cruz

9. A new Control Synthesis Approach of P-Time Petri Nets 161
   Bonhomme Patrice

10. CL-MAC: Cross-layer MAC Protocol for Delay Sensitive Wireless Sensor Network Applications 179
    Kechar Bouabdellah and Sekhri Larbi

11. Distributed Implementation of Petri nets in Control Applications 207
Ramon Piedrafita, Danilo Tardioli and Jose Luis Villarroel

12. Hybrid state Petri nets which have the analysis power of stochastic hybrid systems and the formal verification power of automata 225
Mariken H.C. Everdij and Henk A.P. Blom

13. Reachability Analysis of Time-Critical Systems 251
Štefan Hudák, Štefan Korečko and Slavomír Šimoňák

14. Supervisory Control and High-level Petri nets 279
Chiheb Ameur ABID, Sajeh ZAIRI and Belhassen ZOUARI

15. Using Petri Net for Modeling and Analysis of a Encryption Scheme for Wireless Sensor Networks 305
Hugo Rodríguez, Rubén Carvajal, Beatriz Ontiveros, Ismael Soto and Rolando Carrasco

16. Diagnosability in Switched Linear Systems 319
Lizette Rubio- Gómez, David Gómez-Gutiérrez, Antonio Ramírez-Treviño, Javier Ruiz-León and Guillermo Ramírez-Prado

17. Fault diagnosis for complex systems using Coloured Petri Nets 333
L. Rodríguez, E. García, F. Morant, A. Correcher and E. Quiles

18. Modelling and Fault Diagnosis by means of Petri Nets. Unmanned Aerial Vehicle Application 351
Miguel Trigos , Antonio Barrientos , Jaime del Cerro and Hermes López

19. Design and Implementation of Hierarchical and Distributed Control for Robotic Manufacturing Systems using Petri Nets 377
Gen'ichi Yasuda

20. Performance Evaluation of Distributed Systems: A Component-Based Modeling Approach based on Object Oriented Petri Nets 391
Aladdin Masri, Thomas Bourdeaud'hui and Armand Toguyeni

21. Using Petri Nets to Model and Simulation Production Systems in Process Reengineering (case study) 419
Pawlewski Pawel

22. Workflow Diagnosis Using Petri Net Charts 445
Calin Ciufudean and Constantin Filote

23. Evaluation of Power System Security with Petri Nets 469
Jose L. Sanchez, Mario A. Ríos and Gustavo Ramos

24. Fault Diagnosis on Electric Power Systems based on Petri Net Approach 491
Alejandra Santoyo-Sanchez, José Alberto Gutiérrez-Robles, Elvia Ruiz-Beltrán, Carlos Alberto De Jesús–Velasquez, Luis Isidro Aguirre-Salas and Víctor Ortiz-Muro

25. GPenSIM: A New Petri Net Simulator 525
Reggie Davidrajuh

26. Assessing Risks in Critical Systems using Petri Nets 539
Lucio Flavio Vismari and João Batista Camargo Junior

27. Modelling and Analysis of Traffic Light Control Systems Using Timed Coloured Petri nets 565
Yi-Sheng Huang and Ta-Hsiang Chung 2

28. Traffic Network Control Based on Hybrid System Modeling 587
Youngwoo Kim

29. Using Petri Nets in the analysis of sequential automata models with direct applications on the transport systems with accumulation areas 623
Dan Ungureanu-Anghel

30. An Approach Based in Petri Net for Requirement Analysis 651
Ermeson Andrade, Paulo Maciel, Gustavo Callou, Bruno Nogueira and Carlos Araujo

31. Intuitive Transformation of UML2 Activities into Fundamental Modeling Concept Petri Nets and Colored Petri Nets 671
Anthony Spiteri Staines

32. Multilevel Petri Nets for the Specification and Development of Workflow Automation Systems 693
Marina Flores-Badillo and Ernesto López-Mellado

33. An Application of Petri Nets to e/m-Learning Environments 713
Cristina De Castro and Paolo Toppan

34. Petri nets-based Models for Web Services Composition 729
Huaikou Miao and Tao He

1

# An Application of GSPN for Modeling and Evaluating Local Area Computer Networks

Masahiro Tsunoyama* and Hiroei Imai **
* *Department of Information and Electronics Engineering, Niigata Institute of Technology*
*1719 Fujihashi, Kashiwazaki 945-1195, JAPAN*
*E-mail: mtuno@iee.niit.ac.jp*
** *University Evaluation Center, Niigata University,*
*8050 Ikarashi-2, Niigata-shi, Niigata 950-2181, JAPAN*
*E-mail: himai@adm.niigata-u.ac.jp*

## 1. Introduction

Multimedia systems connected by computer networks are widely used in applications such as telecommunications, distance-learning, and video-on-demand (Nerjes et al., 1997;Kornkevn & Lilleberg, 2002;Shahraray et al., 2005). Since multimedia data have real-time properties that must be processed and delivered within given deadlines, the demand on such systems is increasing (Althun et al., 2003;Gibson & David, 2007). In order to maintain the required quality, several systems using QoS techniques have been proposed (Furguson & Huston, 1998;Park, 2006;Villalon et al., 2005). The IEEE802.11e (IEEE Standard, 2003) is one of these techniques. It provides two functions for QoS support: enhanced distributed channel access (EDCA) and hybrid coordination function controlled channel access (HCCA). HCCA uses concentrated control and guarantees the required propagation delay. On the other hand, EDCA uses distributed control, has good scalability, and requires less overhead than HCCA, but cannot guarantee the required propagation delay. In order to assess the dependability of multimedia systems using QoS, such as the IEEE802.11e supporting EDCA, the propagation delay and its standard deviation (jitter) must be quantitatively evaluated (Claypool & Tanner, 1999;Fan et al., 2006;Gibson & David, 2007;Park, 2006).

Several evaluation methods have been proposed, such as queuing networks (Ahmad, et al., 2007;Cheng & Wu, 2005), stochastic process models (German, 2000;Nerjes et al., 1997), and simulation models (Adachi et al., 1998;Bin et al., 2007;Grinnemo & Brunstrom, 2002). However, these methods have several problems. Queuing networks and stochastic process models are analytical models, which do not require a long time for computation. However, it is difficult to model the given systems, since the number of states in a model increases exponentially as the system increases in size, particularly when the systems are large and complex. Though simulation models are used for evaluating systems, they require a long time to obtain statistical data regarding the standard deviation (jitter). This chapter proposes a method for evaluating systems using the Generalized Stochastic Petri Net and the tagged task approach

(Imai et al., 1997;Kumagai et al., 2003). GSPNs are an extension of the Petri Nets that can be easily used to model the timing behavior of systems. The tagged task approach can reduce the number of states in a model by tracing the behavior of a tagged task.
A method for evaluating local area computer network systems, such as the IEEE802.11e WLAN supporting EDCA, based on delay jitter analysis using the Generalized Stochastic Petri Net (GSPN) and the tagged task approach, is fully explained. The system is modeled using GSPN with the tagged task approach, then the state transition diagram of the Markov chain is constructed from the reduced reachability graph of the GSPN model. Processing paths are extracted, and the mean value and variance of the delay time are calculated using the equations derived from the Markov chain. An evaluation example is also given. Section 2 explains system modelling using GSPN, while Section 3 presents the evaluation method that will be used. Section 4 describes the evaluation example, which is a system built using IEEE802.11e WLAN supporting EDCA. Finally, Section 5 summarizes the results of this chapter.

## 2. Modeling Network Systems Using GSPN

### 2.1 GSPN

GSPN can be defined as follows (Marson et al., 1995). The set of all natural numbers will be denoted as *N*, while the set of all real numbers will be denoted as *R*.
[Definition1]

$$N_{GSPN} = (P, T, W^-, W^+, W^h, \Lambda, \omega, M_0) \tag{1}$$

$P = \{p_i \mid 1 \le i \le |P| = n\}$ ; Set of places,

$T = \{t_j \mid 1 \le j \le |T| = m\}$ ; Set of transitions,

$T = T_I \cup T_T, T_I \cap T_T = \varnothing$ ; $T_I$ is a set of immediate transitions, $T_T$ is a set of timed transitions,

$W^- : P \times T \to N$ ; Input connection function,

$W^+ : T \times P \to N$ ; Output connection function,

$W^h : P \times T \to N$ ; Inhibitor arcs,

$\Lambda = \{\lambda_i \mid 1 \le i \le |T_T|\}$ ; Firing rates,

$\omega : T_I \to R$ ; Weighting function of immediate transitions,

$m_0$ : Initial marking.

In GSPN, places are represented by circles; timed transitions by boxes; and immediate transitions by thin bars. An inhibitor arc ends in a small circle. A timed transition fires according to the firing rate assigned to the transition when the firing condition is satisfied. Fig.1 shows a typical GSPN for M/M/1/1/3. In the figure, $p_1$, $p_2$, $p_3$, $p_4$, and $p_5$ are places; $t_1$ and $t_3$ are the timed transitions; $t_2$ is an immediate transition; and $\lambda_1$ and $\lambda_3$ are the firing rates for transitions $t_1$ and $t_3$.

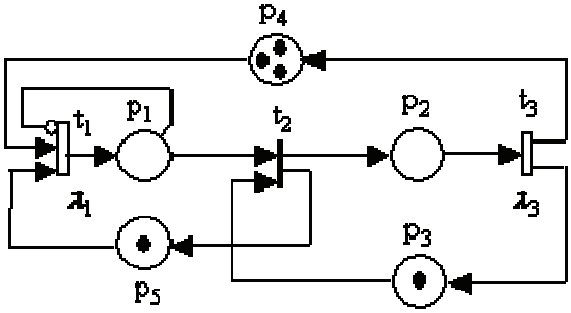



Fig. 1. Sample GSPN

### 2.2 Reachability Graph and Markov Chain

In the example net, the transition $t_1$ fires after the time determined by the exponential probability distribution function with parameter $\lambda_1$, and the tokens in places $p_4$ and $p_5$ move to place $p_1$. The assignment of tokens to places is called marking. In this example, the marking changes from the initial marking $m_0$ to the next marking $m_1$ when $t_1$ fires, as shown in Fig.2. The change in markings is represented by Equation (2). In Equation (2), $m_0[t_1 > m_1$ indicates that the marking $m_0$ changes to $m_1$ after the transition $t_1$ fires.

$$m_0[t_1 > m_1[t_2 > m_2[t_3 > m_0$$
$$m_0[t_1 > m_1[t_2 > m_2[t_1 > m_3[t_3 > m_0 \quad (2)$$

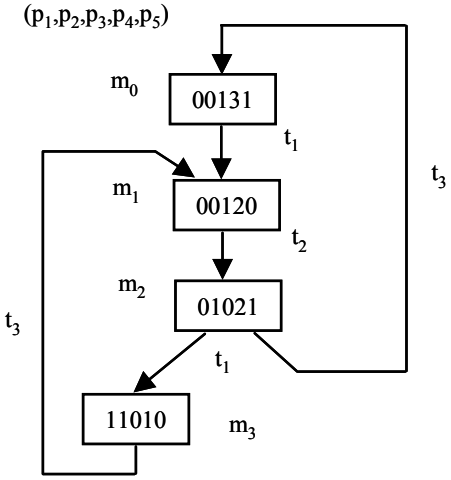



Fig. 2. Reachability graph for the sample GSPN.

The set of markings reached from $m_0$ is called a reachability set and is defined as follows: [Definition 2]

The minimum set of markings satisfying the following condition is called the reachability set of the initial marking $m_0$ and is represented by $RS(m_0)$.

$$
\begin{aligned}
& m_0 \in RS(m_0), \\
& m_1 \in RS(m_0) \wedge \exists t \in T : m_1[t > m_2 \\
& \Rightarrow m_2 \in RS(m_0)
\end{aligned}
\tag{3}
$$

The change of markings in a reachability set can be represented by a graph. The graph of all reachable markings from the initial marking is called the reachability graph and is defined as follows.

[Definition 3]

A labeled digraph is called a reachability graph and is represented by $RG(m_0)$ when the set of nodes in the graph is $RS(m_0)$, and the set of edges A in the graph is defined by the following equation:

$$
\begin{aligned}
& A \subseteq RS(m_0) \times RS(m_0) \times T \\
& (m_i, m_j, t) \in A \Rightarrow m_i[t > m_j, m_i, m_j \in RS(m_0)
\end{aligned}
\tag{4}
$$

The GSPN has two kinds of markings: tangible and vanishing. Tangible markings allow timed transitions to fire, while vanishing markings allow immediate transitions to fire. Vanishing markings can be reduced by eliminating them from the reachability graph. The reduced reachability graph is equivalent to the state transition diagram of a Markov chain for the GSPN model (Marson et al., 1995) and is shown in Fig.3.

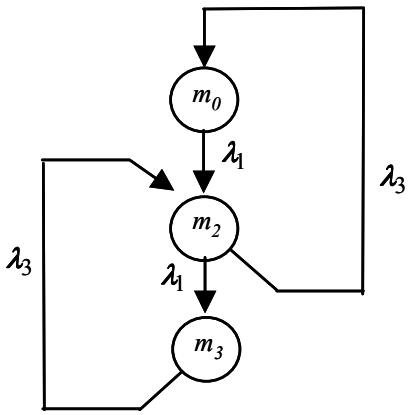



Fig. 3. State diagram of the Markov chain for the sample GSPN.

## 3. System Model

In network systems processing multimedia data with QoS control, tasks are processed according to their priorities for satisfying their QoS requirement. The following system assumptions are useful for analysis.

[Assumption1]

Each task has a priority, which determines when it is processed and delivered.

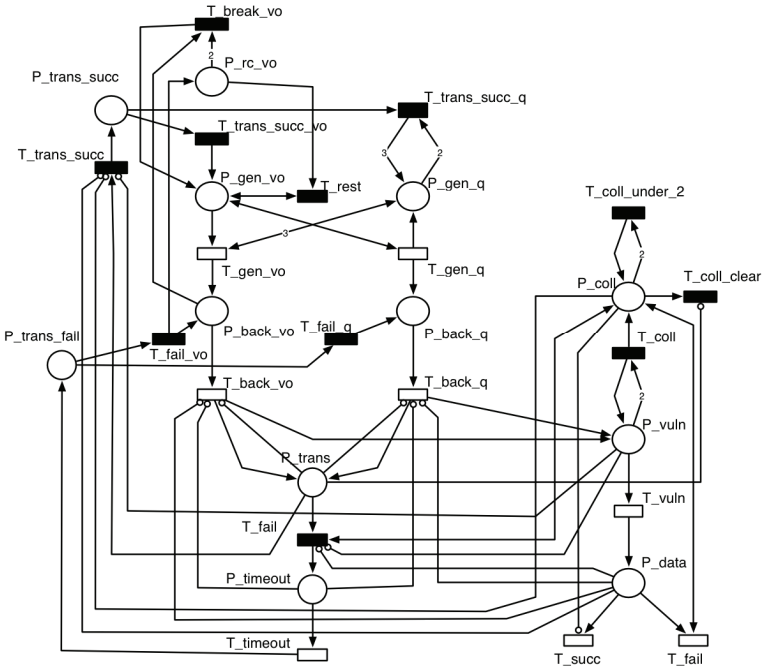
In network systems containing many hosts, tasks occur randomly, and the processing time for tasks may be an arbitrary value. Thus, the following assumptions are made about the tasks:

[Assumption2]

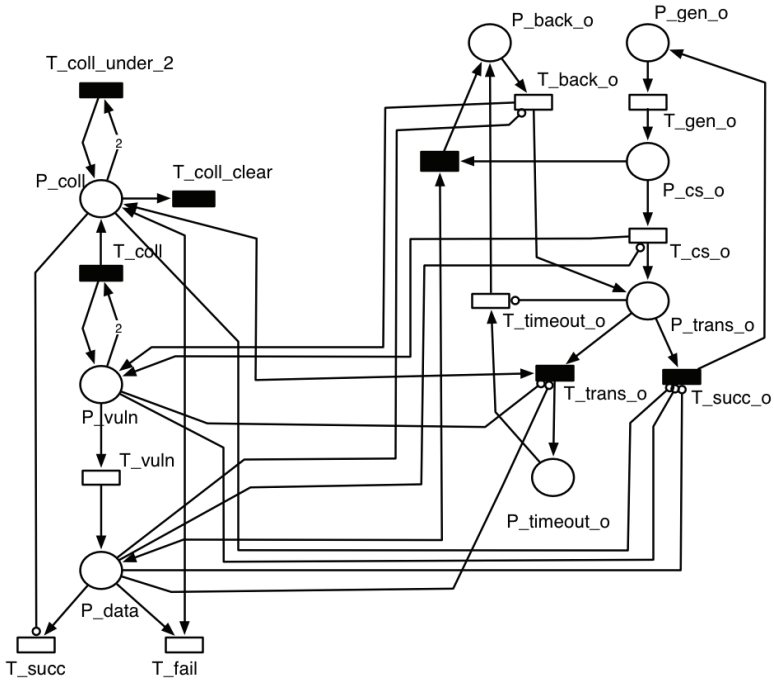
(1) Tasks occur according to a Poisson process.

(2) Task processing time is determined by the exponential probability distribution function.

The IEEE 802.11e WLAN supporting EDCA is used as the example for explaining the system model and the evaluation method. The IEEE 802.11e WLAN supporting EDCA has four access categories (ACs): AC_VO, AC_VI, AC_BE, and AC_BK. The access category AC_VO is the category for voice tasks and has the highest priority. AC_VI is the category for video and has the second-highest priority. AC_BE is the category for best-effort tasks and has the third-highest priority. AC_BK is the category for background tasks and has the lowest priority. The GSPN model for analyzing mean delay and its jitter for the AC_VO task is shown in Fig.4 (Ikeda et al., 2005) (Tsunoyama et al., 2008). The model is constructed based on the tagged task approach in order to decrease the increase in the number of states in the Markov chain.




(a) Target host part.




(b) Nontarget host part.

Fig. 4. GSPN Model of AC_VO in IEEE802.11e WLAN.

In this example, the mean delay and its jitter are analyzed for the AC_VO task generated from a host. In the analysis, the AC_VO task is called the tagged task, and the host is called the target host. Fig.4 (a) shows part of the model and represents the behavior of the tasks from the target host. The right part of the figure represents the interaction between the tasks of the other access categories in the target host and the tasks from the nontarget hosts in the WLAN. Fig.4 (b) also shows part of the model and represents the behavior of tasks from the nontarget hosts in the WLAN.

When an AC_VO task is generated in the target host, the transition *T_gen_vo* fires, and a token moves from *P_gen_vo* to *P_back_vo*. After the back-off time, *T_back_vo* fires and the token moves to *P_trans*. If no task is being sent from the nontarget hosts, the token moves to *P_trans_succ* and also moves back to *P_gen_vo*, since no collision occurs. If another task is being sent from the nontarget hosts, the token moves to *P_timeout* and moves to *P_trans_fail* after the time determined by the firing rate for *T_timeout*. When a task with another access category is generated from the target host, the transition *T_gen_q* fires and a token moves to *P_back_q*. The collision is examined by *T_fail* and *T_timeout,* as with AC_VO.

## 4. Evaluation Method

### 4.1 Delivery path and its selection probability

The delay time for task processing can be obtained by accumulating the sojourn time for states in a state sequence from a start state, where the task occurs, to an end state, where the task has been processed and delivered successfully. A reduced reachability graph is equivalent to a state diagram of a Markov chain for task processing. Thus, the delay time can be obtained from the firing rate of transitions in the path corresponding to the state sequence. A path in a reduced reachability graph is defined by the following definition. In the definition, $m_i (a \le i \le b)$ are markings and $t_j (\alpha \le j \le \beta)$ are transitions.

**[Definition4]**

A sequence of markings and transitions, $m_a[\ t_a > \ldots\ m_c > t_\beta > m_b]$, starting at marking $m_a$ and ending at marking $m_b$, for a reduced reachability graph is called a path from $m_a$ to $m_b$. The number of paths from $m_a$ to $m_b$ is denoted by $N_{ab}$, while the $i^{th}$ path is denoted by $P_{ab}{}^{(i)}$ ($1 \le i \le N_{ab}$).

When there are a number of paths from start marking $m_a$ to end marking $m_b$, task processing is made along one of the paths with the given probability. The probability of a path selected in all paths from $m_a$ to $m_b$ is called the path selection probability and is denoted by $P_r(P_{ab}{}^{(i)} \mid m_a)$, where $1 \le i \le N_{ab}$ .

The probability of transition from marking $m_j$ to next marking $m_k$ is determined by the following equation, where $A_j$ is the set of subscripts of outgoing arcs from the marking $m_j$ (Marson et. al., 1995).

$$\Pr(m_j \to m_k) = \frac{\lambda_j}{\Lambda_j}, \Lambda_j = \sum_{l \in A_j} \lambda_l \tag{4}$$

The path selection probability for path $P_{ab}{}^{(i)}$ is obtained by the product of the above probabilities for a path and given by the following lemma (Kumagai et al., 2003).

**[Lemma1]**

$$\Pr(P_{ab}{}^{(i)} \mid m_a) = \prod_{j=a}^{c} \frac{\lambda_{n_j}}{\Lambda_j} \tag{5}$$

### 4.2 Sojourn Time for the Path and Delay Jitter

The sojourn time for a path is given by the summation of the sojourn time for all markings in the path. Therefore, the probability density function of the sojourn time for a path can be obtained by the convolution of the probability density function of the sojourn time for every marking in the path. The probability density function of sojourn time, $\tau_{ab}^{(i)}$, for path $P_{ab}{}^{(i)}$ can be obtained using Equation (5) and Assumption 2. The result is given by the following lemma (Kumagai et al., 2003).

**[Lemma2]**

$$f_{\tau_{ab}^{(i)}}(t) = -\prod_{j=a}^{b}(-\Lambda_j)\left\{\sum_{m=a}^{b}\frac{\exp(-\Lambda_m t)}{\prod_{\substack{n=a \\ n\neq m}}^{b}(\Lambda_m - \Lambda_n)}\right\} \tag{6}$$

The mean value $E_\tau$ and the variance $V_\tau$ of the delay time can be obtained from Equation (6). The following results are presented as a theorem: (Ikeda et al., 2005;Kumagai et al., 2003).

**[Theorem1]**

$$E_\tau = \sum_{a \in S_{gen}} \Pr(m_a) \sum_{i=1}^{N_{ab}} \left\{ \Pr(P_{ab}^{(i)} \mid m_a) \sum_{j=a}^{b} \left( \frac{1}{\Lambda_j} \prod_{\substack{k=a \\ k \neq j}}^{b} \frac{-\Lambda_k}{\Lambda_j - \Lambda_k} \right) \right\} \tag{7}$$

$$V_\tau = \sum_{a \in S_{gen}} \Pr(m_a) \sum_{i=1}^{N_{ab}} \left[ \Pr(P_{ab}^{(i)} \mid m_a) \left\{ \sum_{j=a}^{b} \left( \frac{2}{{\Lambda_j}^2} \prod_{\substack{k=a \\ k \neq j}}^{b} \frac{\Lambda_k}{\Lambda_j - \Lambda_k} \right) - {E_\tau}^2 \right\} \right] \tag{8}$$

### 4.3 Evaluation procedure

Fig.5 shows a flow chart for evaluation. A network is first modeled using GSPN. The GSPN model is then analyzed and a reachability graph is obtained using the Petri Net tool, Time Net (German et al., 1995). The set of start markings is extracted from the reachability graph, and the delivery paths are searched. The delay time and its jitter are calculated for all searched delivery paths.

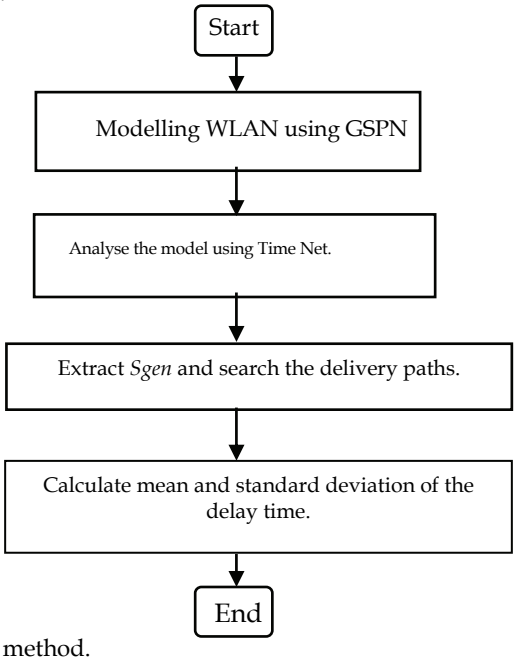



Fig. 5. Flow chart of the method.

## 5. Example

An example network using IEEE802.11e over the IEEE802.11a consisting of three hosts is evaluated. Table 1 shows the parameters for the simulation.

| Access Categories | AIFSN | CW min | CW max | TXOP Limit |
|---|---|---|---|---|
| AC_BK | 7 | 15 | 1023 | 1 frame |
| AC_BE | 3 | 15 | 1023 | 1 frame |
| AC_VI | 2 | 7 | 15 | 3 ms |
| AC_VO | 2 | 3 | 7 | 1.5 ms |

Table 1. Parameters for the ACs.

Each AC has four parameters, and ACs are distinguished by assigning different values to the parameters. Table 1 shows the default values for the parameters. A smaller value implies a higher priority. In the example, AC_VO is first analyzed by assigning a tagged task, and then AC_VI is analyzed.

Figs.3 and 4 show the mean delay and jitter for AC_VO and AC_VI, respectively. The figures show that the mean delay for AC_VI increases by about 7.5 [ms] and the jitter for AC_VI increases by about 4.3 [ms] when the virtual load on the network increases from 0.1 to 10.0 . However, when the virtual load increases, the mean delay and jitter for AC_VO decrease by about 1 [ms] less than AC_VI (Ikeda et al., 2005) (Tsunoyama & Imai 2008).

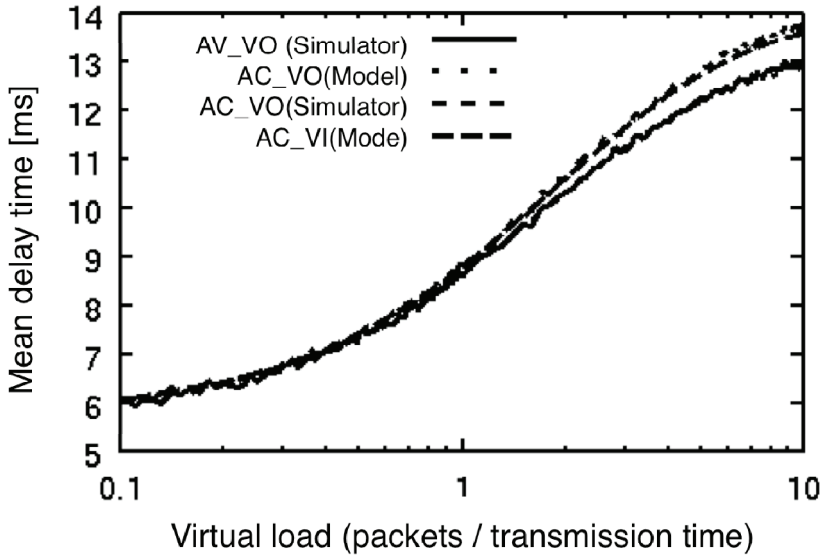



Fig. 6. Mean delay time for AC_VO and AC_VI.

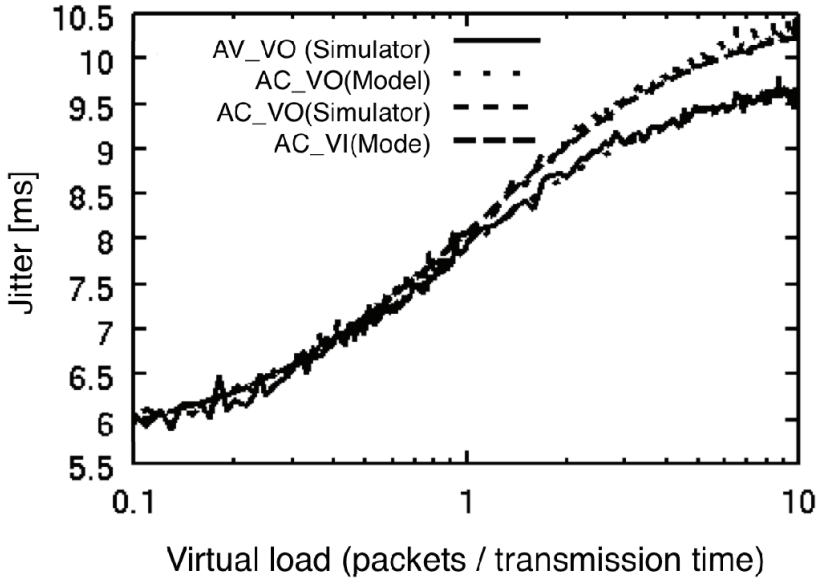



Fig. 7. Jitter for AC_VO and AC_VI.

## 6. Conclusions

A method for modelling local area computer networks used for processing and delivering multimedia data is proposed. The proposed method can evaluate the mean delay time and its jitter (standard deviation) for systems based on the GSPN model and tagged task approach. The systems can be modeled by the method presented, and both of the values can be evaluated easily using the equations shown in this chapter. An example of modeling and evaluating local area computer networks using IEEE802.11e WLAN supporting EDCA was shown. From the results, it can be concluded that the system can be modeled easily. The mean delay and jitter for AC_VO obtained using the proposed method agrees well with the values obtained using simulations. However, when the virtual load of the network exceeds one, the value of the jitter for AC_VI differs slightly from that by simulation.
Future efforts will improve the model to reduce the observed difference and to compose a compact model to reduce the number of states in the Markov chain for the network.

## 7. Acknowledgements

The authors would like to thank Messrs. Kumagai, Ikeda, and Maruyama for their helpful discussions and comments. The authors would also like to thank Professor Ishii and Professor Makino for their helpful comments.

## 8. References


Adachi, N.; Kasahara, S. ; Asahara Y. & Takahashi, Y. (1998). Simulation Study on Multi-Hop Jitter Behavior in Integrated ATM Network with CATV and Internet, *The Transactions of the Institute of Electronics, Information and Communication Engineers*, Vol. E81-B, No.12, pp.2413-2422.

Ahmad, S. ; Awan, I. & Ahmad, B. (2007). Performance Modeling of Finite Capacity Queues with Complete Buffer Partitioning Scheme for Bursty Traffic, *Proceedings of the First Asia International Conference on Modeling & Simulation (AMS'07)*, pp. 264-269.

Althun, B. & Zimmermann, M. (2003). Multimedia Streaming Services: Specification, Implementation, and Retrieval, *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval*, pp.247-254.

Bin, S.; Latif, A.; Rashid, M.A. & Alam, F. (2007). Profiling Delay and Throughput Characteristics of Interactive Multimedia Traffic over WLANs Using OPNET, *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)* , pp. 929-933.

Cheng, S.T. & Wu, M. (2005). Performance Evaluation of Ad-Hoc WLAN by M/G/1 Queuing Model, *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)*, Vol. II, pp. 681-686.

Claypool, M. & Tanner, J. (1999). The Effect of Jitter on the Perceptual Quality of Video, *ACM Multimedia '99*, pp.115-118.

Fan, Y.; Huang, C.Y. & Tseng, Y.L. (2006). Multimedia Services in IEEE 802.11e WLAN Systems, *Proceedings of the 2006 International Conference on Wireless communications and mobile computing*, pp.401 – 406.

Ferguson, P. & Huston, G. (1998). *Quality of Service: Delivering QoS on the Internet and in Corporate Networks*, John Wiley & Sons, Inc.

German, R. (2000). *Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets*, John Wiley & Sons, Inc.

German, R.; Kelling, C.; Zimmermann, A. & Hommel, G. (1995). TimeNET-a toolkit for evaluating non-Markovian stochastic Petrinets, *Proceedings of the Sixth International Workshop on Petri Nets and Performance Models*, pp.210-211.

Gibson, L. & David, R. (2007). Streaming Multimedia Delivery in Web Services Based e-Learning Platforms, *Proceedings of the IEEE International Conference on Advanced Learning Technologies*, pp. 706-710.

Grinnemo, K.J. & Brunstrom, A. (2002). A Simulation Based Performance Analysis of a TCP Extension for Best-Effort Multimedia Applications, *Proceedings of the 35th Annual Simulation Symposium*, pp.327.

IEEE Standards Board (2003). *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: Medium Access Control (MAC) Enhancements for Quality of Service (QoS)*, IEEE Draft 802.11e, Rev. D5.1.

Ikeda, N.; Imai, H.; Tsunoyama, M. & Ishii, I. (2005). An Evaluation of Mean Delay and Jitter for 802.11e WLAN, *Proceedings of the Fourth IASTED International Conference on Communication Systems and Networks*, pp.202-206, Sept.

Imai, H.; Tsunoyama, M.; Ishii, I.; & Makino, H. (1997). An Analyzing Method for Tagged-T ask-Model, *The Transactions of the Institute of Electronics, Information and Communication Engineers D-I*, Vol.J80-D-1, No.10, pp.836-844.

Imai, H.; Tsunoyama. M.; Ishii, I. & Makino, H. (1999). Modeling and Evaluating Computer Systems for Multimedia Data Processing, *Proceedings of the Eighteenth IASTED International Conference on Modeling, Identification and Control*, pp.353-358.

Kornkevn, S. & Lilleberg, N. (2002). Enhancing support and learning services for instructors and students who engage in course-related multimedia and web projects, *Proceedings of the 30th annual ACM SIGUCCS Conference on User services*, pp.56-59.

Kumagai, K.; Tsunoyama, M.; Imai, H., & Ishii, I. (2003). An evaluation Method for Network Systems based on Delay Jitter Analysis", *Proceedings of the 4-th EURASIP Conference focused on Video/Image Processing and Multimedia Communications*, pp.569-574.

Marson, M.A.; Balvo, G.; Conte, G.; Donatelli, S. & Franceschinis, G. (1995). *Modeling with Generalized Stochastic Petri Nets*, John Wiley & Sons, Inc.

Nerjes, G.; Muth, P. & Weikum, G. (1997).Stochastic Performance Guarantees for Mixed Workloads in a Multimedia Information System, *Proceedings of the 7th International Workshop on Research Issues in Data Engineering (RIDE '97)*, pp. 131.

Park, S. (2006). DiffServ Quality of Service Support for Multimedia Applications in Broadband Access Networks, *Proceedings of the 2006 International Conference on Hybrid Information Technology*, Vol.2, pp. 513-518.

Shahraray, B.; Ma, W.Y.; Zakhor, A., & Babaguchi, N. (2005). Mobile Multimedia Services, International World Wide Web Conference, pp.795-795.

Tsunoyama, M. & Imai, H. (2008). An Evaluation Method for Delay Time and Its Jitter of WLAN using GSPN Model, *Proceedings of the 8th International Workshop on Wireless Local Networks (WLN2008)*, pp.811-812.

Villalon, J.; Mico, F.; Cuenca, P. & Barbosa, L.O. (2005). QoS Support for Time-Constrained Multimedia Communications in IEEE 802.11 WLANs, *Proceedings of the A Performance Evaluation, Systems Communications (ICW'05, ICHSN'05, ICMCS'05, SENET'05)*, pp. 135-140.

2

# Architecture of Computer Intrusion Detection Based on Partially Ordered Events

Liberios Vokorokos and Anton Baláž
*Technical University of Košice*
*Slovak Republic*

## 1. Introduction

Information technologies became part of our daily life. Nowadays, contemporary society is dependent on functioning of miscellaneous information systems providing daily community motion. The attack aim is often to disrupt, deny of service or at least one of its parts required for proper functionality, or to acquire unauthorized access to information [Vokorokos (2004)]. Nowadays, solid system assecuration becomes one of the main priorities. Basic way of protection is realized through specialized devices firewalls allowing to define and control permitted communications in boundary parts of computer network or between protected segments and surrounding environment. Present firewalls often detect some unauthorized attack activities but their functionality is limited. Unauthorized intrusion detection systems allow increase of information systems security against attacks from the Internet or organization intranet, by means of passive inform about arising intrusion or active interfere against defecting intrusion.
The existing intrusion detection approaches can be divided in two classes - anomaly detection and misuse detection [Denning (1987)]. The anomaly detection approaches the problem by attempting to find deviations from the established patterns of usage. On the other hand, the misuse detection compares the usage patterns to known techniques of compromising computer security. Architecturally, the intrusion detection system(IDS) can be categorized into three types - host-based IDS, network-based IDS and hybrid IDS [Bace (2000)]. The host-based IDS, deployed in individual host-machines, can monitor audit data of a single host. The network-based IDS monitors the traffic data sent and received by hosts. The hybrid IDS uses both methods. The intrusion detection through multiple sources represents a difficult task. Intrusion pattern matching has a non-deterministic nature where that same intrusion or attack can be realized through various permutations of the same events. The purpose of this paper is to present authors' proposed intrusion detection architecture based on the partially ordered events and the Petri nets.
Project is proposed and implemented at the Department of Computers and Informatics in Košice supported by VEGA 1/4071/07. (Security architecture of heterogeneous distributed and parallel computing system and dynamical computing system resistant against attacks) a APVV 0073-07 (Identification methods and analysis of safety threats in architecture of distributed computer systems and dynamical networks).

## 2. State of art

Several intrusion detection systems were designed and implemented till today. Most of these systems are based on statistical methods derived from work of Denning [Denning (1987)]. Some of them, as source of information, use log system of operation system [Anderson et al. (1995)]. Other one, as input data, use network traffic [Zhang et al. (2003)] [Spirakis et al. (1994)] [Servilla (1990)]. Systems, as MADIDS [Guangchun et al. (2003)], extend this network traffic with distribution of intrusion data within single analyzing network systems that perform partial intrusion detection. Among systems not working with statistical methods, there can be inserted system of authors [Teng et al. (1990)] that analyzes single user events and tries to find mutual relations among them. IDS architectures based on misuse detection are systems, as [Ilgun et al. (1995)] [Ilgun (1993)], that search for already known intrusions, derived state of intrusion based on present system state.
According to present state of intrusion detection systems, this work is focused on intrusion detection and system penetration variability, which can reduce time needed to evaluate potential intrusion.

## 3. Architecture of designed IDS system

Proposed system architecture includes part of planning and matching, figure 1. The matching means that the system gets into a state of intrusion when a sequence of events leading to the mentioned state occurs. The intrusion is a system state which overtakes previous states represented by particular system events. If there exists such a fine-grained log system, it is possible to detect the states with intrusion. Single attacks to the systems represents mentioned single events that in a final implication leads to the state of intrusion. Characteristic feature of intrusions is their variability; permutation of same events leads to same state of intrusion. Single intrusions are characteristic with their non-determination. Designed IDS system solves this problem with planning [Russell & Norvig (2003)] that responds to lay-out of possible sequence of steps leading to the final intrusion. Planning part creates the intrusion plan by first-order logic when it describes known activities and disturber's goals to specify attack sequences. Result of planning is intrusion specification and its single steps that uses the matching part of the system to the intrusion detection provided by Petri Net automata. System architecture designed on the Department of Computers and Informatics is on the figure 1.

## 4. Partially ordered state analysis

One of the main problems related to the intrusion detection of the system refers to the variability of possible attacks. It is possible to realize the same attack by many ways. Suggested IDS architecture uses the analysis of partially ordered states in a difference from the classical analysis of the transition by the states of the monitored system. In the classical scheme of the state analysis [Axelsson (2000)], the attacks are represented as a sequence of the transition states. States in the scheme of the attack correspond with the states of the system that have their Boolean statement related to these states. These expressions must fulfill the conditions to realize the transition to the next state. The constituent next states are interconnected by the oriented paths that represent events or conditions for the change of the states. Such a state diagram represents the actual state of the monitored system. The change of the states considers about the intrusion as the event sequence that is realized by the attacker. These events start in the initial state and end in the final compromised state. The initial state represents the states of the system before starting the penetration. The final compromised state

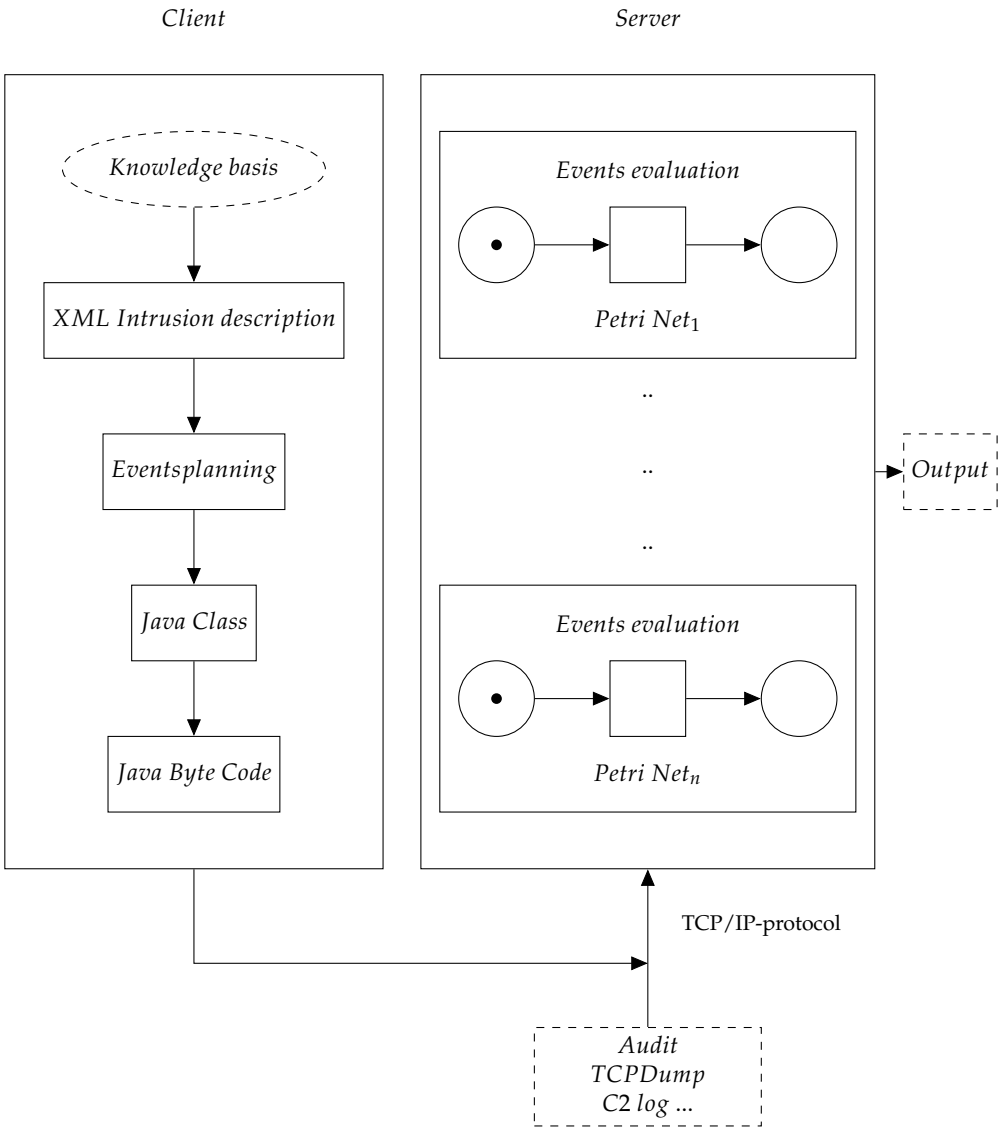



Fig. 1. Architecture of Designed IDS System

represents the state of the system which follows from the finished penetration. The transition of the states that the intruder must do for the achievement of the final result of the system intrusion, are among the initial and final states. In the figure 2 there is an example of the attack that consists of four states of the attack.

Classical method of the state transition [Anderson (1980)] strictly analyzes intrusion signatures as ordered sequence of states without any chance of overlaying sequence of single

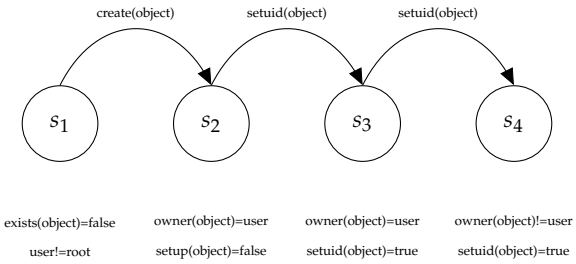



Fig. 2. State Transition Diagram

events. Designed IDS architecture increases the flexibility of states analysis by using partially ordered events. Partially ordered events specify option when the events are ordered one according to another while the others are without this option of ordering. Analysis of partially ordered states enables several event sequences to form one state diagram. By using partially ordering against total ordering it is possible to use only one diagram to representation permutation of the same attack. In the proposed architecture partially ordered state transitions are generated by partially ordered planner. Representation by partially ordered plan is more indicating according to total ordered form of states. It enables planner to put off or to ignore unnecessary ordering selection. During the state transition analysis, the number of total ordering increases exponentially with increasing the number of the states. This property of complexity coupled with total ordering is eliminated in case of partially ordered planning. Applying partially ordered notification and its property of decomposition, it is possible to deal with complex domains without any exponential complexity. Partially ordered planner seeks state space of plans in contrast to state space of cases. The planner begins with a simple, incomplete plan that is extended in sequence by planner till it gets complete plan of solution of the problem. The operators in this process are operators on the plans: addition of steps, instructions appointing order of one step before another and other operations. The result is final plan of order of particular states based on the dependence within these states. The acquired representation allows through the partly ordered plans to operate a broad range of troubleshooting domains in the planner as well as systems of intrusion detection. The partly ordered scheme provides more exact representation of intrusion patterns as the completely ordered representation, because only inevitable dependencies are considered within particular events. figure 3 is the only dependency between operations touch and chmod.

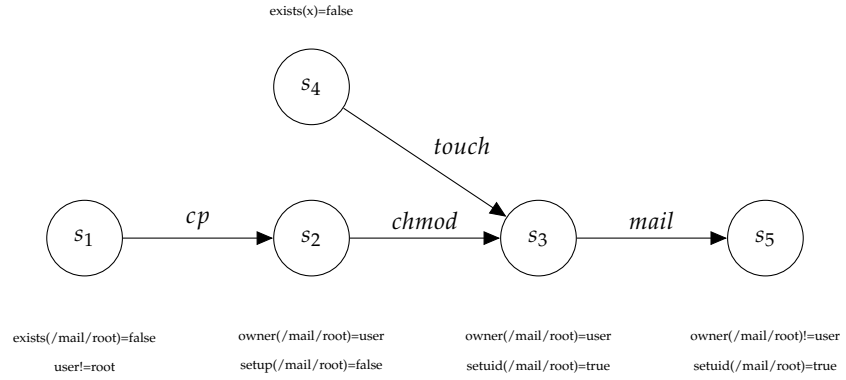



Fig. 3. Partially Ordered Intrusion States

In the figure 2, it is not clear which dependencies are necessary within single states. Whereas in the figure 3, it is clear which events fore come which. Compromised state in the figure 3 is possible represented by the first-order logic as:

$$\begin{aligned}
&\exists\ /var/spool/mail/root\ x \\
&\quad /var/spool/mail/root \in x \wedge \\
&\quad owner(/var/spool/mail/root) = root \wedge \\
&\quad setuid(/var/spool/mail/root) = enable \\
&\quad \Rightarrow compromised(x) = true
\end{aligned}$$

Proposed approach of intrusion analysis outcomes from the demand assumption of identification of minimal set of intrusion signatures and necessary dependencies within these signatures. Minimal set of signatures assumes the elimination of irrelevant signatures that do not create the intrusion. A possible example of attack, creating a link to file of different owner with different rights with consequential executing link and obtaining rights of original owner:

1. *ls*
2. *ln*
3. *cp*
4. *rm*
5. *execute*

The first, third and fourth commands do not have an influence on the attack; tendency is to mask the attack. By elimination of these commands, it is possible to get minimal set describing attack together with single dependencies within events. Example in form of the first-order logic:

$$
\begin{aligned}
&\exists\, file1, file2, x \\
&\quad owner(file1) \neq x \wedge \\
&\quad owner(file2) = x \wedge \\
&\quad ln(file2, file1) \wedge \\
&\quad execute(file2) \wedge \\
&\quad ln(file2, file1) \prec execute(file2) \wedge \\
&\quad \Rightarrow compromised(x)
\end{aligned}
$$

## 5. Intrusion signature sequence planning

Intrusion is defined as a set of events with a focus on compromise integrity, confidentiality and resources availability. Designed architecture of IDS includes the planning part to construct event sequence plan of which consists the intrusion. Planning includes goals, states and events. According to what is necessary to do in final plans, planning combines actual environment state with information depending on the final result of events.
State transition is characterized as a sequence of events performed by intruders leading from initial state do final compromised state. Planning can be formulated as a problem of state transition:

- *Initial state:* Actual state description.
- *Final state:* Logical expression of concrete system state.
- *Intrusion signatures:* Events causing change of a system state.

Planning is defined as:

1. Set of single steps of the plan. Every step represents control activity of the plan.
2. Set of ordered dependencies. Every dependency is in a form of $S_i < S_j$, where, step $S_i$ is executed before $S_j$.
3. Set of variable bindings. Every binding is in a $v = x$ form, where $v$ is a variable in some step and $x$ is a constant or other variable.
4. Set of causal bindings. Causal binding is in a form of $S_i \xrightarrow{c} S_j$. From state $S_i$ by auxiliary $c$ state $S_j$ , where $c$ is a necessary pre-condition for the $S_j$.

Each signature has an associated pre-condition that indicates what has to be completed before it is possible to apply event bind with the signature. Post-condition expresses event result connected to the intrusion signature. A task of the planning is to find events sequence responsible for the intrusion. The goal of planning in the designed IDS architecture is to find event sequence and their dependencies and construct result sequence of an intrusion. Partially ordered planning allows representing plans in which some steps are ordered according to other steps. Intrusion signatures and their nature of non-determination are suitable for fundamentals of partially ordered planning. Planning consists of database of intrusions and events planner - figure 1. Knowledge base includes information about each intrusion signature including pre and post conditions of these events in the form of first-order logic. The planner generates set of events and their dependencies for each initial and final intrusion state. Furthermore, knowledge base includes state dependencies for each event signature. This

information is used by planner for defining partially ordering in between intrusion signature. For instance pre-condition intrusion signature consists of k terms. These are represented in form of symbols

$$\{PS_1, PS2, \ldots, PS_k\} \wedge \{PS_j < PS_k\} \wedge \ldots \wedge \{PS_l < PS_m\}$$

An algorithm of partially ordered planning begins with minimal plan and in each step this plan is extended through available pre-condition step. This is realized by selecting intrusion signature that fulfill some of the unfulfilled pre-conditions in the plan. For a newly fulfilled pre-conditions of event signatures are causal bindings stored in between them. These bindings are necessary for partially event ordering. An ordering result is represented by set of events and their dependencies in between these event signatures. Let intrusion sequence to consist of $n$ event signatures: $SA_1, SA_2, \ldots, SA_n$, then intrusion structure is specified as

$$\{SA_1, SA_2, \ldots, SA_k\} \wedge \{SA_j < SA_k\} \wedge \ldots \wedge \{SA_l < SA_m\}$$

First part of this term $\{SA_1, SA_2, \ldots, SA_k\}$ is a set of event signatures. Next part of the term is ordering dependency between signatures. The intrusion example referring to figure 3 is specified as

$$\{cp, chmod, touch, mail\} \wedge \{cp < chmod\} \wedge \{chmod < mail\} \wedge \{touch < mail\}$$

Each formulation represents an intrusion signature variation that leads to the same compromised states. In the case of the intrusion signatures it is necessary to deliberate this intrusion variability from the view of memory requirements. Further, if it comes to the alternation of initial state, it may have a consequence of complete intrusion plan alternation. The next advantage of partially ordered planning is that the time between two intrusion signatures does not have an influence on the analysis during capturing system data and state changing.

### 5.1 Events planning

This session represents planning algorithm in the designed IDS that's result is partially ordered events of intrusion signature. It is possible to represent the intrusive plan through the triple $\langle A, O, L \rangle$, where $A$ is a set of events, $O$ is a set of ordered dependencies on the $A$ set, and $L$ is a set of casual connections. The planner starts its activity with a blank plan, and it specifies this plan in stages with being obligated to consideration of consistence requirements defined in the $O$ set. The key step of this activity is to preserve states of the past conclusions and requirements for these conclusions. For the provision of consistence within various events, the recording of relations within the events is performed through the casual connections. Casual connection is a structure consisting of two references to plan events (producer $A_p$ and consumer $A_c$), and $Q$ assertion that is the result of the $A_p$ and the $A_c$ precondition. The expression is represented by $A_p \xrightarrow{Q} A_c$ and connections themselves are stored in the $L$ set. Casual connections are used for the detection of interference within new and old conclusions. Marked as threats. This means that $\langle A, O, L \rangle$ represents a plan and $A_p \xrightarrow{Q} A_c$ is a connection in $L$. Let the $A_t$ be another event in $A$, than the $A_t$ endangers the $A_p \xrightarrow{Q} A_c$ if:

- $O \cup \{A_p < A_t < A_p\}$ a
- $A_t$ has $\neg Q$ as the result

If the plan involves threats, it cannot suit the scheduled requirements defined in $\langle A, O, L \rangle$. The threats must be considered by the planner during assembling the final plan. The algorithm can add supplementary order dependencies by assurance of performance of $A_t$ before the $A_p$. The core of the planning is represented by the algorithm of planning, mentioned below, that searches the state environment of the plans. The algorithm begins with a blank plan and performs non-deterministic selection of the event sequence in stages, till all the preconditions are considered through their casual connections and till potential threads of the plan are eliminated. Partially ordered dependencies of the final plan are over again represented by only partially ordered plan, that resolves the problem of planning. The algorithm arguments are the planning structure, and the plan agenda. Each agenda item is a pair $\langle Q, A \rangle$, where $Q$ is a conjunction of $A_i$ preconditions.

Planning ($\langle A, O, L \rangle$ , $agenda, \Lambda$)

1. ***Completion:***If the agenda is empty, return $\langle A, O, L \rangle$.
2. ***Target selection:*** $\langle Q, A_{need} \rangle$ is a pair in the agenda (according $A_{need} \in A$ and $Q$ is conjunction of preconditions $A_{need}$).
3. ***Event selection:*** $A_{need} = event\ selection$ that adds to $Q$ one of the new events from $\Lambda$, or the event already in $A$, possible to be ordered according to $A_{need}$. If there does not exist any of the mentioned events, return error. Let $L^{'} = L \cup \{A_{add} \overset{Q}{\rightarrow} A_{need}\}$, and $O^{'} = O \cup \{A_{add}\}$ and $O^{'} \cup O^{'}\{A_0 < A_{add} < A_\infty\}$ otherwise $A^{'} = A$).
4. ***Update set of events:*** Let $agenda^{'} = agenda - \{\langle Q, A_{need} \rangle\}$. If $A_{need}$ is a new instance, than for each conjunction $Q_i$ of its precondition add $\langle Q_i, A_{add} \rangle$ to $agenda^{'}$.
5. ***Protection of casual connections:*** For each operation $A_t$, that can threaten the casual connections $A_p \overset{R}{\rightarrow} A_c \in L^{'}$ select consistent ordered dependencies:
   - *Factorization :* Add $A_t < A_p$ to $O^{'}$
6. ***Recursive calling***:
   Planning ($\left\langle A^{'}, O^{'}, L^{'} \right\rangle$ , $agenda^{'}, \Lambda$)

Result of the planning algorithm is the plan of partially ordered events, that considerates possible variations of the described attack.

### 5.2 Events evaluation

Petri Nets represent automatas based on events and conditions. Events are actions that are executed and their existence is controlled by system states. Every system state represents set of conditions and their values. In the proposed IDS system, there are these sets in form of first-order expressions presenting fulfilled or not fulfilled conditions. Some of the events only occur on specific conditions where state description represents preconditions for those events. Presence of specific events may terminate validity of one precondition and setup validity of other one.

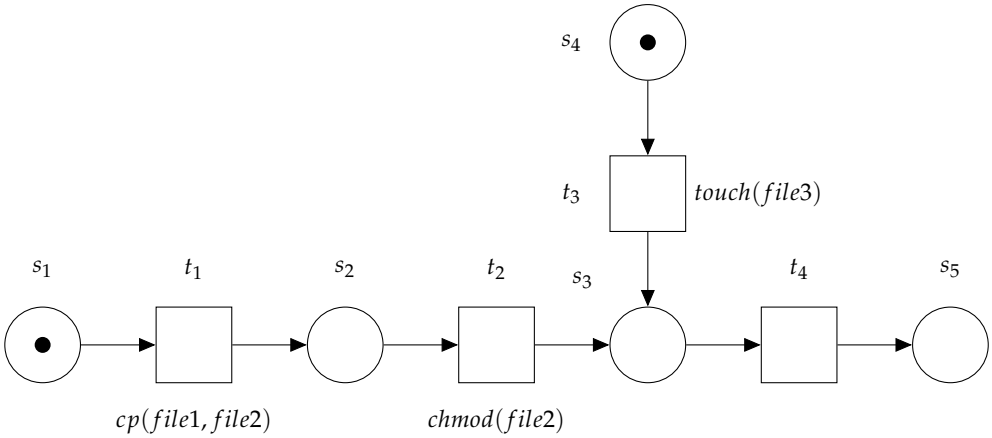



Fig. 4. Petri Net Intrusion Example

Each intrusion is in the proposed IDS system represented by a Petri Net. Petri Net places represent states or pre - post events conditions. Input for Petri Net creation is plan of partially ordered events forming intrusion. Petri Net transitions correspond with characteristic event pattern. Detection architecture evaluates single intrusions in form of Petri Nets evaluating input events of miscellaneous input data. Initial states represent initial system states and final state represents state that implies intrusion.

## 6. Experimental validation of proposed IDS

Presented architecture of IDS system is implemented in Java. The goal of this implementation is to generate a uniform set of classes that can be used for general generation of IDS. By the designed architecture, there exist two critical points of the system that affect the efficiency of entire sample intrusion evaluation. Given points:

1. Time needed for capture and generation of instance of an input event into the object of Java language
2. Time needed for evaluation of the single intrusion represented through the Petri net

On the basis of these two critical points, there were assembled and executed following experiments consisting of network attacks:

**Experiment 1** Time needed for generation of input event object and saving already processed input events on the list. The experiment was performed in the network environment with various types of attacks on the TCP/IP protocol. An input event flow included 2500 (5000) packets. In order to elimination of possible external influences and to achieve more objective results, the test was performed with 300 iterations.

The testing environment consists of various computer systems mentioned in table 1. Designed IDS system presents type of host IDS, but from the implementation perspective, single input objects of input events are transferred though the TCP/IP protocol on basis of Client-Server type, where both Server (accepts and handles information) and Client (sends input objects

for evaluation) operate. Therefore, tests containing experiments performed in the network environment on basis of Ethernet were added to testing formation as well. The test results of single configurations are in figures 5 and 6. Average times needed for instance of one input event generation are mentioned in tables 2 and 3 according to:

$$\overline{Time} = \frac{Time\ for\ creating\ (2500\ or\ 5000)\ packets}{2500\ or\ 5000}[ms] \tag{1}$$

| Number | Configuration |
|---|---|
| 1. | AMD Duron 800MHz, 512MB SDRAM |
| 2. | Intel Celeron 2.4GHz, 512MB DDRAM |
| 3. | AMD Sempron 2.0Ghz, 512MB DDRAM |
| 4. | Intel P4 2.4GHz HT, 1GB DDRAM |
| 5. | AMD Opteron 2.21GHz, 1GB DDRAM |
| 6. | Ethernet 100Mbit |
| 7. | Ethernet 1000Mbit |

Table 1. Testing Configuration of Computer Systems

| Description | Average time 2500 packets [ms] |
|---|---|
| Ethernet 1000Mbit | 0,200496 |
| Ethernet 100Mbit | 0,815273469 |
| Intel Celeron | 0,2646 |
| Amd Duron | 1,712097959 |
| Amd Opteron | 0,194256 |
| Intel Pentium 4 | 0,813512 |
| Amd Sempron | 0,274432653 |

Table 2. Average Time Need for Generation of One Instance of Input Event

| Description | Average time 5000 packets [ms] |
|---|---|
| Ethernet 1000Mbit | 0,229192 |
| Ethernet 100Mbit | 0,94105 |
| Intel Celeron | 0,496812 |
| Amd Duron | 0,8239 |
| Amd Opteron | 0,158636 |
| Intel Pentium 4 | 0,578310204 |
| Amd Sempron | 0,409856 |

Table 3. Average Time Need for Generation of One Instance of Input Event

Experiment 1 was focused on speed of transformation flow of input events into object instance at Java language. Within simulation, it was detected that the best results are provided by performance the speediest platform AMD Opteron and the weakest performance from the set of testing systems is provided by AMD Duron. To consider input event transfer and transformation into input object, the transfer of packets through the TCP/IP protocol is the

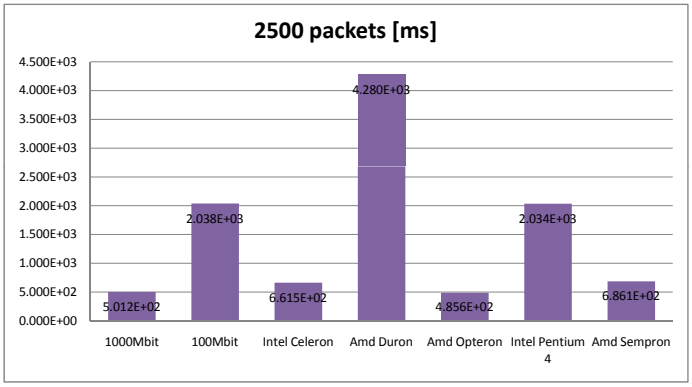



Fig. 5. Results of Experiment 1

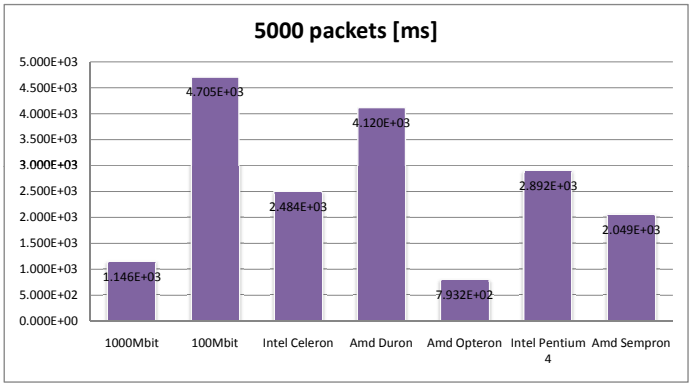



Fig. 6. Results of Experiment 1

most decisive factor. This determination results from comparison of the speediest platform AMD Opteron results and data transfer in Ethernet 1000Mbit network type, where the results of these two simulations are very similar.

**Experiment 2** Time needed for attack evaluation at various arithmetic of attacks evaluated at the same time. Time is measured by the object of attack description transfer period till the final time of attack detection.

The experiment was performed on the same configurations mentioned in table 1. The amount of tested attacks is in range of 1 to 20 attacks evaluated at the same time. The testing input flow contains 2500 (5000) packets including packets generating attack. For more objective results acquirement, single tests were performed 300 times repeatedly. The acquired results are displayed in graphs 7, 8, 9, 10 and 11 Summary of experiment 2 results is displayed in graphs 12 and 13.

Results of the simulation experiments were realized on the group of various performance platforms. In order to test performance, the system was implemented in Java language. Development environment was IDE Eclipse, operation system MS Windows XP and MS

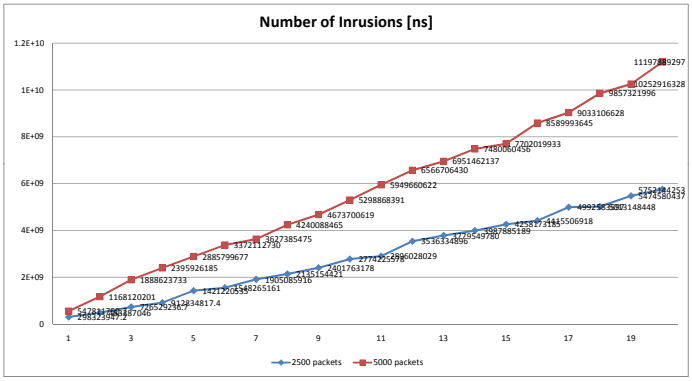



Fig. 7. Experiment 2 - Intel Celeron

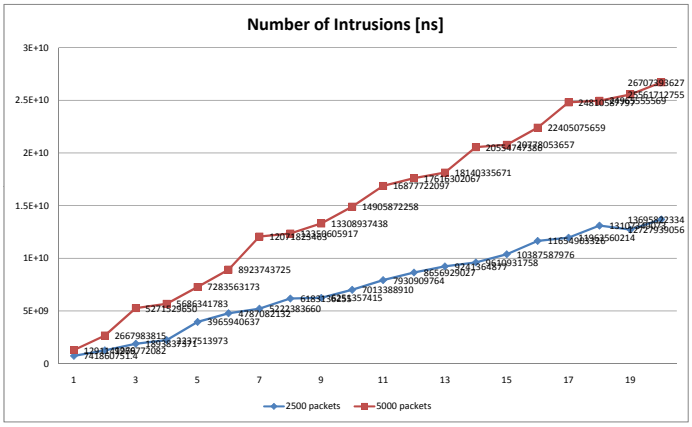



Fig. 8. Experiment 2 - AMD Duron

Windows 2003. The experiments were performed in order to performance evaluation. On the same purpose, a special group of attacks was created, focused on the limitations of the TCP/IP protocol. Single tests were executed 300 times repeatedly in order to elimination of possible fault in case of single measuring.
The results achieved during experiments mean:

- Officially, the most efficient platform AMD Opteron provides better results. The more efficient computing performance, the less the time needed for evaluation.
- At single systems (loopback), the inner interface provides approximately the same permeability as the Ethernet 1000Mbit network.
- Time needed for evaluation of the rising amount of attacks evaluated at the same time, rises linear.

On the basis of the results of the experiments, decisive and main factor of the entire designed architecture is memory subsystem of the tested computer system.
Less affecting speed factors of the architecture:

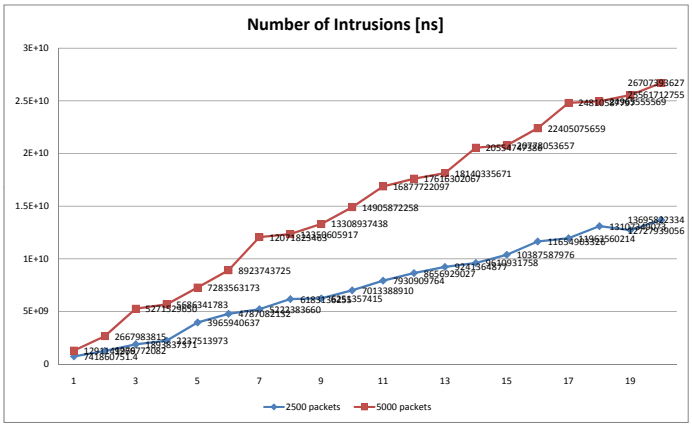



Fig. 9. Experiment 2 - AMD Opteron

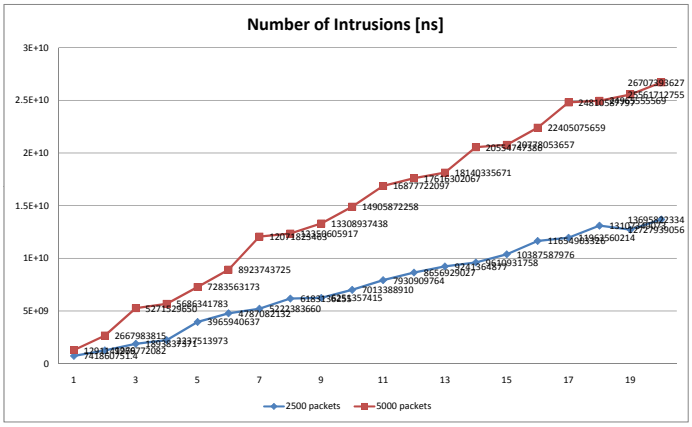



Fig. 10. Experiment 2 - Intel Pentium 4

- *Performance of CPU*. Faster processor presents faster evaluation of input flow. Cooperation customization of the memory subsystem and the processor presents narrow effectiveness socket of the entire architecture.
- *Faster logging system*. More effective retrieval of the input event means continuous processing of objects by the evaluation unit without waiting for write and read. Customization of the logging system and its effectiveness means another important effectiveness role of the entire system.
- *More effective data structures*. The system was designed during its implementation in regard of general IDS, with possibility of another expansion and specification. Effectiveness of some used data structures does not have to be optimal and it requires its profilation in order to force the entire functionality to be more effective.

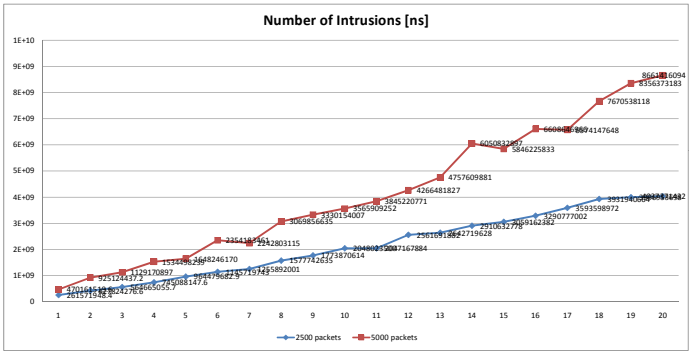



Fig. 11. Experiment 2 - AMD Sempron

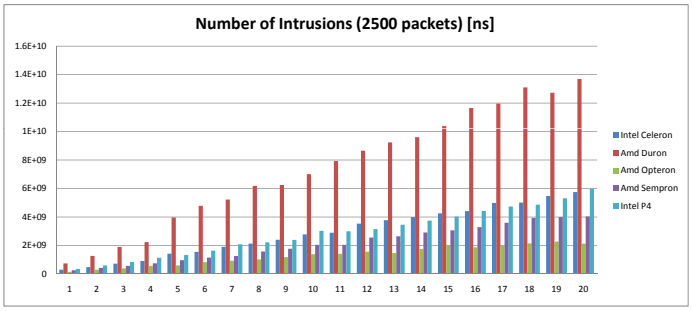



Fig. 12. Experiment 2 - Summary 2500 packets

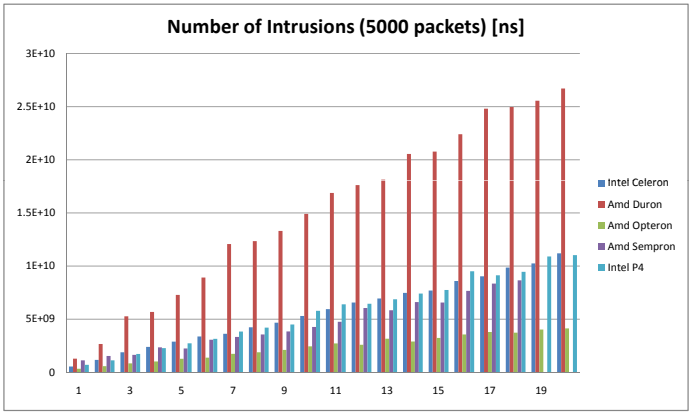



Fig. 13. Experiment 2 - Summary 5000 packets

## 7. Conclusion

Information technology security nowadays presents one of the main priorities of modern society dependent on information. Protection of data access, availability, and integrity represents basic security properties insisted on information sources. Intrusion of one of the properties mentioned above may form penetration or attack on the computer system. Within protection mechanisms, various methods providing security rules relating individual levels of possible behavior are classified. Other protection mechanisms are diversified systems detecting suspicious behavior. Intrusion detection systems belong to these systems as well.
One of the main problems of intrusion detection is potential attack variability. From the detection perspective, generation of exact intrusion attribute sequence is deficient. The property of attack sequence non-determination is not be described by the entire sequence of events forming intrusion. One of the goals of this work was to solve attack variability mentioned above. Upon attribute properties and their context research, classificatory hierarchy describing mutual references within attributes and events was formed. Analysis result is a new method of penetration representation in form of partially-ordered events scheme enabling generation of some dependencies only, within the whole set of events describing intrusion. The resultant event scheme is transformed into Petri Nets that evaluate input event flow and detect possible attributes of represented intrusions.
The aim of this work was to introduce designed intrusion detection architecture based on partially-ordered events and patterns. The main work goal was production of intrusion detection and its alternatives method. Produced method identities possible system intrusions by means of monitoring computer system state patterns. Individual states of the monitoring system are described through performed events, and individual dependencies within the performed events. The resultant detection model is realized by the Petri Nets.
Upon designed IDS system architecture, system prototype was implemented and tested. From test results, functionality and practical usability of designed IDS architecture is resulted. From experimentation conclusions, interactivity of central processing unit CPU and memory subsystem is the determining factor with influence on entire intrusion detection effectiveness. The resultant system represents live system with possibility of dynamic addition and removal of other detected intrusions.
The work is one of reached results within projects VEGA 1/4071/07 (Security architecture of heterogeneous distributed and parallel computing system and dynamical computing system resistant against attacks), and APVV 0073-07 (Identification methods and analysis of safety threats in architecture of distributed computer systems and dynamical networks) being solved at Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice.

## 8. References

Anderson, J. P. (1980). Computer security threat monitoring and surveillance, *Technical report*, James P. Anderson Co.

Anderson, Lunt, Javits, Tamaru & Valdes (1995). Detecting unusual program behavior using the statistical components of nides.
**URL:** *http://www.csl.sri.com/papers/5sri/*

Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy, *Technical Report 99-15*, Chalmers Univ.
**URL:** *citeseer.comp.nus.edu.sg/axelsson00intrusion.html*

Bace, R. G. (2000). *Intrusion detection*, Macmillan Publishing Co., Inc., Indianapolis, IN, USA.

Denning, D. E. (1987). An intrusion-detection model, *IEEE Trans. Softw. Eng.* **13**(2): 222–232.

Guangchun, L., Xianliang, L., Jiong, L. & Jun, Z. (2003). Madids: a novel distributed ids based on mobile agent, *SIGOPS Oper. Syst. Rev.* **37**(1): 46–53.

Ilgun, K. (1993). USTAT: A real-time intrusion detection system for UNIX, *Proceedings of the 1993 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, pp. 16–28.
**URL:** *citeseer.ist.psu.edu/ilgun92ustat.html*

Ilgun, K., rd A. Kemmerer & Porras, P. A. (1995). State transition analysis: A rule-based intrusion detection approach, *Software Engineering* **21**(3): 181–199.
**URL:** *citeseer.ist.psu.edu/ilgun95state.html*

Russell, S. & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*, 2. edn, Prentice-Hall, Englewood Cliffs, NJ.

Servilla, R. H. G. L. A. M. M. (1990). The architecture of a network level intrusion detection system, *Technical report*, Department of Computer Science, University of New Mexico.

Spirakis, P., Katsikas, S., Gritzalis, D., Allegre, F., Darzentas, J., Gigante, C., Karagiannis, D., Kess, P., Putkonen, H. & Spyrou, T. (1994). SECURENET: A Network Oriented Intrusion Prevention and Detection Intelligent System, *Proceedings of the 10th International Conference on Information Security, IFIP SEC94*, The Netherlands.

Teng, H. S., Chen, K. & Lu, S. C.-Y. (1990). Security audit trail analysis using inductively generated predictive rules, *Proceedings of the sixth conference on Artificial intelligence applications*, IEEE Press, Piscataway, NJ, USA, pp. 24–29.

Vokorokos, L. (2004). *Digital Computer Principles*, Typotex, Budapest.

Zhang, Y., Lee, W. & Huang, Y.-A. (2003). Intrusion detection techniques for mobile wireless networks, *Wirel. Netw.* **9**(5): 545–556.

3

# Particle Filter for Depth Evaluation of Networking Intrusion Detection Using Coloured Petri Nets

Chien-Chuan Lin and Ming-Shi Wang
*Department of Engineering Science, National Cheng Kung University*
*Taiwan*

## 1. Introduction

In this chapter, we investigated and proposed an approach that used the particle filter concept in a network intrusion detection system and simulated in the Coloured Petri Nets tools (CPN Tools) platform to trace and pre-detect networking attack and intrusion behaviors. We call this proposed approach as Network Particle Filter (NPF) scheme. We can realize what it happened by analyzing and simulating an intrusion in detail. The experimental results demonstrated that the Coloured Stochastic Petri Nets (CSPN) model approach is an efficient and helpful to evaluate an intrusion detection system in depth.

The motivation of this investigation is to consider that almost networking behavior could be marked when it has been done in a network. We are interesting to and focusing on trace and analyze the steps of networking intrusion behavior.

Any single attack or incident can be decomposed into multiple steps of intrusion behavior is called multistage attack. We can set up the defense scenarios by analyzing and combining those multi-stages. The most abnormal incidents may not be detected at their initial steps, but once their signatures are caught , the attack behaviors would also be known. On the other hand, the first steps of abnormal incidents look like normal events. The great damage has been done when they are been detected. To consider the advantages when those intrusion incidents can be pre-detected at, they did not make any damage yet. We list the following four advantages: (1) to reduce the attacked risk, (2) to reduce detect cost, (3) to increase system security and (4) to reach pre-detection.

Dynamic detecting for an intrusion detection system is also provided in this investigation. In traditional IDSs, the novel attacks and intrusion are impossible to detect and prevent. However, how to estimate the cost and risk to prevent attack or intrusion network systems are critical issues for network systems manager. According to flaw hypothesis methodology and Coloured Stochastic Petri Nets modeling, each attack model is given a corresponded with threshold. Monitoring and tracing behavior of attacks concurrently. The most attacks, however, can be detected or prevented by simulating and analyzing their incident types in depth.

The particle filter, also known as Sequential Monte Carlo method or said Condensation (Isard and Blake 1998), it is the use of the concept that the probability of particle sets to be used in any form of state space. Its core idea is likely to engage in the posterior to express its

distribution. In brief, the particle filter is a means to find a group in the state space of the random sample spread teams to approximate the probability density function to replace the sample mean points operations, thereby gaining the status of the process of distribution of minimum variance. As the number of samples is near infinite, the particle filter scheme can approach any form of probability density function.
The Kalman Filter, however, is based on the assumption target is the linear-type and the Gaussian distribution. The particle filter can be used in non-linear and non-Gaussian distribution model. The Particle filter can obtain a high detection accuracy and target trace rate which the main reason is that it can track the status of a random number of assumptions made at the same time retaining the possibility of a higher number of assumptions, not only left the state of a forecast. Therefore, when the target state of a sudden change in a matter of time before the prediction is wrong, the other particles can state the possibility of a higher state to amend the error.
Kristensen et al (Kristensen, Jorgensen et al. 2004) presented four case studies where CP-nets and their supporting computer tools are used in system development projects with industrial partners. The case studies have been selected such that they illustrate different application areas of CP-nets in various phases of system development. Kristensen and Jensen (Kristensen and Jensen 2004) presented two case studies where CP-nets and their supporting computer tools are used for ad-hoc networks. Dahl (Dahl 2005) and Dahl and Wolthusen (Dahl and Wolthusen 2006) addressed the flaw hypothesis methodology (FHM) to work at the intrusion detection system.
IP trace back is another issue for attack detection and analysis. In our investigation, IP trace back technologies are helpful to analyze and evaluate intrusion detection. Savage et al (Savage, Wetherall et al. 2001) described that trace back is only effective at finding the source of an attack traffic, not necessarily the attacker themselves. Savage et al also defined some basic assumptions and limitations for traffic trace back those are as follows. An attacker may generate any packet, multiple attackers may conspire, attackers may be aware they are being traced, packets may be lost or reordered, attackers send numerous packets, the route between attacker and victim is fairly stable, routers are both CPU and memory limited and routers are not widely compromised.
Snoeren (Snoeren, Partridge et al. 2002) gave another several important assumptions that a trace back system should make about a network and the traffic it carries. The packets may be addressed to more than one physical host, duplicate packets may exist in the network, routers may be subverted, but not often, attackers are aware they are being traced, the routing behavior of the network may be unstable, the packet size should not grow as a result of tracing and hosts may be resource constrained.
Steffan and Schumacher (Steffan and Schumacher 2002) presented the fault tree analysis (FTA) scheme, which fault tree technologies have been used to analyze the failure conditions of complex technical systems for a long time. Attack tree methods can capture the steps of an attack and their interdependencies. Attack tree methods are also used to represent and calculate probabilities, risks, cost, or other weightings. The main building blocks of attack trees are nodes. Each fault tree has a single top node which represents the achievement of the attack's ultimate goal. Interdependencies of goals are modeled by the tree hierarchy. Attack steps that have to be performed successfully before another step can occur are represented by child nodes. To each node either a logical AND or a logical OR gate is associated. An OR-node can occur when any of its child events occurs. For an AND-node

to occur its entire child events are necessary. Fault Tree nodes can be augmented with probabilities or costs, so that the most likely or inexpensive attack path can be calculated. However, those weightings are too specific to be applied to attack trees describing general attack scenarios.

Gordon (Gordon, Salmond et al. 1993) first proposed an algorithm of particle filters, known as a sequential importance resampling (SIR) filter. A key issue in SIR is the selection of the proposal distribution, which determines the approximation performance. Much research ofthe particle filtering focuses on improving the proposal distribution and importance sampling strategies by utilizing the measurements, such as the auxiliary particle filter (Pitt and Shephard 1999). Recently, some kernel based particle filters have been introduced, including Gaussian sum particle filter (Kotecha and Djuric 2003), kernel particle filter (Hurzeler and Kunsch 1998) and Parzen particle filter (Lehn-Schioler, Erdogmus et al. 2004), which enhance the ability of the particles in the posterior distribution representation by the kernel density estimators.

In a traditional particle filter scheme almost applied into trace visual object. In this research, we extend the particle filter function to analyze the network flows and evaluate the risk and cost of intrusion detection system work.

## 2. Background

### 2.1 Intrusion Detection System

Intrusion detection systems (IDS) detect attempted or successful misuses of computer systems. IDS can be classified according to their (1) data sources: network or host audit trails; (2) analysis technique: misuse or anomaly detection; and (3) overall architecture: distributed or autonomousagents.

The Host-based audit trails application and system logs, file attributes, system call and process monitoring, kernel audit facilities. Its problems are as follows. (1) It can't trust audit trail from a compromised host; (2) there is performance impact of active monitoring on target systems.

The Network-based audit trails raw packet data, network flow, and firewall and router logs. Its problems are as follows. (1) The passive network monitoring is easily defeated by clever attackers; (2) the traffic normalizer can help deal with ambiguity; (3) they require the higher bandwidth, end-to-end encryption and switched networks.

The misuse detection looks for specific, identifiable attacks, for example, expert knowledge IDS is rules-based according to attack signatures. Its problems are as follows. (1) It cannot detect novel attacks and (2) it is extremely brittle in the face of mutating attacks or subterfuge.

The Anomaly detection looks for anything that doesn't fit a normal profile. Those methods include following. (1) Equality matching that is a simple anomaly detection - detect deviance from specified normal behavior. Its main problems are an inability to generalize from past observed behavior and subject to state-holding or other denial of service attacks. (2) Statistical profiling that comprise profiles of normal behavior from various statistical measures. Its problems are insensitive to an event ordering and the threshold determination. (3) Machine learning that applies AI techniques (Elman, Petri, neural nets, etc.) to learn normal profiles. Its problems include those are extremely high false positives due to high sensitivity to variance, subject to bad training, and poor real-time performance, questionable real-world applicability.

In popularly, host IDS (HIDS) and network IDS (NIDS) are two kind IDSs. HIDS is to detect the possible intrusion and attack on a host by reviewing the audited data of the host. NIDS is to detect the possible intrusion and attack on a LAN by checking each networking packet on the LAN. The features matching scheme is the main technology for IDS. Although IDS can detect intrusion and attacks, but if the feature data were not been updated in time, then the detection rate would be decreased. Due to the IDS does not find out any new attack or intrusion behavior.

### 2.2 Coloured Petri Nets

Coloured Stochastic Petri Nets are now in widespread use for many different practical purposes (Jensen 1992). The main reason for the great success of these kinds of net models is the fact that they have a graphical representation and a well-defined semantics allowing formal analysis. Real-world systems often contain many parts, which are similar, but not identical. Using CSPN, these parts must be represented by disjoint sub nets with a nearly identical structure. The practical usages of CSPN to describe real-world systems have clearly demonstrated a need for more powerful net types, to describe complex systems in a manageable way.

The formal definition of a Petri Net graph is as follows (Dahl 2005): A *Petri net graph* G is a bipartite directed multigraph, G = (V, A), where V = $v_1$, $v_2$, $v_3$, …, $v_n$ is a set of vertices and A = $a_1$, $a_2$, $a_3$, …, $a_n$ is a multiset of directed arcs, $a_i$ = ($v_j$, $v_k$), with $v_j$, $v_k \in$ V. The set V can be partitioned into two disjoint sets P and T such that V = P$\cup$T, P$\cap$T = $\Phi$, and for each directed arc, $a_i \in$A, if $a_i$ = ($v_j$, $v_k$), then either $v_j \in$ P and $v_k \in$ T or $v_j \in$ T and $v_k \in$P.

Furthermore, the formal definition of a Coloured Petri Net is as follows: A non-hierarchical coloured Petri net is a tuple CPN = ($\Sigma$, P, T, A, N, C, G, E, I) satisfying the requirements below: (1) $\Sigma$ is a finite set of non-empty types, called *colour sets*. (2) P is a finite set of *places*. (3) T is a finite set of *transitions*. (4) A is a finite set of arcs such that: $P \bigcap T = P \bigcap A = T \bigcap A = \varnothing$. (5) N is a *node* function. It is defined from A into $P \times T \bigcup T \times P$. (6) C is a *colour* function. It is defined from P into $\Sigma$. (7) G is a *guard* function. It is defined from T into expressions such that: $\forall t \in T : [Type(G(t)) = B \wedge Type \quad (Var(G(t))) \subseteq \Sigma]$. (8) E is an *arc expression* function. It is defined from A into expressions such that: $\forall a \in A : [Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$ where p(a) is the place of N(a). (9) The I is an *initialization* function. It is defined from P into closed expressions such that: $\forall p \in P : [Type(I(p)) = C(p)_{MS}]$.

The formal definition of timed Coloured Petri Nets (Jensen 1997), i.e., the formal definition of Stochastic Coloured Petri Net, is as follows: A *timed* non-hierarchical Coloured Petri Net is a tuple TCPN = (CPN, R, $r_0$) such that (1) Coloured Petri Net satisfied the requirements of a non-hierarchical Coloured Petri Net as defined in the abovesection when in arc expression function and the initialization function. We allow the type of E(a) and I(p) to be a timed or an un-timed multiple set over C(p(a)) and C(p), respectively. (2) R is a set of *time values*, also called *time stamps*. It is a subset of $\Re$ closed under + and containing 0. (3) $r_0$ is an initial element of R, called the *start time*.

The interval definition is as follows: (1) TS is the time set, $TS = \{x \in \Re \mid x \geq 0\}$, i.e. the set of all non-negative real numbers. (2) $INT = \{[y,z] \in TS \times TS \mid \leq z\}$, represent the set of all closed intervals. If $x \in TS$ and $[y,z] \in INT$ then $x \in [y,z]$ if and only if $y \leq x \leq z$.

The basic elements of a CSPN graph are listed as follows (Haas 2002): (1) A finite set $D = \{ d_1, d_2, \ldots, d_L \}$ of *places*. (2) A finite set $E = \{ e_1, e_2, \ldots, e_M \}$ of *transitions*. (3) A (possibly empty) set $E' \subset E$ of *immediate transitions*. (4) A finite set $U$ of C*olours* with a fixed enumeration. (5) Colour domains UD(d) $\subseteq$ U for d $\in$ D and UE(e) $\subseteq$ U for e $\in$ E. (6) An input incidence function w− and an output incidence function w+, each defined on $U_{e \in E, d \in D}$ ({e}× UE(e)×{d} × UD(d)) and taking values in the nonnegative integers.

### 2.3 Particle Filter

The particle filter is an inference technique that estimates the unknown state from the sampling particle collection of observation $Y_{1:t}=\{Y_1, \ldots, Y_t\}$. It approximates the posterior distribution $p(S_t \mid Y_{1:t})$ by a set of weighted particles $Z_t = \{Y_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ with $\sum_{i=1}^{N} w_t^{(i)} = 1$.

The dynamic state system consists of the state transition model and the observation model. The s*tate transition model:* $S_t = F_t(S_{t-1}, V_t)$, and the o*bservation model:* $Y_t = H_t(S_t, W_t)$. The state transition function $F_t$ approximates the dynamics of the object being tracked using the previous state $S_{t-1}$ and the system noise $V_t$, and the measurement function $H_t$ models a relationship between the observation $Y_t$ and the state $S_t$ given the observation noise $W_t$. We usually characterize the state transition model with the state transition probability $p(S_t \mid S_{t-1})$ and the observation model with the likelihood $p(Y_t \mid S_t)$.

A general procedure of the particle filter consists of three steps: re-sampling, prediction, and update step. In the re-sampling step, we resample the particles $Z_{t-1}$ to obtain the non-weighted set of particles with equal weights $\{S_{t-1}^{'(i)}, 1\}_{i=1}^N$. In the prediction step, we draw the particles $\{V_t^{(i)}\}_{i=1}^N$ and generate the particles $\{S_{t-1}^{'(i)}, 1\}_{i=1}^N$ using the state transition model $S_t = F_t(S_{t-1}, V_t)$. In the updating step, we update the weight of each particle based on the observation likelihood as $w_t^{(i)} \propto p(Y_t \mid S_t^{(i)})$.

Particle filter is entitled by a group of weighting particles to calculate posterior probability. The Equation (1) is a formula of the Bayes theorem of posterior probability.

$$
\begin{aligned}
P(S_t \mid Z_{1:t}) &= \frac{P(Z_{1:t} \mid S_t)P(S_t)}{P(Z_{1:t})} = \frac{P(Z_t, Z_{1:t-1} \mid S_t)P(S_t)}{P(Z_t, Z_{1:t-1})} \\
&= \frac{P(Z_t \mid Z_{1:t-1}, S_t)P(Z_{1:t-1} \mid S_t)P(S_t)}{P(Z_t \mid Z_{1:t-1})P(Z_{1:t-1})} \\
&= \frac{P(Z_t \mid Z_{1:t-1}, S_t)P(S_t \mid Z_{1:t-1})P(Z_{1:t-1})P(S_t)}{P(Z_t \mid Z_{1:t-1})P(Z_{1:t-1})P(S_t)} \\
&= \frac{P(Z_t \mid S_t)P(S_t \mid Z_{1:t-1})}{P(Z_t \mid Z_{1:t-1})}
\end{aligned} \tag{1}
$$

In the Equation (1), we can find that the posterior probability $P(S_t \mid Z_{1:t})$ could be presented by the priori probability $P(S_t \mid Z_{1:t-1})$ and the observation model $P(Z_t \mid S_t)$. Therefore, we can obtain the Equation (2) and (3).

$$
\begin{aligned}
P(S_t \mid Z_{1:t-1}) &= \int P(S_t, S_{t-1} \mid Z_{1:t-1})dS_{t-1} \\
&= \int P(S_t \mid S_{t-1}, Z_{1:t-1})P(S_{t-1} \mid Z_{1:t-1})dS_{t-1} \\
&= \int P(S_t \mid S_{t-1})P(S_{t-1} \mid Z_{1:t-1})dS_{t-1}
\end{aligned} \tag{2}
$$

$$
\begin{aligned}
P(Z_t \mid Z_{1:t-1}) &= \int P(Z_t, S_{t-1} \mid Z_{1:t-1})dS_{t-1} \\
&= \int P(Z_t \mid S_t, Z_{1:t-1})P(S_t \mid Z_{1:t-1})dS_{t-1} \\
&= \int P(Z_t \mid S_t)P(S_t \mid Z_{1:t-1})dS_{t-1}
\end{aligned} \tag{3}
$$

Therefore, the Equation (2) and (3) substitute for (1)，we can obtain the Equation (4).

$$
P(S_t \mid Z_{1:t}) = kP(Z_t \mid S_t)\int P(S_t \mid S_{t-1})P(S_{t-1} \mid Z_{1:t-1})dS_{t-1}\text{, where } k \text{ is a normalized constant.} \tag{4}
$$

According to the Equation (4), we can evaluate and forecast the real state of the object. $P(Z_t \mid S_t)$ is the observation model of the probability density function that is a likelihood function. $P(S_t \mid S_{t-1})$ is the state transform model and $P(S_{t-1} \mid Z_{1:t-1})$ is the posterior probability at time t-1. So that, we can substitute and update each state of posterior probability of object along with the initial state distribution $P(S_0)$. As $N \to \infty$, the Equation (2) can be presented by the Equation (5).

$$
\begin{aligned}
P(S_t \mid Z_{1:t-1}) &= \int P(S_t \mid S_{t-1})P(S_{t-1} \mid Z_{1:t-1})dS_{t-1} \\
&\approx \sum_{i=1}^{N} P(S_t \mid s_{t-1}^{(i)})w_{t-1}^{(i)}
\end{aligned} \tag{5}
$$

Therefore, the Equation (2-8) could be substituted by the Equation (6).

$$\begin{aligned} P(S_t \mid Z_{1:t}) &= kP(Z_t \mid S_t)\int P(S_t \mid S_{t-1})P(S_{t-1} \mid Z_{1:t-1})dS_{t-1} \\ &\approx kP(Z_t \mid S_t)\sum_{i=1}^{N} P(S_t \mid s_{t-1}^{(i)})w_{t-1}^{(i)} \end{aligned} \tag{6}$$

The principal steps in the particle filter algorithm:
// Input: the object that would be analyzed, detected and traced.
// Output: a set of particles $\{S_t^{(i)}, w_t^{(i)}\}_{i=1}^{N}$ that can be used to approximate the posterior distribution.
Step 1: Initializing particles.
Set t = 1;
Generate particle set from the initial distribution $p(S_0)$ to obtain $\{S_0^{(i)}, w_0^{(i)}\}_{i=1}^{N}$, the initial state N particles $\left\{s_0^{(i)}\right\}_{i=1}^{N}$ and setting their weights $\left\{w_0^{(i)}\right\}_{i=1}^{N}$, where each $w_0^{(i)} = 1/N$.
Step 2: The forecasting the next state of particles.
In the set of particles, $\left\{s_t^{(i)}\right\}_{i=1}^{N}$ presents the state of each particle at time t, according to the transition model $p(S_k^{(i)} \mid S_{k-1}^{(i)})$.
Step 3: Observing and evaluating particles.
Evaluate the new state of detected particles by the importance likelihood:
$w_t^{(i)} = \dfrac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}}$ $i = 1,...,N$. Let the new weights at time t be $w_t^{(i)} = P(Z_t \mid S_t^{(i)})$ $i = 1...N$.
Step 4: Output
Output a set of particles $\{S_k^{(i)}, w_k^{(i)}\}_{i=1}^{N}$ that can be used to approximate the posterior distribution as $p(S_t \mid Z_t) = \sum_{i=1}^{N} w_t^{(i)} \delta(S_t - S_t^{(i)})$ where δ () is the Dirac delta function.
Step 5: Resample
Resample particle set $\left\{s_t^{(i)}\right\}_{i=1}^{N}$ with probability $w_t^{(i)}$ to obtain *N* independent and identically distributed random particle set $\left\{s_t^{(j)}\right\}_{j=1}^{N}$ approximately distributed according to $p(S_t \mid Z_t)$. The resample sub-algorithm is as follows.

Let $New_w_t^{(1)} = w_t^{(1)}$;
For i = 2 : N
$New_w_t^{(i)} = New_w_t^{(i-1)} + w_t^{(i)}$
End for

For i = 1 : N
  r = random(0,1);
  for j = 1 : N
    if ( $New_w_t^{(j)} \geq r$ ) then
      k = j;
      break for;
    end if
  end for
  if (i <> k) then
    $New_s_t^{(i)} = s_t^{(k)}$
  End if
End for

Step 6 Set t = t + 1, and return to Step 2.

## 3. The NetworkParticle Filtering Model in Intrusion Detection

A time window is during three seconds. Moving a time window per one second or two seconds, i.e., there are two one or two seconds overlap between two time windows. We Used thenetwork particle filter scheme to classify network packets into two classes those include normal or abnormal network behaviors in each time window. To classify packets within the continue time window is to classify packets in each time window during a longer time. The system builds the relationship within these classes. The basic information in each network packet includes source IP, destination IP, source TCP port number and destination TCP port number.

### 3.1 The Definition of the Network Intrusion

Firstly, we give some definitions to describe the meanings and behaviors of network flow in IDSs.

***Definition 1 Time Window:*** A time window is a time interval that covers many network flow packets.

***Definition 2 Malicious Event:*** An event generated by a single attempt to violate certain security policies, regardless of whether the attempt achieves its goal.

According to definition 2, even if an attempt fails to violate a security policy, the events it generates are still malicious. This conforms to the common understanding of a malicious event. For example, an attempt to overflow a buffer on no vulnerable web server is still malicious, even though it fails.

***Definition 3 Suspicious Event:*** No malicious event generated by an attempt that has a strong logical connections with the malicious events.

For example, some Snort signatures detect IP sweep attempts that do not violate the security policies of many sites. However, these events often have a strong connection to intrusion attempts because the attackers are trying to identify active computer systems.

***Definition 4 Attack:*** A malicious or suspicious event detected by the IDSs.

We shall concentrate on the events that IDSs detect, because usually attacks are only discernable in terms of IDS alerts. Moreover, alert correlation only works on the alerts, and not

on the events that the IDSs do not detect. In addition, this definition of an attack makes it interchangeable with the IDS alert in the following. Thus, we will not always explicitly state that an attack is represented by the alerts.

***Definition 5 Alert:*** A message reported by the IDSs as the result of an attack.

***Definition 6 Intrusion Incident:*** A sequence of related attacks within a time frame against a single computer system by an attacker to achieve some goals.

The definition 5 and 6 describe the output of the IDS. The Alert can talk to a system or system manager to make a response for this Alert automatically or artificially.

***Definition 7 Alert Fusion (Aggregation):*** Grouping alerts by their common characteristics; typically, grouping alerts of the same signature and network addresses.

***Definition 8 Requires/Provides (Prerequisite) Relation:*** If an early attack provides logical support, e.g., information of or access to the system under attack, for a later attack that requires it, there is a requires/provides relation between the two attacks and the corresponding alerts of the attacks.

***Definition 9 Alert Correlation:*** Grouping alerts by their required or provided relation.

The definitions 7, 8 and 9 are to analyze and build the relationship between Alerts, and then the IDS can supply more useful report or response policies.

### 3.2 Network Particle Filtering Model

The Fig. 1 shows the proposed network particle filtering model that is implemented in the CPN tools. In this model includes senders who send network packetsto some hosts. The NPF recognizes and classifies each network packet into normal or abnormal classes by network particle filter scheme. Those hosts are the destination computer of the sent network packet from a sender.

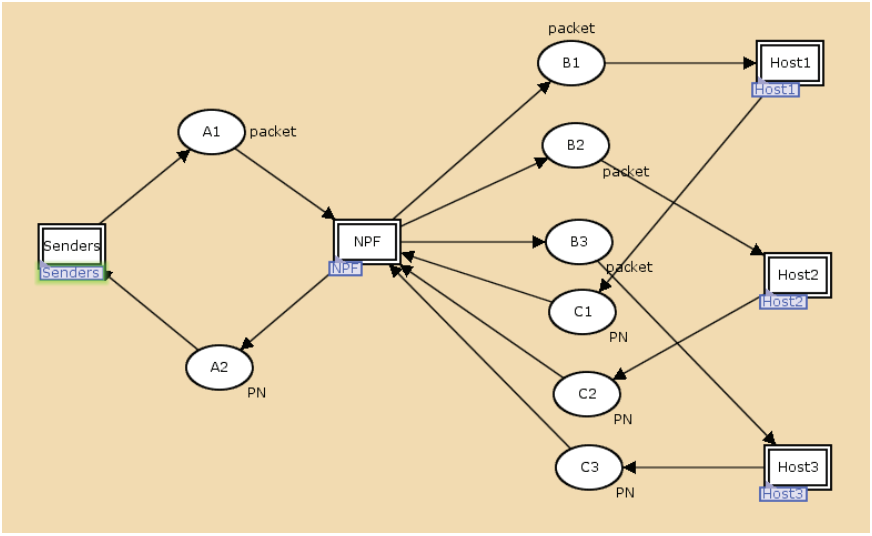



Fig. 1. The hierarchical network particle filtering model.

In this research, we considered four cases to analyze network packets and two kinds of attacksto detect intrusion behaviors by network particle filtering model. The first case is one

packet analysis, to consider each packet, and select N features as particles. Each particle is given a different weight. Through multi-step filtering, each packet could be defined as normal or abnormal behavior. If one packet is found to be an abnormal behavior, then tracking corresponding packages with the same as source IP address, TCP port number, and UDP port number and increasing the corresponding weight of particles. The second case is multiple packet analysis, the timed packets flow. To find and analyze the relationship between multiple packets, how to decide each next related packet is normal or abnormal behavior. To increase normal particle weight and decrease abnormal particle weight when the last related packet is belonged to normal behavior. On the other hand, to decrease normal particle weight and increase abnormal particle weight when the last related packet is belonged to abnormal behavior. Then evaluate the score for each related packet particle. If the normal score is greater than the abnormal score, then this packet is belonging to the normal behavior. In the similar, if the abnormal score is greater than the normal score, then this packet is belonging to the abnormal behavior.

The next case is in a time window, the number of the source and destination IP address (NSDIP) and the number of source and destination TCP port (NSDTCP) for each packet would be summarized and given a weight and probability for each NSDIP and NSDTCP in a time window. The value of weights is between 0 and 1, and their sum is equal to 1. A threshold of the weight would be given to evaluate whether some packets are abnormal behaviors or not.

The final case is within multiple time windows and overlapping time windows. The next step, we selected those abnormal packets from multiple time windows. Those abnormal packets would be analyzed and found the relationships betweenthem. Therefore, those abnormal packets would be classified and named one attack. And then IDS creates the attack pattern and update into the pattern database. The IDSs could make a response to each detected attack. For example, IDSs could send alerts to system manager, log each detected incident and attack, and auto-response by system defined.

We assume that there are some relationships for some packets between two neighbor time windows. If the relationship exists, then we can work at the packet trace. Otherwise, it will be failed to trace. Therefore, we should extend the filtering field to more time windows that maybe cover some packets with relationship; or begin another packet trace because the last trace packet is the end of sequence.

The detection models can be divided into offline and online cases. Fig. 2 (a) shows the flowchart of offline detection case. In the offline case, the input is the collected data during a time interval that included more than one time window. The next step is to analyze and classify packets during one time window using the network particle filter scheme. And then the step is to detect intruded behaviors and update the intrusion pattern database when the new intrusion behaviors were been found.

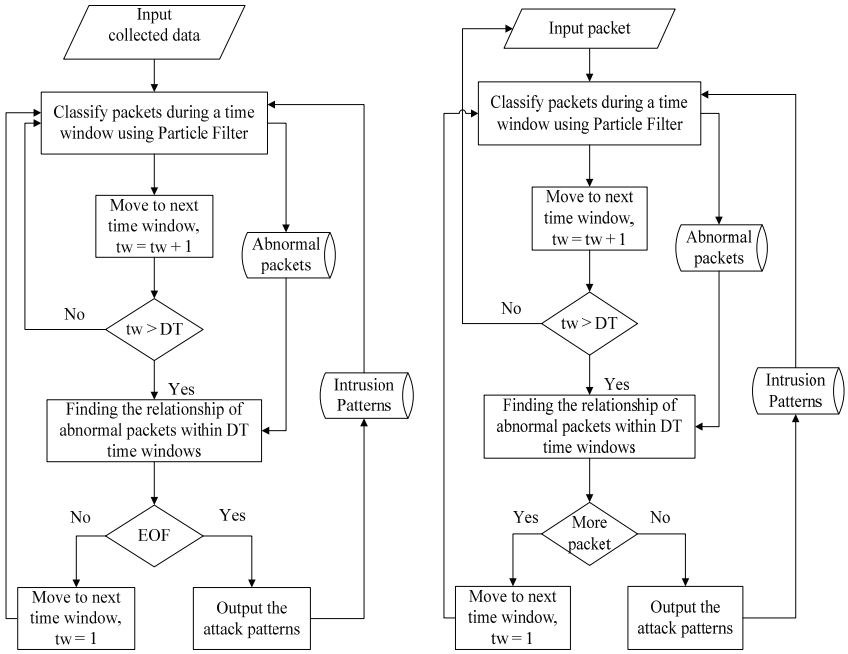



Fig. 2. (a) The offline analysis flowchart of the network particle filtering IDS. (b) The online detection flowchart of the network particle filtering IDS.

Fig. 2 (b) shows the flowchart of offline detection case. The most difference between offline and online cases is the input data. In the online case, the input data are the real time received network packets from senders. Therefore, we analyze and classify packets using the network particle filter scheme when the time is up for a time window. The abnormal packet database is to keep the dubious packets those could be used after some time windows. The intrusion pattern database is to save the patterns that had been confirmed as intrusionbehaviors. The intrusion pattern database could be updated when the system found a new intrusion pattern.

## 4. Experimental Results

In our experiment include two simulation cases Intrusion detection and Trojan detection. In the Intrusion detection case, we assume that the almost intrusion behaviors come from senders. Therefore, we just design the network particle filter scheme to detect the packets those have been sent from senders. On the other hand, the Trojan detection case, we assume almost dubious packets come from receiver's acknowledge. Therefore, we set the network particle filter scheme on the outward path. The network particle filter scheme is designed to analyze and classify each network packet into normal or abnormal class for inward and outward, respectively.

The simulation platform is CPN Tools for Coloured Petri Nets that supports good interface and tools to implement the Coloured Petri Nets model. We design a hierarchical network that includes four main parts system view, sender, NPF, and hosts. We also let each kind attack simulation be executed 4000 steps to claim the trend of the results.

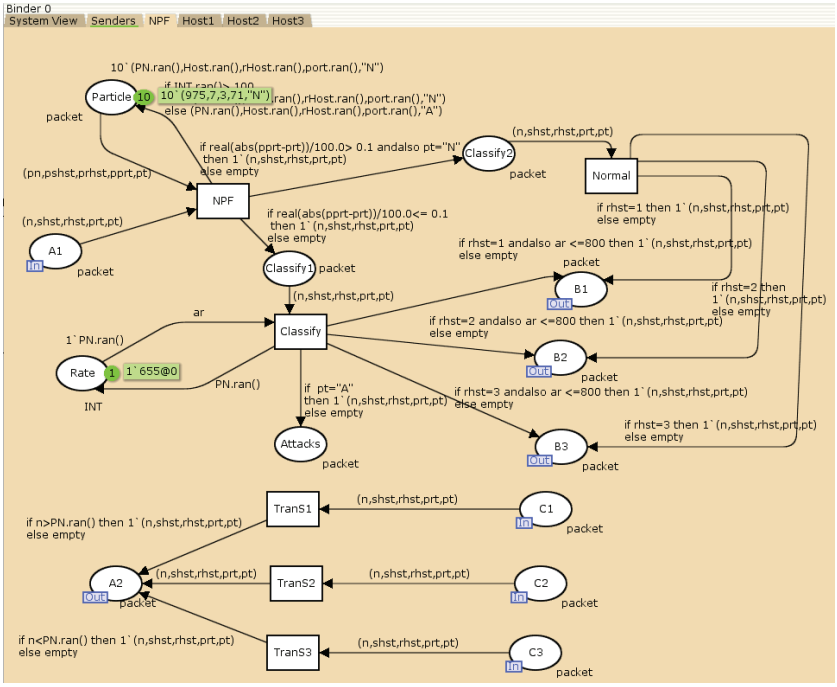
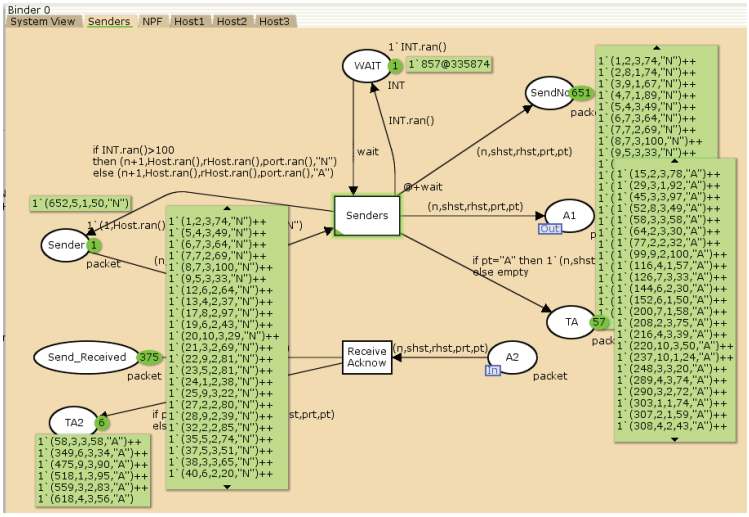



Fig. 3. The CPN simulation for NPF-Intrusion. The initial status.

Fig. 3 shows the initial status of NPF part in the Intrusion detection case using CPN Tools. The 'Particle' place is to be designed to create 10 particles randomly. The 'NPF' transition is to set up the filtering conditions and classify packets into normal class or dubious class. The 'Classify' transition refines the classification of 'NPF'. And then sends the attack packets to the 'Attacks' place. The 'Attacks' place is to record the attack packets those have been captured by 'NPF' transition.



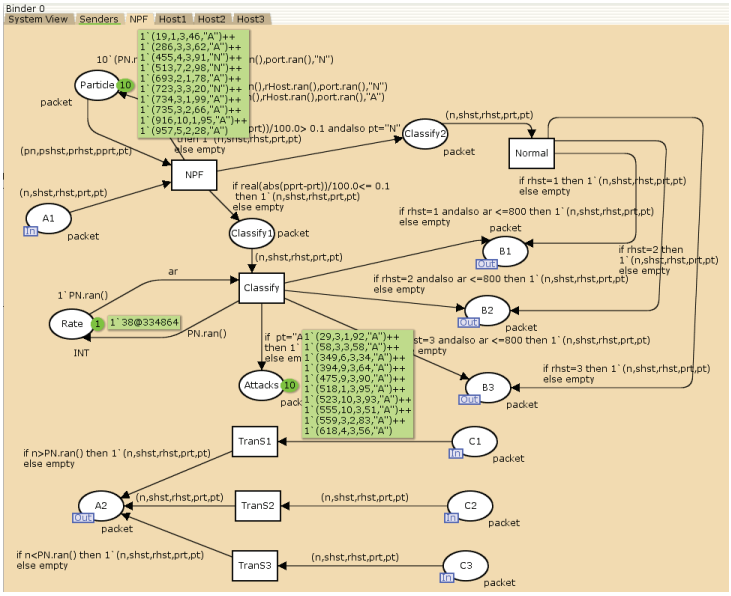




Fig. 5 shows the status of NPF part after 4000 steps. There are 10 attack packets in 'Attacks' place, i.e., the system captured 10 attack or intrusion behaviors. At the same time, there are 57 attack packets in 'TA' place. So that, we can obtain the total detection rate is 17.54%.
Fig. 6 shows the initial status of NPF part for the Trojan detection case. The 'Collect' place is to collect all acknowledged packets from the receiver , and then sends them to the 'NPF' transition. The 'NPF' transition is to detect each passed packet is the Trojan behavior or not.

The ‘Trojan’ place saved the possible Trojan packets those have been captured from the ‘NPF’ transition.
Fig. 7 shows the status of the Sender part after 4000 steps on the Trojan detection case. The total sent packets are 693. At the same time, there are 470 acknowledged packets have been received. Fig. 8 shows the status of NPF part after 4000 steps on the Trojan detection case. The total number of Trojan packets is 66 in the ‘Trojan’ place. So that, we can obtain the total Trojan behavior rate is 14.04%.

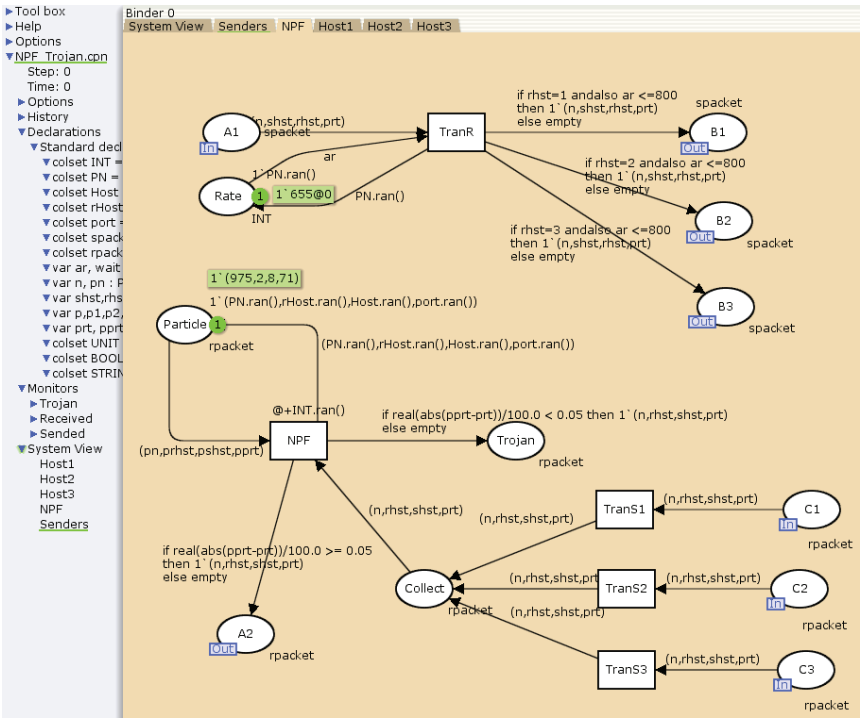


Fig. 6. The CPN simulation for NPF-Trojan. The initial status.

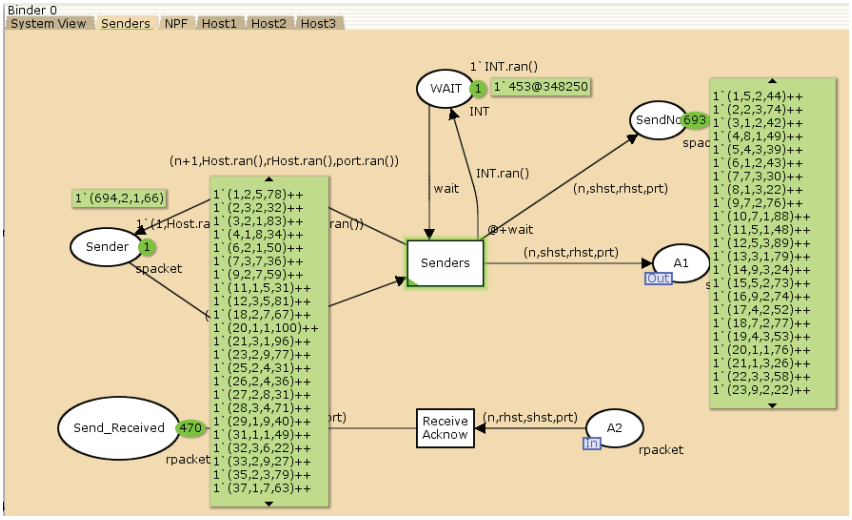



Fig. 7. The CPN simulation for NPF-Trojan. The status of Sender after 4000 steps.

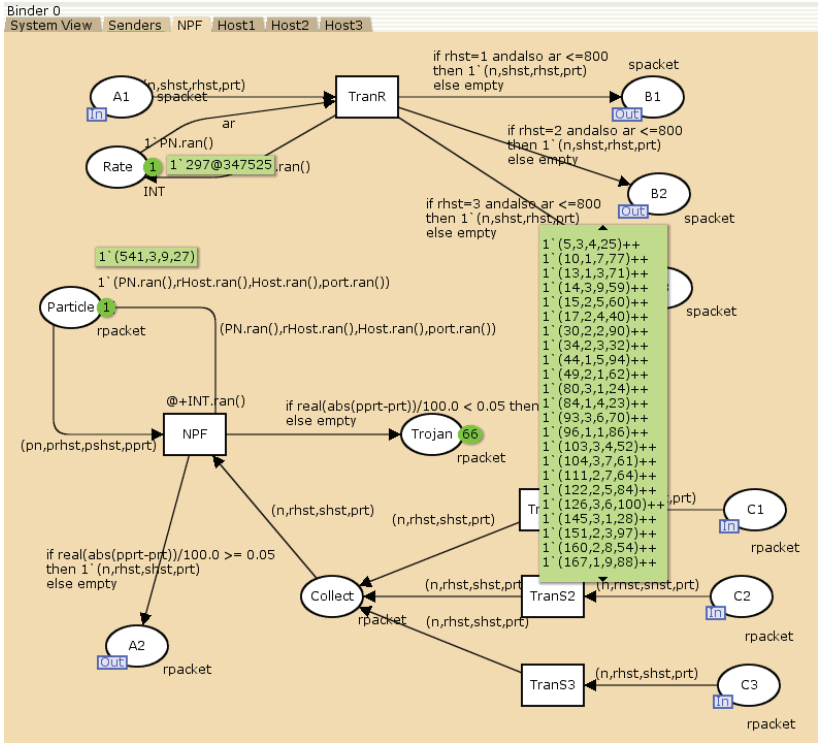



Fig. 8. The CPN simulation for NPF-Trojan. The status of the NPF after 4000 steps.

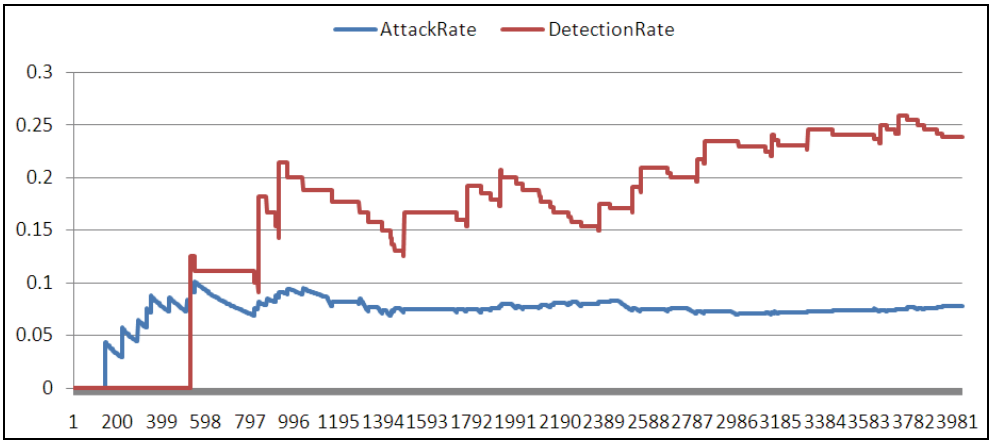



Fig. 9. The attack rate and detection rate of the CPN simulation for NPF-Intrusion after 4000 steps.

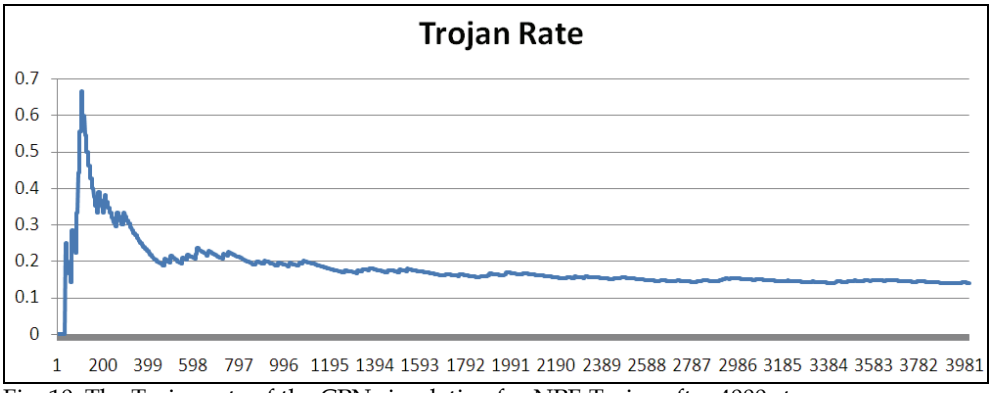



Fig. 10. The Trojan rate of the CPN simulation for NPF-Trojan after 4000 steps.

In our simulation process, we record the number of some places, for example, "sendNo", "Attacks" and „Send_Received" at eachstep. Fig. 9 shows the attack packet rate and attack detection rate. The attack packet rate is the ratio of attack packets against all sent packets. The attack detection rate is the ratio of attack packets against all signed 'A' packets. Fig. 10 shows the Trojan rate that is the ratio of Trojan packet against all acknowledged packets from the receivers. In the first 200 steps illustrates a suddenly rise high interval, the reason is the most of the acknowledged packets passed through the NPF node but have not yet arrived at the sender node. The trend, however, of the results matches our design.

## 5. Conclusions

In this paper, we have provided a network particle filtering with a stochastic model for an intrusion detection system, and simulated this scheme in the ColouredPetri Nets Tools plat-

form. To build a real test environment and collect real attack case data is not easy. We proposed a test bed platform that can support to test and simulate the network attack cases.
The particle filter scheme applied into network analysis is a difficult work, because the related behaviors of network flow are not continuous and it is very difficult to know and control them. Therefore, the accuracy of our simulation results is not enough good. The design of network flow also does not touch the real attack cases. Our approaches, however, can be applied to the risk and cost evaluation to practice an IDS.

## 6. References


Dahl, O. M. (2005). Using Coloured Petri Nets in Penetration Testing. Department of Computer Science and Media Technology. Gjøvik, Gjøvik University College. Master: 1-89.

Dahl, O. M. and S. D. Wolthusen (2006). Modeling and execution of complex attack scenarios using interval timed colored Petri nets. the Fourth IEEE International Workshop on Information Assurance.

Gordon, N. J., D. J. Salmond, et al. (1993). "Novel approach to nonlinear/non-Gaussian Bayesian state estimation." Inst. Elect. Eng. F, Radar Signal Process 140: 107-113.

Haas, P. J. (2002). Stochastic Petri Nets Modelling, Stability, Simulation Springer-Verlag New York, Inc.

Hurzeler, M. and H. R. Kunsch (1998). "Monte Carlo approximations for general state space models." J. Computat. Graph. Statist 7(2): 175-193.

Isard, M. and A. Blake (1998). "CONDENSATION—Conditional Density Propagation for Visual Tracking." Int. J. Comput. Vision 29: 5-28.

Jensen, K. (1992). Coloured Petri Nets Basic Concepts, Analysis Methods and Practical Use Springer-Verlag.

Jensen, K. (1997). Coloured Petri Nets Basic Concepts, Analysis Methods and Practical Use Springer-Verlag.

Kotecha, J. H. and P. M. Djuric (2003). "Gaussian sum particle filtering." IEEE Trans. Signal Process 51(10): 2602-2612.

Kristensen, L. M. and K. Jensen, Eds. (2004). Specification and Validation of an Edge Router Discovery Protocol for Mobile Ad-hoc Networks. Lecture Notes in Computer Science, Springer-Verlag.

Kristensen, L. M., J. B. Jorgensen, et al., Eds. (2004). Application of Coloured Petri Nets in System Development. Lecture Notes in Computer Science, Springer-Verlag.

Lehn-Schioler, T., D. Erdogmus, et al. (2004). "Parzen particle filters." IEEE Int. Conf. Acoust., Speech, Signal Process 5: 781-784.

Pitt, M. and N. Shephard (1999). "Filtering via simulation: Auxiliary particle filters." J. Amer. Statist. Assoc. 94(446): 590-599.

Savage, S., D. Wetherall, et al. (2001). "Network Support for IP Traceback." IEEE/ACM TRANSACTIONS ON NETWORKING 9(3): 226-237.

Snoeren, A. C., C. Partridge, et al. (2002). "Single-Packet IP Traceback." IEEE/ACM TRANSACTIONS ON NETWORKING 10(6): 721-734.

Steffan, J. and M. Schumacher (2002). Collaborative attack modeling. Proceedings of the 2002 ACM symposium on Applied computing, Madrid, Spain, ACM Press.

4

# Modeling and Analyzing Software Architecture Using Object-Oriented Petri Nets and $\pi$-calculus

Zhenhua Yu[1,2], Xiao Fu[1], Yu Liu[2], Jing Wang[2] and Yuanli Cai[3]
[1]*School of Telecommunication Engineering, Air Force Engineering University*
[2] *Xi'an Applied Optics Institute*
[3]*School of Electronic and Information Engineering, Xi'an Jiaotong University*
*People's Republic of China*

## 1. Introduction

Software architecture has recently emerged as a new discipline of software engineering to effectively develop and maintain complex and large-scale software systems and reduce costs of developing applications. Software architecture provides a high-level abstraction for representing components, their relationships to each other and environment, and their constraints. The overall system structure design and specifications are far more important than the selection of the specific algorithms and data structures. Therefore, software architecture is a critical factor to success for system design and development (Shaw & Clements, 2006).

Software architecture can be characterized according to its evolution at runtime (Oquendo, 2004): 1. static architectures: the architecture does not evolve during the execution of the system; 2. dynamic architectures: the architecture can evolve during the execution, e.g. components can be created, deleted, reconfigured, or updated at run-time. Dynamic software architectures have several practical applications (Medvidovic & Taylor, 2000). In public information systems with high availability and in mission- and safety-critical systems, the implementation of architectural evolvement at run-time can decrease the cost and risk. To support architecture-based development, architecture description languages (ADLs) and formal models have been proposed to represent software architecture in a formal way, such as UniCon (Shaw et al., 1995), Darwin (Magee, 1995), Rapide (Luckham et al., 1995), Wright (Allen et al., 1998), $\pi$-ADL (Oquendo, 2004), SAM (He et al., 2004), XYZ/ADL (Luo et al., 2000). However, the major attentions have been focused on the description of static architectures, while the description of dynamic architecture has not yet to receive the attention it deserves. Darwin and Rapide only depict predefined dynamic evolvement and cannot verify the integrality and liveness of the systems. Wright can describe the dynamic evolvement, but it is so complicated. For a two-tier client/server system, process algebra (such as $\pi$-calculus) uses two processes and one or two channels to depict it, while Wright only needs seven processes and eight channels. $\pi$-SPACE and $\pi$-ADL cannot analyze the key characteristics. Although the existing approaches provide support for dynamic software architecture, most of them cannot analyze and verify the key characteristics. Therefore, software systems cannot be ensured robustness, consistency and maintenance.

To support the development of correct and robust dynamic software architectures, a visual software architecture formal model (SAFM) based on two complementary formalisms, namely Object-oriented Petri nets (OPN) and $\pi$-calculus, is proposed. SAFM divides software systems

into components, connectors and configuration module. In SAFM, OPN are employed to visualize the static architecture and depict the behavior of software systems, and $\pi$-calculus is used to describe software architecture evolution, including component joining, exiting, updating, load balancing and architecture reconfiguration. As $\pi$-calculus, which is based on the interleaving semantics, cannot depict the true concurrency and has few supporting tools, the $\pi$-calculus model of architecture evolution is translated into Petri nets (Yu et al., 2007). Consequently the structural analysis techniques allow the qualitative analysis of properties that may be proved directly on the structure of Petri nets, and the final model can be directly analyzed and verified using existing Petri net tools. SAFM approach supports detection of design errors in an early software design stage and the quality of the software can be significantly improved.

## 2. Object-oriented Petri Nets, $\pi$-calculus and Their Integration

### 2.1 A New Object-oriented Petri Nets (OPN)

The ordinary Petri nets models are very complicated, which highly depend on the system and lack the modularity and flexibility. Consequently state explosion in ordinary Petri net modeling is easily occurred. To solve the complexity and state explosion, Petri nets are combined with Object-oriented methods to set up Object-oriented Petri nets. Object-oriented Petri nets can tersely and independently represent all kinds of resources in a complex system, increase the flexibility of the model. Many kinds of Object-oriented Petri nets (Miyamoto & Kumagai, 2005) are presented. However many of them cannot completely describe the characteristics of objects. From software components perspective, a new Object-oriented Petri nets (OPN) are presented. In OPN both the modularity and flexibility are better than those of ordinary Petri nets, and the state explosion problem is a little more alleviated.

The OPN model of a physical object is defined as follows.

*Definition* 1 OPN is a 9-tuple, $OPN = (\Sigma, P, T, IT, OT, F, E, G, C)$, where $\Sigma$ is color sets, which is a finite set of data types, variables and functions; where $P$ is a finite set of places, $P = \{p_1, p_2, \ldots, p_j\}$; $T$ is a finite set of transitions, $T = \{t_1, t_2, \ldots, t_k\}$; $IT$ (Input Transition) and $OT$ (Output Transition) are sets of input and output transitions, $IT = \{it_1, it_2, \ldots, it_l\}$, $OT = \{ot_1, ot_2, \ldots, ot_m\}$; $F \subseteq (P \times T) \bigcup (T \times P) \bigcup (P \times IT) \bigcup (IT \times P) \bigcup (P \times OT) \bigcup (OT \times P)$ is the input and output relationships between transitions and places; $E : F \rightarrow (ID, CDS)$ is expression functions in the arcs, $ID$ is the identification of the arc and $CDS \subseteq \Sigma$ is a complicated data structure; $G$ is the guard function of the transitions, which is a boolean expression. $C(P) \subseteq \Sigma$ is a set of color associated with the places $P$.

A system is composed of objects and their interconnection relations, and its formal definition is given as follows.

*Definition* 2 A system is a 3-tuple, $S = (OPN, Gate, C)$, where $OPN$ is a finite set of physical objects in the system, $O = \{OPN_1, OPN_2, \ldots, OPN_i\}$; $Gate$ is a finite set of communication places, which are message passing relations among $OPN$; $C(Gate) \subseteq \Sigma$ is a set of color associated with the places $Gate$.

OPN can represent the object-oriented characteristics, such as encapsulation, inheritance and polymorphism. The behavior equivalence of models can be judged by the branch bisimilarity (Yu, 2006)

### 2.2 $\pi$-Calculus

The $\pi$-calculus (Milner et al., 1992) is an extension of the process algebra CCS (Calculus of Communicating Systems) in order to allow dynamic reconfiguration of systems. The model-

ing entities are *names* and *processes*. Systems are represented as a set of *processes* which interact by means of *names*. The *names* can be regarded as shared channels, variables or constants, which act as subjects for interaction. The *process* can use the *name* as a subject for future transmissions, which allows an easy and effective reconfiguration of the system.
We assume an infinite set of *names* $\mathcal{N}$, ranged over by $a, b, \ldots, z$, which will function as all of channels, variables and data values; a set of *process identifiers* $\mathcal{K}$ is ranged over by $A, B, \ldots$, each with an arity (an integer $\geq 0$); the *processes* are ranged over by $P, Q, R, \ldots$, which are of seven kinds as follows:

1. A *Sum* $\sum_{i \in I} P_i$ representing the process that can enact one or other of the $P_i$.
2. A *prefix* form $\overline{y}x.P$, $y(x).P$, or $\tau.P$.
   $\overline{y}x.$ is called *negative* prefix. $\overline{y}$ may be thought of as an *output port* of a process; $\overline{y}x.P$ outputs the name $x$ at port $\overline{y}$ and then behaves like $P$.
   $y(x).$ is called *positive* prefix. $y$ may be thought of as an *input port* of a process; $y(x).P$ inputs an arbitrary name $z$ at port $y$ and then behaves like $P\{z/x\}$.
   $\tau.$ is called *silent prefix*, which represents an agent that can evolve to $P$ without interaction with environment. $\tau.P$ performs the *silent action* $\tau$ and then behaves like $P$.
3. A *Parallel Composition* $P|Q$, which represents the combined behaviors of $P$ and $Q$ executing in parallel. The processes $P$ and $Q$ can act independently, and may also communicate if one performs an output and the other an input along the same port.
4. A *restriction* $(\nu x)P$. This process behaves as $P$ but the name $x$ is local, meaning it cannot immediately be used as a port for communication between $P$ and its environment.
5. A *match* $[x = y]P$. This process behaves like $P$ if the names $x$ and $y$ are identical, otherwise it does nothing.
6. A *defined agent* $A(y_1, \ldots, y_n)$. For any process identifier $A$ (with arity $n$) used thus, there must be a unique *defining equation* $A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P$, where the names $x_1, \ldots, x_n$ are distinct and are the only names which may occur free in $P$.
7. A *Replication* $!P$. $!P$ is given by the definition $!P \stackrel{\text{def}}{=} P|!P$, which represents an unbounded number of copies of $P$.

The $\pi$-calculus can be varied in many ways. There are many useful subcalculi, e.g. the polyadic $\pi$-calculus (Milner, 1993). The polyadic $\pi$-calculus allows multiple objects in communications: outputs of type $\overline{a}\langle y_1, \ldots, y_n\rangle.P$ and inputs of type $a(x_1, \ldots, x_n).Q$. In this paper, the polyadic $\pi$-calculus is adopted as the modeling tool.
$\pi$-calculus can address the description of system with a dynamic or evolving topology, and analyze the key properties, such as deadlock, bisimulation, and bisimilarity.

### 2.3 The Integration of Petri Nets and $\pi$-calculus

Petri nets are graphical representation and a promising tool to describe the static characteristics of the system, represent the dynamic behaviors, and express causality and concurrency in system behavior. Structural properties of Petri nets, such as P-invariants and T-invariants, are employed to analyze the relations of the structure and behaviors of a system. Furthermore, Petri nets provide a variety of well-established mathematical methods to analyze, simulate and validate the systems. These properties make Petri nets as an excellent tool for the validation of models by non-technical end users. However the structure of Petri nets is static, it is hardly possible to model dynamic system architecture.

$\pi$-calculus is suitable for describing software system with an evolving communication topology. $\pi$-calculus can specify and reason about the design of complex concurrent computing systems by means of algebraic operators corresponding to common programming constructs Best et al. (2001). However,the processes of $\pi$-calculus are complicated, and it cannot visually model the system architecture (Jiang, 2003). Moreover, as $\pi$-calculus, which is based on the interleaving semantics, cannot depict the true concurrency and has few supporting tools.
The treatment of the structure and semantics of concurrent systems provided by Petri nets and $\pi$-calculus is different, so it is virtually impossible to take full advantage of their overall strengths when they are used separately. Therefore the idea of combining Petri nets and $\pi$-calculus is proposed, where Petri nets are employed to visually model the system architecture and system behaviors, and $\pi$-calculus is employed to describe the system evolution. To remedy the deficiencies of $\pi$-calculus, $\pi$-calculus is mapped into Petri nets to visualize system structure as well as system behaviors. Therefore, the structural analysis techniques allow direct qualitative analysis is of the system properties on the structure of the nets.
The use of dual complimentary formal methods has many advantages over a single formalism (Clarke, 1996), including modeling and analyzing different aspects of software architecture using different formalism to improve understandability. The integration of Petri nets and $\pi$-calculus provides a bridge between graphical specification techniques and dynamic modeling techniques. $\pi$-calculus and Petri nets can complement each other very well.

## 3. Software Architecture Formal Model

A visual software architecture formal model (SAFM) based on Object-oriented Petri nets and $\pi$-calculus, is proposed. SAFM models and analyzes software architecture, and it describes the components, connectors and configuration.
*Definition* 3 SAFM is a three-tuple, $SAFM = (Comp, Conn, Conf)$, where $Comp = (Comp_1, Comp_2, \ldots, Comp_o,)$ is a set of components, $Conn = (Conn_1, Conn_2, \ldots, Conn_p)$ is a set of connectors, and $Conf$ is architecture configuration.

### 3.1 Modeling Components

A component is a unit of data or computation, loci of status store and computation with extended and integrated. A Component is 3-tuple, $Comp_o = (ID, OPN, \Pi)$, where $ID$ is the identifier of a component; $OPN$ defines the interfaces and internal implementation of a component; $\Pi$ describes the evolvement of a component by $\pi$-calculus.
In OPN, $IT$ and $OT$ describe a component's interfaces that are a set of interaction points between it and the external world, $Comp_o.Interface = \{(t_1, t_2) | t_1 \in IT, t_2 \in OT\}$. The interface specifies the services a component requires and provides, especially the messages a component receives and sends. The implementation of a component is described by other tuples of OPN. A component interacts with other components by interfaces, and its internal implementation is invisible for other components. The evolve process of components will be described in the next section.
Components are reusable software units, including composite components and atomic components. A composite component may be composed of other composite components or atomic components. An atomic component is no longer divided.

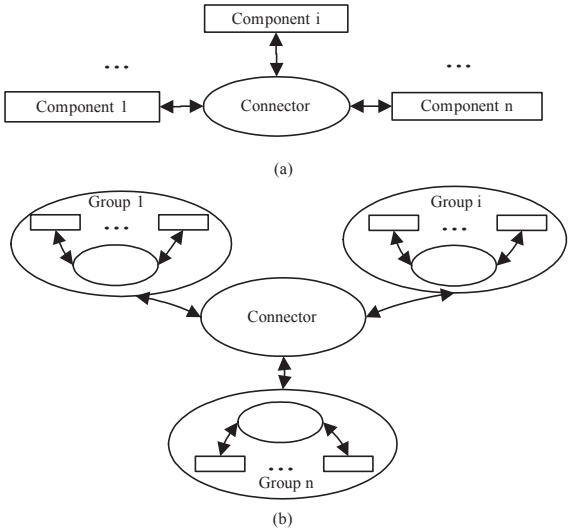



Fig. 1. The connector model

### 3.2 Modeling Connectors

Connectors are used to model the interactions among components, define the rules that govern those interactions. Connectors coordinate and supervise components from the high level, and manage the resources of the system.

A connector is defined as $Conn_p = (ILP, Gate, KBP, Role, \Pi)$, where $ILP$ is a intelligent link place denoted by a ellipse. The information obtained from the external is saved in the ILP to set up message passing channels among components. $Gate$ is the tuple in OPN model. $KBP$ represents Knowledge-base Place which is defined to apperceive the external environment, acquire requisite knowledge, and describe services which components provide via interfaces. $Role$ is a set of components interact with the connector, which is defined as $Role = \{CID_1, \ldots, CID_n\}$. $\Pi$ addresses the evolvement of connectors by $\pi$-calculus, which will be described in the next section.

From the point of view of communication, the connector controls and manages the communication and collaboration among components; from the point of view of the system connection and conglutination, the connector plays the role of the glue conglutinating the software system.

In the connector, the roles identify the logical participants in the interaction. There are two types of roles, static and dynamic role, respectively. Dynamic role will change with the components deleted or added.

A software system may consist of some connectors. If a system is composed of a connector and some components to achieve a certain goal, then it is called a group, which is shown in Fig. 1(a). Fig. 1(b) represents several groups constitute a large-scale system, and these groups is connected by a connector.

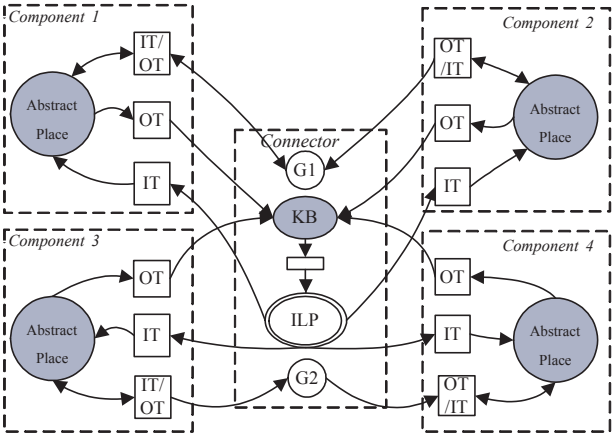



Fig. 2. software architecture configuration

### 3.3 Modeling Configurations

$Conf$ is the architectural configuration, which addresses the connected graphs of components and connectors. $Conf$ is studied from the macro-level, where software systems are conceived as a multitude of interacting components and connectors. In the macro-level, the key point is the overall structure and behaviors, rather than the mere behaviors of individuals.

The architectural configuration is shown in Fig. 2. For simplicity and clarity of the diagrams, this configuration model is predigested. The components are represented by IT, OT and abstract places denoted by shaded circles. The abstract places can be refined according to requirements. The static semantics of architecture is visually described in Fig. 2, and the dynamic semantics is represented by the firing of transitions. The firing of the transition makes the Token dispatch, which expresses the message passing and well depicts interactions among components.

## 4. The Dynamic Evolvement of Software Architecture

To address dynamic software architecture, the scheme in SAFM is defined as follows.

1. The supervising processes are defined in components and connectors. The supervising processes in components send messages to connectors, and describe the internal reason of dynamic evolvement, such as computation errors or abnormity. The supervising processes in connectors interact with the supervising processes in components and environment, and describe the exterior reason of dynamic evolvement.
2. The connectors is the supervisor of a system, and defined the operators, such as $Create, Delete, Update$, to describe the dynamic evolvement.
3. After addressing the dynamic evolvement, the correctness and consistency must be analyzed, which will be described in the next section.

The supervising processes in components can be defined as follows.

$$Monitor(request, config) = \overline{request}\langle id, updinfo\rangle.config(x,y)([x = id, y = begin]Update).$$
$$Reginfo\langle id, s\rangle.Monitor(request, config) \quad (1)$$

where the process *Monitor* sends *id* and update information $updinfo$ to the supervising process in a connector via the channel *request*, and then waits for the notice to update. After the component updates using the process $update$, the process $Reginfo$ will register its related information in the connector.
The supervising processes in connectors can be defined as follows.

$$Supervisor(request, info, config) = request(u, v).\overline{info}\langle ids, wait\rangle.\overline{config}\langle id, begin\rangle. \\ CReginfo\langle x, y\rangle.Supervisor(request, info, config) \quad (2)$$

where the process *supervisor* receives the component updating information via the channel *request*, and notifies the related components suspend their services via the channel $info$, and then accepts the enrollment information of the new component.

### 4.1 Components Joining and Exiting

In software systems, new components first enroll their information (such as name, address, interface and capability) in connectors, and set up the channels for interacting with the other components via $ILP$. The creating process of a component is

$$NewComp(id, s) = Create(id, s) \quad (3)$$

It means that a component is created with a identifier *id* and providing a service $s$. The enroll process of a component is

$$RegInfo(id, s) = (\nu id, s)(\overline{register}\langle id, s\rangle) \quad (4)$$

It means that a component enrolls a service $s$ and its identifier *id* via the channel *register* to a connector, and the $id, s$ is private names. The corresponding enroll process in the connector is

$$CRegInfo(x, y) = (\nu x, y)(register(x, y)) \quad (5)$$

It means that the connector obtains a service information via the channel *register*, and the $x, y$ is private names.
When a component requests a service, the connector queries the knowledge base to search a corresponding component providing the service to send its identifier to the requesting component. If the requesting component receives the identifier of the service component, it sends the message to the service component through the connector; if the service component does not exist, the requesting component can subscribe for this service. The connector will inform the requesting component as long as it becomes aware of the information that a corresponding component registers. This requesting process in the requesting component is

$$RequestService(i, r, l) = \bar{i}\langle a\rangle.r(z).([z = nil].\overline{subscribe}\langle a\rangle + \bar{z}\langle l\rangle) \quad (6)$$

The requesting component sends the request $a$ through the channel $i$ to the connector to query the corresponding service component, and then wait a response from the connector through the channel $r$. After the requesting component receives the identifier of the service component $z$, it sends the requesting address $l$ to the service component by the channel $z$.
The service query process in the connector is

$$QueryService(i, r, p) = i(y).(\overline{kb}\langle y\rangle | Belief(y)).(\bar{r}\langle nil\rangle.subscribe(y) + \bar{r}\langle P\rangle) \quad (7)$$

and the corresponding process in the service component is

$$ProvdService(p,s) = p(x).\overline{x}\langle s\rangle \tag{8}$$

The service component sends the service through the channel $x$ to the requesting component. According to the above analysis, the entire dynamic process of the service requesting and providing is modeled as follows:

$$\begin{aligned}
&RequestService(i,r,l)|QueryService(i,r,p)|ProvdService(p,s)\\
&= \overline{i}\langle a\rangle.r(z).([z=nil].\overline{subscribe}\langle a\rangle + \overline{z}\langle l\rangle)|i(y).(\overline{kb}\langle y\rangle|Belief(y)).(\overline{r}\langle nil\rangle.subscribe(y) +\\
&\overline{r}\langle P\rangle)|p(x).\overline{x}\langle s\rangle\\
&\xrightarrow{\tau} r(z).([z=nil].\overline{subscribe}\langle a\rangle + \overline{z}\langle l\rangle)|(\overline{kb}\langle a\rangle|Belief(a)).(\overline{r}\langle nil\rangle.subscribe(y) +\\
&\overline{r}\langle P\rangle)|p(x).\overline{x}\langle s\rangle\\
&\xrightarrow{\tau} r(z).([z=nil].\overline{subscribe}\langle a\rangle + \overline{z}\langle l\rangle)|\overline{r}\langle P\rangle|p(x).\overline{x}\langle s\rangle\\
&\xrightarrow{\tau} \overline{p}\langle l\rangle|p(x).\overline{x}\langle s\rangle\\
&\xrightarrow{\tau} \overline{l}\langle s\rangle
\end{aligned}$$

If a component achieves its goal and wants to exit from the system, it must delete its information, so the information in the connector will not fall into confusion. The exiting process of a component is

$$Comp(id,s) = Delete(id,s) \tag{9}$$

It means that a component is deleted with an identifier $id$ and a service $s$.

### 4.2 Component Update

Component update can be classified into two categories. 1. The algorithm of a component may be error or its deficiency is lower, so the component must be updated, which is called holding semantic update. 2. For new system requirements appearing, a component with new functions will update the former component, which is called extended update.

For the first case, the weak equivalent of $\pi$-calculus can be used to judge whether the new component substitutes for the old component. If the behaviors of two components are equivalent, a component can substitute for the other one, which the environment cannot apperceive. The holding semantic update is defined as follows.

*Rule* 1 Suppose the behaviors of the components $Comp_0$ and $Comp_1$ are the processes $P$ and $Q$, respectively. If $P \approx Q$, $Comp_1$ can update $Comp_0$ denoted as $P \rhd Q$, which is called holding semantic update.

Holding semantic update means that the internal algorithms of components are updated, while their behaviors are not changed.

For the second case, a powerful component updates the former component. The extended update is defined as follows.

*Rule* 2 Suppose the behaviors of the components $Comp_0$ and $Comp_1$ are the processes $P$ and $Q$, respectively. $P$ and $Q$ satisfy the following conditions.

1. $fn(P) \subseteq fn(Q)$;
2. If $P \xrightarrow{\tau} P'$, then $\exists Q', Q \Rightarrow Q'$;
3. If $P \xrightarrow{x(z)} P'$, then $\exists Q', Q \xrightarrow{x(z)\ldots x_i(z_i)} Q'$($\xrightarrow{x(z)\ldots x_i(z_i)}$means that $Q$ can execute the other actions $x_i(z_i)$ except $x(z)$);

4. If $P \xrightarrow{\overline{x}\langle y\rangle} P'$, then $\exists Q', Q \xrightarrow{\overline{x}\langle y\rangle \ldots \overline{x_i}\langle y_i\rangle} Q'$($\xrightarrow{\overline{x}\langle y\rangle \ldots \overline{x_i}\langle y_i\rangle}$means that $Q$ can execute the other actions $\overline{x_i}\langle y_i\rangle$ except $\overline{x}\langle y\rangle$);
5. If $P \xrightarrow{\overline{x}y} P'$, then $\exists Q', Q \xrightarrow{\overline{x}y \ldots \overline{x_i}y_i} Q'$.

Then $Q$ updates $P$, therefore $Comp_1$ can update $Comp_0$ denoted as $Comp_1 \rhd \rhd Comp_0$, which is called extended update.
Extended update means that the new component provides new functions except remaining the former behaviors.

### 4.3 Load Balancing

In distributed systems, if a new server is added, the requests of clients must be assigned to different servers to balance load. In SAFM, connectors are employed to balance load of servers. For example, if a system consists of two server components, load balancing rule is

$$\begin{aligned} &\text{if}(Server_1.cn < Server_2.cn) \\ &\text{then } QueryService(i_1, r_1, p_1) \\ &\text{else } QueryService(i_2, r_2, p_2) \end{aligned}$$

It means that before the connector sets up a channel for a client and a server, it must judge the number of clients interacting with the two servers. If the number of clients interacting with $Server_1$ is less than that of clients interacting with $Server_2$, the identifier of $Server_1$ is transmitted to the clients.

### 4.4 Architecture Reconfiguration

To improve the stability of systems, some backup components are added. When the primary component goes down, the backup component is used until the primary component returns to service. In Fig. 2, $component_1$ interacts with $component_2$ via the channel $G_1$ (denoted as $y$). During the system running, a backup component $component_{1'}$ is added to backup the data via the channel $bak$. If $component_1$ goes down, $component_2$ needs switch its channel to $component_{1'}$. The dynamic configuration process is shown as follows.

$$\begin{aligned} &(\nu bak)(\overline{y}\langle bak\rangle.Comp_1 | Comp_{1'}) | y(x).Comp_2 \\ &\xrightarrow{\tau} (\nu bak)(Comp_1 | Comp_{1'}) | Comp_2\{bak/x\} \end{aligned}$$

It means that before $component_1$ going down, the private channel $bak$ is transferred to $component_2$. Therefore $component_2$ can interact with $component_1$ via the channel $bak$.

## 5. Analyzing Software Architecture

For analyzing static software architecture, the related methods and supporting tool (such as INA (Roch & Starke, 2009)) can be employed to analyze deadlock, boundness and reachbility of models. In this section, the major attentions will be focused on analyzing dynamic software architecture. Owing to components interacting with each other via interfaces, the internal structures are omitted and the processes of $\pi$-calculus are used to describe the behaviors of interfaces to analyze the consistency of dynamic software architecture. To verify the structure characteristics (Yu et al., 2007), the $\pi$-calculus models are mapped into Petri nets to analyze the liveness of models.

### 5.1 The Compatibility of The Internal Implementations and Interfaces in Components

A component is composed of interfaces and internal implementations. Interfaces represent the providing or needing services of components. The internal implementations of components must be compatible with their interfaces. Therefore, components must execute the behaviors which interfaces represent. In this section, the name hiding (Canal et al., 2001) of $\pi$-calculus are used to analyze the compatibility.

*Definition* 4 (Name Hiding) Suppose a process $P$ and a set of names $N \subseteq fn(P)$, $P/N$ represents the process hides the names belong to $N$ in $P$,

$$P/N = (\nu N)(P| \prod_{n \in N} Hide(n)) \tag{10}$$

where $Hide(n) = n(m).(Hide(n)|Hide(m)) + (\nu m)(\overline{n}\langle m\rangle.(Hide(n)|Hide(m)))$.

For each name $n \in N$, the process $Hide(n)$ hides it in $P$. $Hide(n)$ means that the input or output prefixes of $n$ are provided to interact with the input or output prefixes of $n$ in $P$. Therefore the behavior is predigested to the internal behavior $\tau$ so as to hide the name $n$.

The hide names are distinct with the restricted names. The restricted names cannot interact with the other processes, but they can be regarded as values to transmit to the other processes. The hide names means that the freedom names in processes are hide to transfer some specific behaviors to internal behaviors.

*Definition* 5 (Component Interface) Suppose a component $Comp$ and a process $P$, if

$$fn(P) \subseteq fn(Comp) \text{ and } P \approx Comp/(fn(Comp) - fn(p)) \tag{11}$$

then $P$ is called an interface in $Comp$.

Definition 5 means that a interface is a subset of internal implementations in a component. Therefore the freedom names of the interface is a subset of the freedom names of the component, and the other names are hidden. Then the process is weak equivalent to $P$.

According to definition 5, the interfaces can be obtained by name hiding. However all processes satisfying definition 6 do not correctly represent the internal implementations of a component. The compatibility of the interfaces and implementations of a component is judged by the following conclusion.

*Conclusion* 1 Suppose a component $Comp$ and its set of interfaces $P = \{P_1, \ldots, P_n\}$ satisfy the following conditions.

1. $fn(P_i) \cap fn(P_j) = \varnothing \ \ \forall i \neq j$;
2. $Comp \Rightarrow 0$ iff $\forall i, P_i \Rightarrow 0$($\Rightarrow$represents $(\xrightarrow{\tau})^*$, which shows 0 or some internal evolvement sequence);
3. If $\exists \alpha(\alpha \neq \tau), Comp \xrightarrow{\alpha} Comp'$, then $\exists i, P_i^{'}, P_i \xrightarrow{\alpha} P_i^{'}$ and the internal implementations of $Comp'$ are compatible with $P' = \{P_1, \ldots, P_i^{'}, \ldots, P_n\}$.

Then the internal implementations are compatible with the interfaces $P$.

### 5.2 Analyzing Consistency of Software Architecture

When new components join or components update, software architecture will evolve. Consequently, the consistency may be changed. The consistency means that all components in systems successfully interact with each other, which is important for dynamic software architecture (Goudarzi, 1998) (Canal et al., 2001). In SAFM, components interact with each other

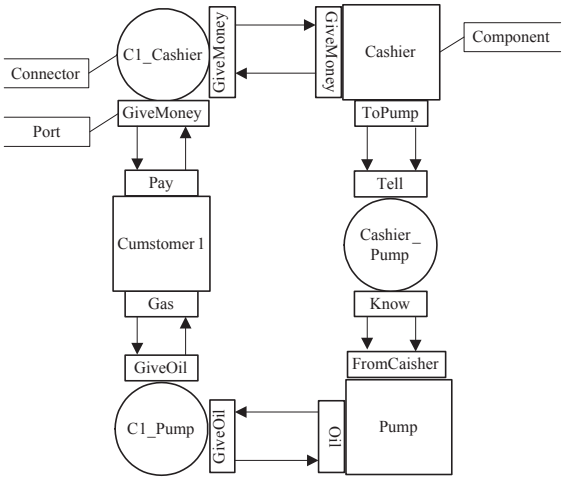



Fig. 3. software architecture model of gas station based on Wright

via interfaces. Therefore, the consistency can be judged from the interface level (Cimpan et al, 2005).

If the processes $P$ and $Q$ are consistent, $P$ interacts with $Q$ via a allelomorph name, such as $P = \overline{x}\langle y\rangle.P',\ Q = x(z).Q'$. Then $P$ and $Q$ are synchronous.

*Definition* 6(Consistency) Suppose the relation $R$ in synchronous processes, for example $PRQ$. If all replacer operators $\sigma \notin fn(P) \cup fn(Q)$ satisfies $P\sigma RQ\sigma$, and

1. If $P \xrightarrow{\tau} P'$, then $P'RQ$;
2. If $\neg(P \sim 0) \wedge \neg(P \equiv 0),\ P \xrightarrow{x(z)} P' \wedge Q \xrightarrow{\overline{x}y} Q'$, then $P'\{y/z\}RQ'$;
3. If $\neg(P \sim 0) \wedge \neg(P \equiv 0),\ P \xrightarrow{x(n)} P' \wedge Q \xrightarrow{\overline{x}(n)} Q'$, then $P'RQ'$;

Then $R$ is called half-consistency. If $R$ and $R^{-1}$ are half-consistency, then they are consistency, denoted as $\backsim$.

The consistency of processes means the processes can communicate with allelomorph names and execute successfully until their final states. If a process $P'$ makes $P \Rightarrow P'$ and $P' \xrightarrow{\tau}$ or $P' \equiv 0$ ($P' \xrightarrow{\tau}$ means $\exists P'',\ P' \xrightarrow{\tau} P''$), then $P$ can successfully execute until its final states. If $P \Rightarrow P',\ \neg(P' \xrightarrow{\tau})$ and $\neg(P' \equiv 0)$, then $P$ is deadlock.

*Conclusion* 2(Consistency of Components) Suppose $P$ and $Q$ are interfaces of $Comp_1$ and $Comp_2$, and $P$ and $Q$ are consistent, $P \backsim Q$. Then $Comp_1$ and $Comp_2$ are consistent, namely, $Comp_1|Comp_2$ can successfully execute.

*Conclusion* 3(Consistency of System) If all components in a system are consistent, then the system can successfully execute.

## 6. Example: Modeling and Analyzing Gas-Station Problem

The gas station system consists of customers, cashiers and pumps (Tsai & Xu, 1999). If a customer arrives at the gas station, he pays for the gas, then the cashier informs the pump.

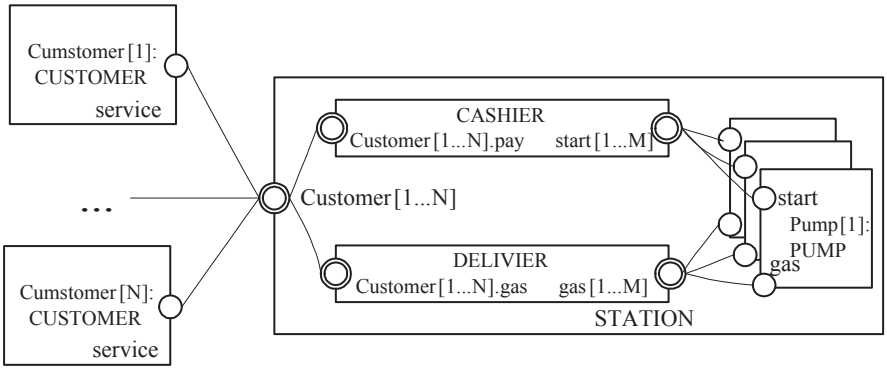



Fig. 4. software architecture model of gas station based on Darwin

Suppose that there exist a customer, a cashier, and a pump in gas station. The software architecture of the gas station using Wright is shown in Fig. 3 (Naumovich et al., 1997), where the arrows represent the connecting attachment. For this gas station, 3 connectors and 12 attachments are needed using Wright, and each connector, component, and port must be described by CSP processes. Therefore, the system model is complicated. The gas station system is modeled by Darwin shown in Fig. 4 (Magee et al., 1999). There no exist connectors in this system. Customers use the same port to pay for expenses and pump oil. Consequently collision may be occurred (Cuesta et al., 2005).

In this section, the gas station is modeled by SAFM shown in Fig. 5, which is abstracted as customer, cashier, pump and connector. The customer, cashier and pump are enrolled in the connector via the transition $RegInfo$. The connector sets up the $pay_1$, $pump_1$ and $info$ via $SevInfo$. Comparing to Wright and Darwin, the system model using SAFM is visual and easily understand, and void the collision in Darwin.

The Petri nets supporting tools– INA can be employed to analyze and verify the software architecture model of gas station. According to the INA analysis results, the model shown in Fig. 5 is bounded, the number of reachable states is 120, and it is live. Therefore, the gas station can successfully execute.

When a new customer arrives or the system provides new functions, the architecture evolves. Suppose that a new customer arrives, it firstly enroll in connector and the channel $pay_2$, $pump_2$ and *change* are set up. Finally, according to the consistency of components, the consistency of the system needs to be analyzed. The interfaces of components in gas station are defined as follows.

$Customer_2$:

$$Pay(money) = \overline{pay_2}\langle money\rangle.(change(u).Pay(money) + paymore(p).Pay(p)) \tag{12}$$

$$PumpGas(x) = pump_2(x).PumpGas(x) \tag{13}$$

*Cashier*:

$$Charge(y) = pay_2(y).([y \geq expense]\overline{change}\langle dibs\rangle.Charge(y) + [y < expense] \\ \overline{paymore}\langle expense - y\rangle.Charge(expense - y)) \tag{14}$$

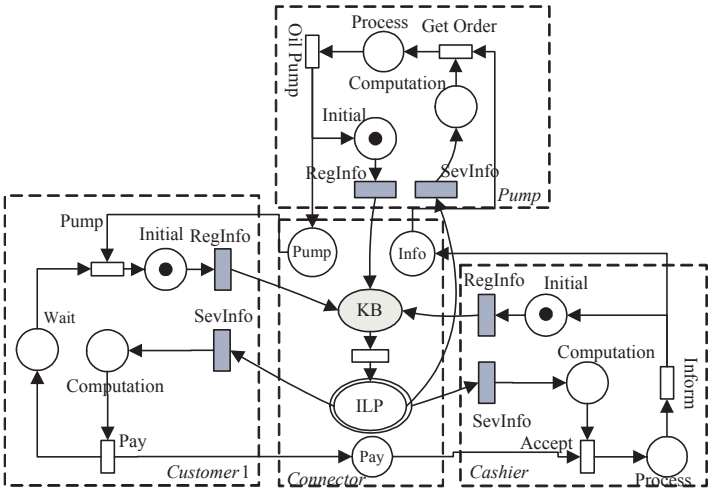



Fig. 5. software architecture model of gas station based on SAFM

$$Inform(id, msg) = \overline{info}\langle id, msg\rangle.Inform(id, msg) \tag{15}$$

*Pump*:

$$Getinfo(z, w) = info(z, w).Getinfo(z, w) \tag{16}$$

$$Pump(gas) = \overline{pump_2}\langle gas\rangle.Pump(gas) \tag{17}$$

The former interfaces of customer and cashier are

$$Pay'(money) = \overline{pay}\langle money\rangle.Pay'(money) \tag{18}$$

$$Charge'(y) = pay(y).Charge'(y) \tag{19}$$

The interfaces of customer and cashier provides new functions. According to rule 2, the cashier components is extended update. According to definition 6, $Pay \backsim Charge$, $Inform \backsim Getinfo$, $Inputgas \backsim Pump$, then components $Customer_2|Cashier|Pump$ can successfully execute. Consequently the gas station can run.

To analyze and verify the evolved model, it is translated into Petri nets by using the algorithm of $\pi$-calculus mapping Petri nets, which is shown in Fig. 6. The pump component is represented as an abstract place. The channels $pay_2$, $pump_2$, *change* and *repay* are mapped into places in the connector, and the interface transitions are added between the customer and cashier. INA is used to analyze the evolved model. According to the INA analysis results, the evolved model is deadlock, and there exists a dead reachable state $S_{137}$. By tracing the firing sequence of $S_{137}$ and analyzing the system flow, an arc is added between $T_{18}$ and $P_{10}$. Finally, the revised model is analyzed by INA. According to the INA analysis results, the revised model is bounded, the number of reachable states is 168, and it is live. Therefore, the system can successfully execute.

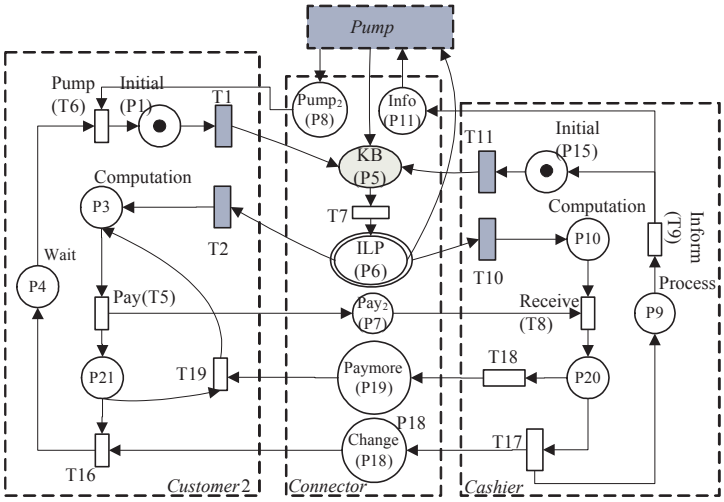



Fig. 6. the evolved software architecture model

According to the above model, the prototype system has been finished. SAFM also is applied to an engagement among Unmanned Ground Vehicles(UGV) (Yu, 2006).

## 7. Conclusion

Based on two complementary formalisms, namely Object-oriented Petri nets and $\pi$-calculus, software architecture formal model (SAFM) is proposed, which describes the components, connectors and configuration. OPN are employed to visualize the static architecture and depict the behavior of software systems, and $\pi$-calculus is used to describe software architecture evolution. SAFM stresses description of dynamic software architecture, and analyze the static and dynamic semantics, and depict the overall and individual characteristics of software architecture. SAFM can be applied to investigate software architecture from the micro-level and macro-level. From the micro-level, the researchers can pay more attention to the implementation details of each component; and from the society level, they can pay more attention to the overall design and interactions among components. An example of software architecture is used to illustrate modeling capability of SAFM.

## 8. References

Shaw, M.; Clements, P. (2006). The golden age of software architecture. *IEEE Software*, Vol. 23, No. 2: 31-39.

Oquendo F. (2006). $\pi$-ADL: an architecture description language based on the higher-order typed $\pi$-calculus for specifying dynamic and mobile software architectures. *ACM Software Engineering Notes*, Vol. 29, No. 4: 1-13.

Medvidovic, N.; Taylor, R. N. (2000). A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, Vol. 26, No. 1: 70-93.

He, X.; Yu, H.; Shi, T.; Ding, J.; et al.(2004). Formally analyzing software architectural specifications using SAM. *Journal of Systems and Software*, Vol. 71, No. 1-2: 11-29.

Shaw, M.; DeLine, R.; Klein, D. V.; et al. (1995). Abstractions for software architecture and tools to support them. *IEEE Transactions on Software Engineering*, Vol. 21, No. 4: 314-335.

Magee, J.; Delay, N.; Eisenbach, S.; Kramer, J. (1995). Specifying distributed software architectures, *Proceedings of the 5th European Software Engineering Conference*, Sitges.

Luckham, D. C.; Kenney, J. J.; Augustin, L. M.; et al. (1995). Specification and analysis of system architecture using Rapide. *IEEE Transactions on Software Engineering*, Vol. 21, No. 4: 336-355.

Allen, R.; Douence, R. & Garlan, D. (1998). Specifying and analyzing dynamic software architectures. *Lecture Notes in Computer Science*, Vol. 1382.

Luo H., Tang Z. & Zheng J. (2000). Visual architecture description language XYZ/ADL. *Journal of Software*, Vol. 11, No. 8: 1024-1029. (in Chinese with English abstract).

Yu, Z.; Cai, Y.; & Xu, H. (2007). On Petri nets semantics for $\pi$-calculus. *Control and Decision*, Vol. 22, No. 8: 864-868. (in Chinese with English abstract).

Miyamoto, T. & Kumagai S. (2005). A survey of Object-Oriented Petri nets and analysis methods. *IEICE Transaction on Fundamentals*, Vol. E88-A, No. 11: 2964-2971.

Yu, Z. (2006). *Approaches to formal modeling methodology for multi-agent systems* [Ph.D. Thesis]. Xi'an: Xi'an Jiaotong University.

Milner, R.; Parrow, J. & Walker D. (1992). A calculus of mobile processes. *Journal of Information and Computation*, Vol. 100, No. 1: 1-77.

Milner, R. (1993). *The polyadic $\pi$-calculus: a tutorial*, Springer-verlag.

Jiang, C. (2003). *Behvior theory and applications of Petri net*, Beijing: Higher Education Press.

Clarke, E. & Wing J. (1996).Formal methods: state of the art and future. *ACM Computing Surveys*, Vol. 28, No. 4: 626-643.

Best, E.; Devillers, R. & Koutny, M. (2001). *Petri Net Algebra*, Spring-Verlag.

Canal, C.; Pimentel, E. & Troya J. M. (2001). Compatibility and inheritance in software architectures. *Science of Computer Programming*, Vol. 41, No. 9: 105-138.

Goudarzi, K. (1998). *Consistency preserving dynamic reconfiguration of distributed systems* [Ph. D. thesis]. Imperial College London.

Cimpan, S.; Leymonerie, F. & Flavio Oquendo, F. (2005). Handling dynamic behaviour in software architectures. *Lecture Notes in Computer Science* , Vol. 3527: 77-93.

Tsai, J. J. P. & Xu, K. (1999). An empirical evaluation of deadlock detection in software architecture specifications. *Annals of Software Engineering*, Vol. 7, No. (1-4): 95-126.

Ma, X.; Yu, P.; Tao, X.; Lv, J. (2005). A service-oriented dynamic coordination architecture and its supporting system. *Chinese Journal of Computer*, Vol. 28, No. 4: 467-477.

Cuesta, C. E.; de la Fuente, P., Barrio-Solorzano, M.; Beato, M. E. (2005). An "abstract process" approach to algebraic dynamic architecture description. *Journal of Logic and Algebraic Programming*, Vol. 63, No. 2: 177-214.

Canal, C.; Pimentel, E. & Troya, J. M. (1999). *Software architectures*. Kluwer Academic Publishers.

Roch, S. & Starke P. H. (2009). *INA: integrated net analyzer, Version 2.2*, www2.informatik.huberlin.de/~ starke/ina.html.

Naumovich, G.; Avrunin, G. S.; Clarke, L. A.; Osterweil, L. J. (1997). Applying static analysis to software architectures. *Lecture Notes in Computer Science*, Vol. 1301, 77-93.

Magee, J.; Kramer, J. & Giannakopoulou D. (1999). *Behaviour analysis of software architectures.* Kluwer Academic Publishers: Software Architecture.

5

# Systolic Petri Nets

Alexandre Abellard and Patrick Abellard
*HandiBio EA4322, IUT, Toulon University*
*France*

## 1. Introduction

In many research fields and applications requiring real time, such as signal, speech or image processing, problems are often characterized by the amount of data to deal with.
However, it can happen that real time constraints are difficult to satisfy without taking into account the intrinsic parallelism of processings to perform. Thus, diverse architectures appeared (SIMD, MIMD...), leading to different classes of architecture. Literature showed the advantages and drawbacks of each of them. Moreover, expensive costs limited their applications during a long time.
Systolic architectures defined by H.T. Kung are a particular class of parallel architectures. They constitute specialized systems characterized by a repetitive structure of identical elementary processors locally and regularly interconnected. Synchronous data circulate through the architecture, which interact at each encounter. Several important difficulties like the central memory sharing, buses access conflicts... can be therefore avoided. All problems are however not systolizable. These networks are designed for the repetitive identical processing of a huge amount of data, which is the case, for example, in many signal and image processing algorithms.
The conception of these networks has been the subject of many studies, the main ones are developped in the first part of this chapter. The limited number of systolic processors available on the market was problematic for the development of methodologies for a long time. Now, programmable components enable to be free from this problem. To ease their implementation, we developped a methodology based on a formal and universal tool (Petri Nets) that is developped in the second part of the paper.

## 2. Definitions

### 2.1 Systolic architecture

This architecture is considered as an intermediate between data flow and pipeline. It has been introduced in (Kung, 1982) and is made of a set of processors or processing cells locally interconnected. Each cell can run a simple or complex operation and links between cells are established so as to minimize the associated paths complexity.

Just like in a pipeline structure, information move in a cascade scheduling form. Communications between outside environment and systolic network are established with peripheral cells that constitute the network I/O ports.

### 2.2 Systolic networks properties

Systolic networks have interesting properties("Quinton & Robert, 1991") :
- Data flow coming from the environment are intensively used
- Networks association is made easier thanks to structure cascadability
- Elementary cells are not complex
- Data flow are simple and regular
- A set of elementary processings is performed synchronously on different cells

These main properties enable to simplify their implementation on VLSI once an automated and rigorous conception method has been defined.

Several kinds of nets exist, according to different basic cells (Fig. 1) : the linear array, orthogonal array and hexagonal array. All can be unidirectional or bidirectional. In this paper, we will mostly focus on linear arrays since it better suits our application, Fig. 1 therefore does not show all possible propagation directions in arrays.

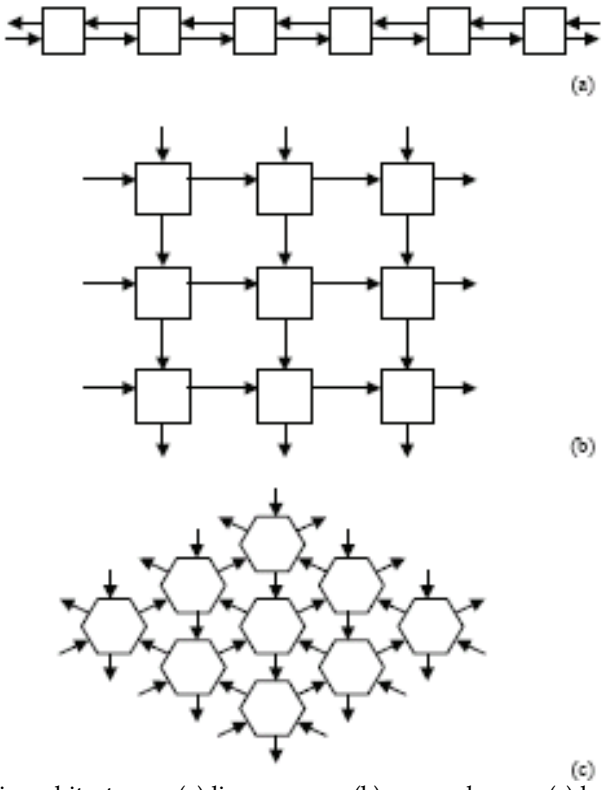



Fig. 1. Basic systolic architectures : (a) linear array, (b) squared array, (c) hexagonal array

These architectures take advantage of the massive parallelism encountered in processing applications (Johnson & Hurson, 1993). They only need a minimum of operators (Sousa, 1998) and memory accesses (Kung, 1988) thanks to a very efficient communications system (Lim & Swartzlander, 1996a). Their combination allows to obtain arrays (Lim & Swartzlander, 1996a) that can be used in a wide range of applications : Discrete Fourier Transform (Lim & Swartzlander, 1999b) (Jackson et al., 2004) (Nash, 2005), convolution (Lee & Song, 2003), filtering (Lee & Song, 2004), matrix operations (Yang et al., 2005), dynamic programming (Lee & Song, 2002).... The regularity of their structures facilitate their hardware, implementation, for instance in FPGAs (Mihu et al., 2001) (Nash, 2002) (Castro-Pareja et al., 2004).

## 2.3 Principles

### 2.3.1 Example of a linear network

Linear equation solving is done thanks to the following equation:

$$y_i^{k+1} = a_{i,k+1}.x_{k+1} + y_i^k \text{ , } 0 \le k \le n\text{-}1, 1 \le i \le n \quad (1)$$

Systolic network defined by Kung is established with a group of interconnected processors, each having 3 registers : $R_y$ for $y^k$, $R_a$ for $a_{i,k}$ and $R_x$ for x. Each register has a connection as input and another one as output. Kung defined 2 kinds of cells : squared (Fig. 2a) and hexagonal (Fig. 2b).

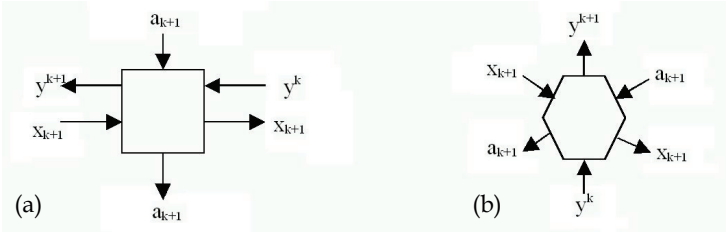



Fig. 2. Square (a) and hexagonal (b) cells

These two kinds of cells work in this operating cycle.
1 – the cell loads inputs $y^k$, $x_{k+1}$ and $a_{i,k+1}$ in respective registers $R_y$, $R_x$ and $R_a$.
2 – $y^{k+1}$ is processed using equation (1)
3 – $y_i^{k+1}$, $x_{k+1}$ and $a_{i,k+1}$ are transferred to the output
Example of the Matrix-Vector Product (MVP)

$$Y = A \,.\, X \quad (2)$$

Relations to implement are then :

$$y_i^{k+1} = a_{i,\,k+1} \,.\, x_{i,\,k+1} + y_i^k \quad (3)$$
$$y_i^0 = 0 \; ; \; y_i = y_i^W$$

with W=dim(X), 0≤k ≤n-1, 1≤i≤n

For example, with W = 3 :

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Equation (3) then gives :

$$y_1 = y_1^0 + y_1^1 + y_1^2 + y_1^3 \Rightarrow y_1 = a_{11}.x_1 + a_{12}.x_2 + a_{13}.x_3$$
$$y_2 = y_2^0 + y_2^1 + y_2^2 + y_2^3 \Rightarrow y_2 = a_{21}.x_1 + a_{22}.x_2 + a_{23}.x_3$$
$$y_3 = y_3^0 + y_3^1 + y_3^2 + y_3^3 \Rightarrow y_3 = a_{31}.x_1 + a_{32}.x_2 + a_{33}.x_3$$

Given an elementary recurrence relation :

$$y_i^{k+1} = a_{i,\,k+1} \,.\, x_{i,\,k+1} + y_i^k$$

successive recurrence steps are performed at consecutive instants. The use of parallelism done via several cells enable to perform a step of many different elementary recurrences at the same time. Systolic network is therefore made of a set of (2.*n*-1) linearly interconnected squared cells, each one receiving $y_i^k$, $x_{k+1}$ and $a_{i,k+1}$ at each step of time $t_j$ (Fig. 3).

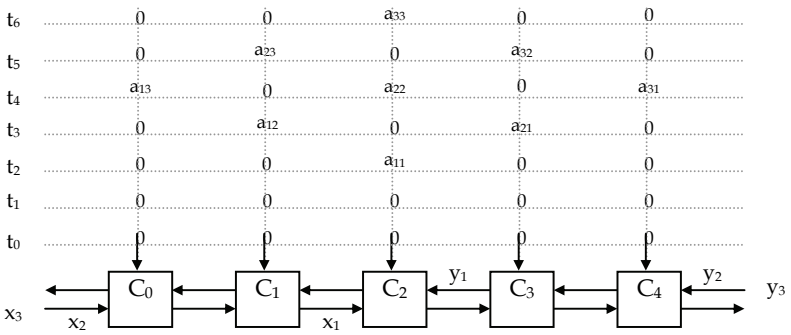



Fig. 3. Linear systolic network of matrix-vector product *Y*=*A*.*X*, *n*=3

$x_1$ is used by $C_0$ cell at $t_0$, then is transmitted to $C_1$ that processes it at $t_1$ and so on from left to right. A similar processing is done for $y_i$ data from right to left. Fig. 4 shows the detail of data propagation on cells performing $y_i^{k+1} = a_{i,\,k+1} \,.\, x_{i,\,k+1} + y_i^k$ .

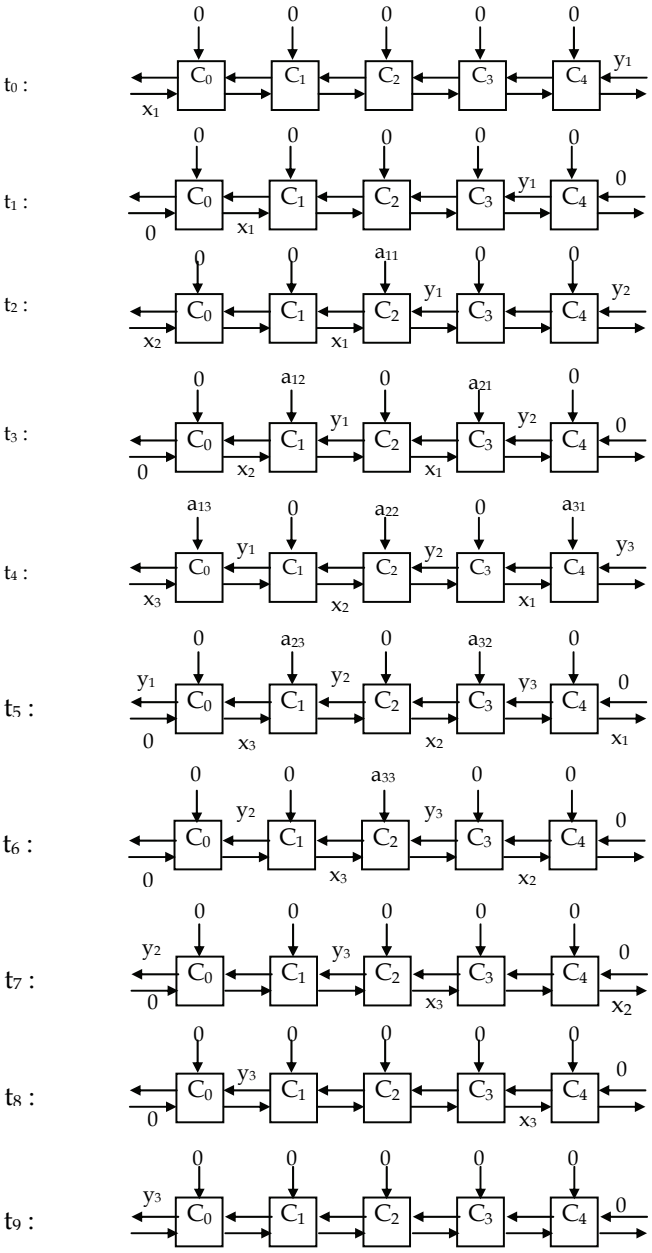



Fig. 4. Linear systolic network processing for matrix-vector product $Y=A.X$ (size 3)

### 2.3.2 Example of a bi-dimensional network

Consider now the matrix product : C = A.B, being sizes of A (m,n), B (n,p) and C (m,p). Each coefficient of C is processed using :

$$c_{i,j} = \text{Sum}(a_{i,k}.b_{k,j})_{\ k=1..n}\ ,\ 1 \le i \le n,\ 1 \le j \le n,\ 1 \le k \le n \qquad (4)$$

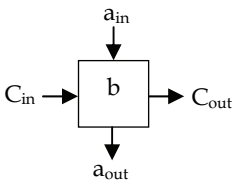



Fig. 5. Elementary cell

$$a_{out} = a_{in}\ ;\ C_{out} = C_{in} + a_{in}.b$$

This relation can be expressed via a recurrence relation like in 2.3.1. Coefficients $a_{ik}$ propagate on j axis, and $b_{kj}$ propagate on i axis. k is a recurrence axis that can be assimilated to a temporal axis. Elementary processings given by (4) are all identical. Network is thus made of a sole kind of cell (Fig. 5) that can be associated in square.

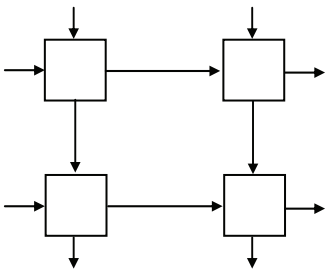


Fig. 6. Example of a 2×2 squared network

Data propagation is detailed on Fig. 7. Other organisation and data propagation possibilities exist. Hexagonal cells can also be used for this processing. In these cells, data propagate in three directions (Fig. 8) so as to be used by neighbouring ones (Fig. 9).

Conceiving systolic networks depends on the problem to be solved and the forced constraints (minimizing number of cells, data flow…). As a consequence, there is no unique method on conception. Several methods exist, some using mathematical equations of the problem to solve, others using problem algorithms. This next section will deal with these questions.

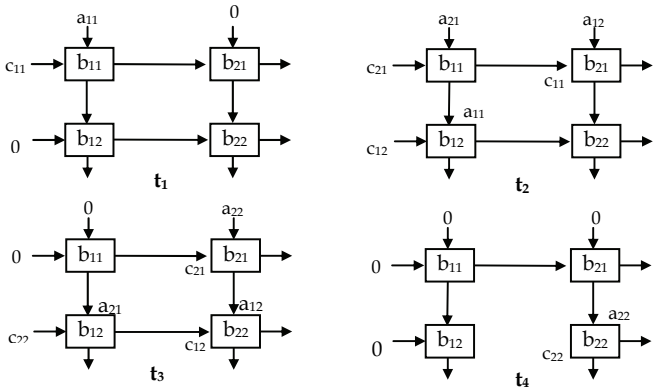
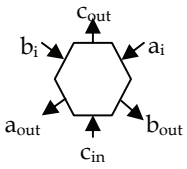



Fig. 7. Data propagation in a squared network




$a_{out} = a_{in}$
$b_{out} = b_{in}$
$c_{out} = c_{in} + a_{in}.b_{in}$

Fig. 8. Elementary hexagonal cell

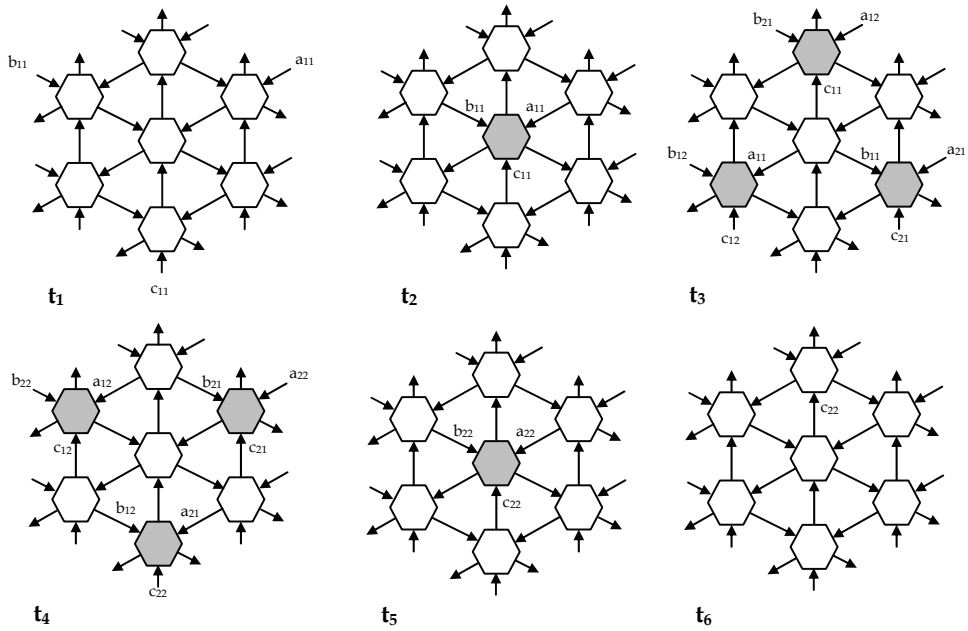



Fig. 9. Hexagonal systolic network operating cycle

## 3. Equation-solving based methods

Among the various approaches done, the three main ones respectively use recurrent equations, sequential algorithms transformation and fluency graphs.

### 3.1 Recurrent equations based method

#### 3.1.1 Quinton method

It is based on the use of geometrical domain projection representing the processing to be done so as to define systolic structures (Quinton, 1983). It has three steps :
- Expressing the problem by a set of uniform recurrent equations on a domain $D \subset \mathbf{Z}^n$
- From this set of equations, defining a temporal function so as to schedule processings
- Defining one or several systolic architectures by applying processing allocation functions to elementary cells

These functions are determined by the different processing domain projections.

##### 3.1.1.1 Step 1 : Creating recurrent equations

Be $\mathbf{R}^n$, the $n$-dimension real numbers space, $\mathbf{Z}^n$ its subset with integer coordinates and $D \subset \mathbf{Z}^n$ the processing domain. On each point $z$ from $D$, a set of equations E($z$) is processed :

$$\begin{aligned} u_1(z) &= f(u_1(z+\theta_1), u_2(z+\theta_2), ..., u_m(z+\theta_m)) \\ u_2(z) &= u_2(z+\theta_2) \\ &... \\ u_m(z) &= u_m(z+\theta_m) \end{aligned} \quad (5)$$

in which vectors $\theta_i \subset \Theta$ called dependency vectors are independent from z. They define which are the values where a point of the domain must take its input values. This system is uniform since $\theta_I$ does not depend on z and the couple (D, Θ) represents a dependency graph. Thus, the processing of A and B (2 n×n-matrices) is defined by :

$$c_{ij} = \mathrm{Sum}(a_{ik}.b_{kj})_{k=1..n} \text{ , } 1 \le i \le n, 1 \le j \le n$$

It can be defined by the following uniform recurrent equations system :

$$\begin{aligned} c(i,j,k) &= a(i,j,k-1)+a(i,j-1k).b(i-1,j,k) \\ a(i,j,k) &= a(i,j-1,k) \\ b(i,j,k) &= a(i-1,jk) \end{aligned} \quad (6)$$

Several possibilities to propagate data on $i$, $j$ and $k$ axis exist. $a_{ik}$, $b_{kj}$ and $c_{ij}$ are respectively independent from $j$, $i$ and $k$, the propagation of these 3 parameters can be done following the ($i$,$j$,$k$) trihedron. The processing domain is the cube defined by D = {(i,j,k), 0≤i≤n, 0≤j≤n, 0≤k≤n}. Dependency vectors are $\theta_a$ = (0, 1, 0) , $\theta_b$ = (1, 0, 0) , $\theta_c$ = (0, 0, 1). With n=3, dependency graph can be represented by the cube on Fig. 10. Each node corresponds to a processing cell. Links between nodes represent dependency vectors. Other possibilities for data propagation exist.

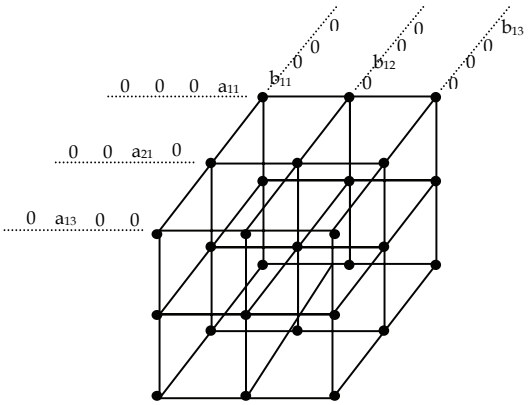



Fig. 10. Dependency domain for matrix product

#### 3.1.1.2 Step 2 : Determining temporal equations

The second step consists in determining all possible time functions for a system of uniform recurrent equations. A time function $t$ is from $D \subset \mathbf{Z}^n \rightarrow \mathbf{Z}^n$ that gives the processing to perform at every moment. It must verify the following condition :

If $x \in D$ depends on $y \in D$, i.e. if a vector dependency $\Theta_i = \overrightarrow{yx}$ exists, then $t(x) > t(y)$.

When $D$ is convex, analysis enables to determine all possible quasi-affine time functions. In this aim, following definitions are used :

- $D$ is the subset of points with integer coordinates of a convex poyedral $\underline{D}$ from $\mathbf{R}^n$.
- $\text{Sum}(\mu_i.x_i)_{i=1...m}$ is a positive combination of points $(x_1, \ldots, x_n)$ from $\mathbf{R}^n$ if $\forall i , \mu_i > 0$
- $\text{Sum}(\alpha_i.x_i)_{i=1...m}$ is a convex combination of $(x_1, \ldots, x_n)$ if $\text{Sum}(\alpha_i)_{i=1...m} = 1$
- $s$ is a summit of $\underline{D}$ if $s$ can not be expressed as a convex combination of 2 different points of $\underline{D}$
- r is a radius of $\underline{D}$ if $\forall x \in \underline{D}$, $\forall \mu_i \in \mathbf{R}^+$, $x+\mu_i.r \in \underline{D}$
- a radius r of $\underline{D}$ is extremal if it can not be expressed as a positive convex combination of other radii of $D$.
- l is a line of $\underline{D}$ if $\forall x \in \underline{D}$, $\forall \mu_i \in \mathbf{R}$, $x+\mu_i.l \in \underline{D}$
- if $\underline{D}$ contains a line, $\underline{D}$ is called a cylinder

If we limit to convex polyedral domains that are not cylinders, then the set $S$ of summits of $\underline{D}$ is unique as well as the set $R$ of $\underline{D}$ extremal radii. $\underline{D}$ can then be defined as the subset of points x from $\mathbf{R}^n$ with x = y + z, y being a convex combination of summits of $S$ and z a positive combination of radii of $R$.

**Definition 1.** $T = (\lambda, \alpha)$ is a quasi-affine time function for $(D, \Theta)$ if $\forall \theta \in \Theta$, $\lambda^T.\theta \geq 1$, $\forall r \in R$, $\lambda^T.r \geq 0$, $\forall s \in S$, $\lambda^T.s \geq \alpha$

Thus, for the uniform recurrent equations system defining the matrix product, $(\lambda,\alpha)$ time functions meets the following characteristics :

$$\lambda^T=(\lambda_1, \lambda_2, \lambda_3) \text{ with } \lambda_1 \geq 1, \lambda_2 \geq 1, \lambda_3 \geq 1 \text{ and } \lambda_1 + \lambda_2 + \lambda_3 > 1.$$

A possible time function can therefore be defined by $\lambda^T = (1,1,1)$, with the following 3 radii (1,0,0), (0,1,0) and (0,0,1).

#### 3.1.1.3 Step 3 : Creating systolic architecture

Last step of the method consists in applying an allocation function $\xi$ of the network cells. This function $\xi=a(x)$ from D to a finite subset of $\mathbf{Z}^m$ where m is the dimension of the resulting systolic network, must verify the following condition (t : time function seen on 3.1.1.2) that guarantees that two processings performed on a same cell are not simultaneous :

$$\forall x \in D, \forall y \in D, a(x)=a(y) \Rightarrow t(x) \neq t(y).$$

Each cell has an input port $I(\mu_i)$ and an output port $O(\mu_i)$, associated to each $\mu_i$ , defined in the system of uniform recurrent equations. $I(\mu_i)$ of cell $C_i$ is connected to $O(\mu_i)$ of cell $C_{i+a.\theta i}$ and $O(\mu_i)$ of cell $C_i$ is connected to $I(\mu_i)$ of cell $C_{i-a.\theta i}$ . Communication time between 2 associated ports is $t(\theta_i)$ time units. For the matrix product previously considered, several allocation functions can be defined. :

- $\xi$ = (0,0,1) or (0,1,0) or (1,0,0), respectively corresponding to a(i,j,k)=k, a(i,j,k)=j, a(i,j,k)=i. Projection of processing domain in parallel of one of the axis leads to a squared shape
- $\xi$ = (0,1,1) or (1,0,1) or (1,1,0), respectively corresponding to a(i,j,k)=j-k, a(i,j,k)=i-k, a(i,j,k)=i-j. Projection of processing domain in parallel of the bisector lead to a mixed shape
- $\xi$ = (1,1,1). Projection of processing domain in parallel of the trihedron bisector lead to a hexagonal shape.

Li and Wah method (Li & Wah, 1984) is very similar to Quinton, the only difference is the use of an algorithm describing a set of uniform recurrent equations giving data spatial distribution, data time propagation and allocation functions for network building.

### 3.1.2 Mongenet method

The principle of this method lies on 5 steps (Mongenet, 1985) :

- systolic characterization of the problem
- definition of the processing domain
- definition of the generator vectors
- problem representation
- definition of associated systolic nets

#### 3.1.2.1 Systolic characterization of the problem

The statement characterizing a problem must be defined with a system of recurrent equations in $\mathbf{R}^3$ :

$$\begin{aligned} y_{ij}^k &= f(y_{ij}^{k-1}, a^1, ..., a^n) \\ y_{ij}^k &= v, v \in \mathbf{R}^3 \\ 0 \leq k \leq b, &\ i \in I, j \in J \end{aligned} \qquad (7)$$

in which $a^1, \ldots, a^u$ are data, I and J are intervals from $\mathbf{Z}$, $k$ being the recurrency index and b the maximal size of the equations system.

$a^q$ elements can belong to a simple sequence ($s_l$) or to a double sequence ($s_{l,l'}$), $l \in L$, $l' \in L'$, $L$ and $L'$ being intervals of **Z**. In this case, $a^q$ elements are characterized by their indexes which are defined by a function $h$ depending on $i$, $j$ and $k$. The result of the probem is a double sequence ($r_{ij}$), $i \in I$, $j \in J$ where $r_{ij}$ can be defined in two ways :
− the result of a recurrency $r_{ij} = y_{ij}^b$
− $r_{ij} = g(y_{ij}^b, a^1, ..., a^n)$
For example, in the case of resolving a linear equation, results are a simple suite $y_i$, , $1 \le i \le n$ , each $y_i$ being the result of the following recurrency :

$$\begin{gathered} y_i^{k+1} = y_i^k + a_{i,k+1}.\, x_{k+1} \\ y_i^0 = 0 \\ 0 \le k \le n-1,\ 1 \le i \le n \end{gathered} \tag{8}$$

### 3.1.2.2 Processing domain

The second step of this method consists in determining the processing domain D associated to a given problem. This domain is the set of points with integer coordinates corresponding to elementary processings. It is defined from the equations system defining the problem.
**Definition 2.** Consider a systolizable problem which recurrent equations are similar to (7) and defined in $\mathbf{R}^3$. The D domain associated to the problem is the union of two subsets $D_1$ and $D_2$.:
- $D_1$ is the set of indexes values defining the recurrent equations system. $b$ being a bound defined by the user, it is defined as $D_1 = \{ (i,j,k) \subset \mathbf{Z}^3, i \in I, j \in J, a \le k \le b\}$
- $D_2$ is defined as :
  - if the problem result is ($r_{ij}$) : $i \in I, j \in J \mid r_{ij} = y_{ij}^b$ , then $D_2 = \varnothing$
  - if the problem result is ($r_{ij}$) : $i \in I, j \in J \mid r_{ij} = q(y_{ij}^b, a^1, ..., a^u)$ , then $D_2 = \{ (i,j,k) \subset \mathbf{Z}^3, i \in I, j \in J, k = b+1 \}$

In the case of the MVP defined in (8), $D_1 = \{ (i,k) \in \mathbf{Z}^2 \mid , 0 \le k \le n-1, 1 \le i \le n\}$ and $D_2$ is empty, since an elementary result $y_i$ is equal to a recurrency result..
**Definition 3**. Systolic specification of a defined problem in $\mathbf{R}^3$ from p data families implies that $D \subset \mathbf{Z}^3$ defines the coordinates of elementary processings in the canonical base ($b_i$, $b_j$, $b_k$). For example, concerning the MVP previously defined, $D = \{ (i,k) \in \mathbf{Z}^2 \mid , 0 \le k \le n-1, 1 \le i \le n\}$.

### 3.1.2.3 Generating vectors

**Definition 4.** Let's consider a problem defined in $\mathbf{R}^3$ from p data families, and d a data family which associated function $h_d$ is defined in the problem systolic specification.
$\psi_d$ is called a generating vector associated to the $d$ family, when it is a vector of $\mathbf{Z}^3$ which coordinates are ($\psi_i$ ,$\psi_j$ ,$\psi_k$) in the canonical base BC of the problem, such as :
- for a point (i , j , k) of the D domain, $h_d(i, j, k) = h_d(i+\psi_i , j+\psi_j , k+\psi_k)$
- highest common factor (HCF) is : $HCF(\psi_i ,\psi_j ,\psi_k) = +1$ or $-1$

This definition of generating vectors is linked to the fact that (i, j, k) and ($i+\psi_i$, $j+\psi_j$, $k+\psi_k$) points of the domain, use the same occurrence of the d data family.
The choice of $\psi_d$ with coordinates being prime between them enables to limit possible choices for $\psi_d$ and to obtain all points ($i+nx\psi_i$, $j+\psi_j$, $k+\psi_k$), $n \in \mathbf{Z}$, from any (i, j, k) point of D.
In the case of the matrix-vector product, generating vectors $\Psi_y = \Psi_a = \Psi_x = (\psi_y , \psi_a , \psi_x)$ are associated to results $h_y$, $h_a$ and $h_x$. Generating vectors are as following :

$h_y(i,k)=h_y(i+\psi_i,\ k+\psi_k) \Leftrightarrow i = i+\psi_i \Leftrightarrow \psi_i = 0$. Moreover, $HCF(\psi_i,\ \psi_k)=\pm 1$, thus $\psi_k=\pm 1$. Generating vector $\psi_y$ can therefore be (0, 1) or (0, -1).
$h_x(i,k) = i+k$. Generating vector $\Psi_a$ must verify $h_a(i,k)=h_x(i+\psi_i,\ k+\psi_k) \Leftrightarrow i+k=i+k+\psi_i+\psi_k$
$\Leftrightarrow \psi_i = -\psi_k$. Moreover, $HCF(\psi_i,\psi_k)=+1$ or -1, thus $\Psi_a=(1,-1)$ or $(-1,1)$
Similar development leads to $\Psi_x=(1,0)$

#### 3.1.2.4 Problem representation

A representation set is associated to a problem defined in $\mathbf{R}^3$. Each representation defines a scheduling of elementary processings. The temporal order relation between the processing requires the introduction of a time parameter that evolves in parallel to the recurrency, since this relation is a total order on every recurrency processings associated to an elementary processing. We thus call spacetime, the space $ET \subset \mathbf{R}^3$. with orthonormal basis (i, j, t), where t represents the time axis.

**Definition 5.** A problem representation in ET is given by :
- the transformation matrix P from the processing domain canonical base to the spacetime basis
- the transformation vector V such as $V=\overrightarrow{O'O}$, where O is the origin of the frame associated to the canonical basis and O' is the origin of the spacetime frame

Point coordinates in spacetime can there for be expressed from coordinates in the canonical basis :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{ET} = P.\begin{pmatrix} i \\ j \\ k \end{pmatrix}_{BC} + V$$

$$\Leftrightarrow \begin{pmatrix} i \\ j \\ k \end{pmatrix}_{BC} = P^{-1}.\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{ET} .P^{-1}.V$$

This representation is given by the example of the Matrix Vector Product of Fig. 11.

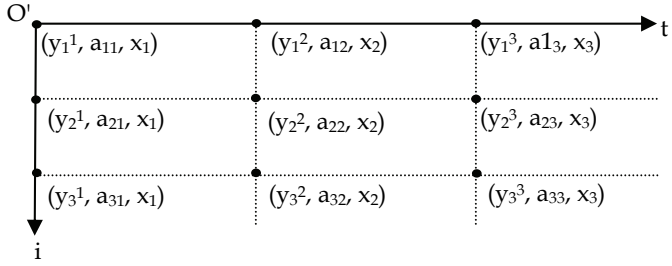



Fig. 11. Representation of the Matrix Vector Product in spacetime (t=k)

We call $R_0$ the initial representation of a problem, the one for which there is a coincidence between the canonical basis and the spacetime basis, i.e. P = I, I being the Identity Matrix, and V the null vector (O and O' are coinciding). For the MVP example, initial representation is given on Fig. 11.
These representations show the occurencies of a data at successive instants. Processings can be done in the same cell or on adjacent cells. In the first case, data makes a systolic network

made of functional cells in which the data can be put in the cell memory. In the second case, data circulate in the network from cell to cell.
The representation of the problem in spacetime defines a scheduling for the processing. To obtain networks with a different order, we apply transformations on the initial representation $R_0$. If, after a transformation, data are still processed simultaneously, a new transformation is applied until the creation of an optimal scheduling. From this representation a set of systolic networks is determined.
Applying a transformation to a representation consists in modifying the temporal abscissa of the points. Whatever the representation is, this transformation must not change the n-uple associated to the invariant points when order and simultaneity of processings is changed. The only possible transformations are thus those who move the points from the D domain in parallel to the temporal axis (O', t). For each given representation, $D_t$ is the set of points which have the same temporal abscisse, resulting in segments parallel to (O', i) in spacetime are obtained.
The transformation to be applied consists in deleting data occurencies simultaneities by forcing their successive and regular use in all the processings, which implies that the image of all lines $d_t$ by this transformation is also a line in the image representation. For instance, for the initial representation $R_0$ of the MVP, $D_t$ straight lines are dotted on Fig. 11. One can therefore see that occurrencies of data $x_k$, $0 \le k \le n-1$ are simultaneously used on each point of straight line $D_k$ with t = k. Therefore, a transformation can be applied to associate a non parallel straight line to the (O', i) axis to each $D_t$ parallel straight line to (O', i).
Two types of transformations can be distinguished leading to different image straight lines :
- $T_c$ for which the image straight line has a slope = +P (Fig. 12a)
- $T_d$ for which the image straight line has a slope = -P (Fig. 12b)

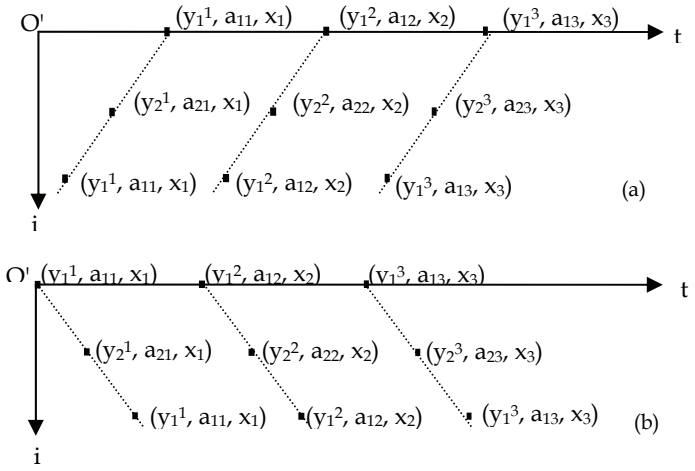



Fig. 12. Applying a transformation on the initial representation : (a) *Tc*, (b) *Td*

The application of a transformation enables to delete the occurencies use simultaneity of data, but increases the processing total execution time. For instance, for the initial representation of Fig. 11, the total execution time is t=n=3 time units, whereas for representations on Fig. 12, it is t=2.n-1 = 5 time units.

Concerning the initial representation, one can notice that 2 points of the straight line $D_t$ having the same temporal abscisse have 2 corresponding points on the image straight line which coordinates differ by 1. It means that two initially simultaneous processings became successive. After the first transformation, no simultaneity in data occurency use is seen, since all elementary processings on $D_t$ parallel to (O', i) use different data. Thus, no other transformation is applied. For the different representations, P (transformation matrices) as well as V (translation vectors) are :

$$P=\begin{pmatrix}1&0\\0&1\end{pmatrix} \text{ and } V=\begin{pmatrix}0\\0\end{pmatrix} \text{ for Fig. 12a.}$$

$$P=\begin{pmatrix}1&0\\1&1\end{pmatrix} \text{ and } V=\begin{pmatrix}0\\3\end{pmatrix} \text{ for Fig. 12b.}$$

#### 3.1.2.5 Determining systolic networks associated to a representation

For a given representation of a problem, the last step consists in determining what is/are the corresponding systolic network(s). The repartition of processings on each cell of the net has therefore to be carefully chosen depending on different constraints. An allocation direction has thus to be defined, as well as a vector with integer coordinates in $\mathbf{R}^3$, which direction determines the different processings that will be performed in a same cell at consecutive instants. In fact, the direction of allocations can not be chosen orthogonally to the time axis, since in this case, temporal axis of the different processings would be the same, which contradicts the definition.

Consider the problem representation of Fig. 12a. By choosing for instance an allocation direction $\xi=(1, 0)_{BC}$ or $\xi=(1, 1)_{ET}$ and projecting all the processings following this direction (Fig. 13), the result is the systolic network shown on Fig. 14. This network is made of $n$=3 cells, each performing 3 recurrency steps. The total execution time is therefore $2n$-1 = 5 time units. If an allocation direction colinear to the time axis is chosen, the network shown on Fig. 15 is then obtained.

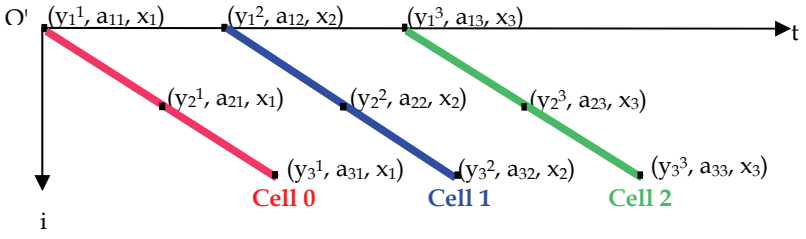



Fig. 13. Processings projection with $\xi=(1,1)_{ET}$

Other networks can be obtained by choosing another value for $D_t$ slope. The nature of the network cells depends on the chosen allocation direction.

Cappello and Steiglitz approach (Capello & Setiglitz, 1983) is close to Mongenet. It differs from the canonical representation obtained by associating a temporal representation indexed on the recurrency definition. Each index is associated to a dimension of the

geometrical space, and each point corresponds to a n-uple of indexes in which recurrency is defined.

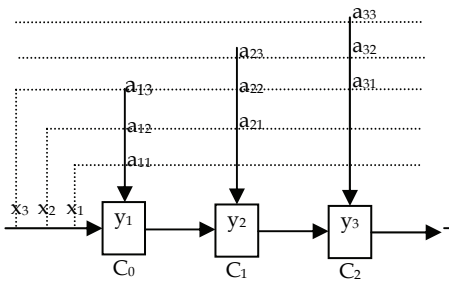



Fig. 14. Systolic network for $\xi=(1,1)_{ET}$

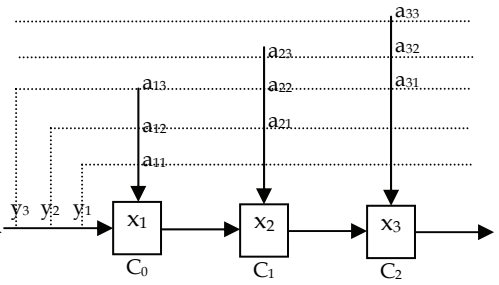



Fig. 15. Systolic network for $\xi=(0,1)_{ET}$

Basic processings are thus directly represented in the functional specifications of the architecture cells. The different geometrical representations and their corresponding architectures are then obtained by applying geometrical transformations to the initial representation.

### 3.2 Methods using sequential algorithms

Among all methods listed in (Quinton & Robert, 1991), we'll detail a bit more the Moldovan approach (Moldovan, 1982) that is based on a transformation of sequential algorithms in a high-level language.

The first step consists in deleting data diffusion in the algorithms by moving in series data to be diffused. Thus, for (n×n)-matrices product, the sequential algorithm is :

$$\forall i \mid 1\le i\le n, \forall j \mid 1\le j\le n, \forall k \mid 1\le k\le n, c_{new}(i,j)=c_{old}(i,j) + a(i,k).b(k,j) \qquad (9)$$

If one loop index on variables a, b and c is missing, data diffusion become obvious. When pipelining them, corresponding indexes are completed and artificial values are introduced so that each data has only one use. New algorithm then becomes :

$$\forall i \mid 1\le i\le n, \forall j \mid 1\le j\le n, \forall k \mid 1\le k\le n$$
$$a^{j+1}(i, k) = a^{j}(i, k)$$
$$b^{i+1}(k, j) = b^{i}(k, j)$$
$$c^{k+1}(i, j)= c^{k}(i, j)+ a^{j}(i, k).b^{i}(k, j)$$

The algorithm is thus characterized by the set $L^n$ of indexes of n overlapped loops. Here,

$$L^3 = \{ (k,i,j) \mid 1\le k\le n, 1\le i\le n, 1\le j\le n \}$$

which corresponds to the domain associated to the problem.

The second step consists in determining the set of dependency vectors for the algorithm. If an iteration step characterized by a n-uple of indexes $I(t) = \{i^1(t), i^2(t), ..., i^n(t)\} \in L^n$ uses a

data processed by an iteration step characterized by another n-uple of indexes $J(t)= \{ j^1(t), j^2(t), ..., j^n(t) \} \in L^n$, then a dependency vector $DE(t)$ associated to this data is defined :

$$DE(t) = J(t) - I(t)$$

Dependency vectors can be constant or depending of $L^n$ elements. Thus, for the previous algorithm, processed data $c^k(i,j)$ at the step defined by (i, j, k-1) is used at the step (i, j, k). This defines a first dependency vector $d_1=(i, j, k) - (i, j, k-1) = (0, 0, 1)$. In the same way, step (i, j, k) uses the $a^j(i, k)$ data processed at the step (i, j-1, k) as well as the $b^i(j, k)$ data processed at the step (i-1, j, k). The two other dependency vectors of the problem are therefore $de^2=(0,1,0)$ and $de^3=(1,0,0)$.
The next step consists in applying on the $<L^n, E>$ structure a monotonous and bijective transformation $T$ ($E$ is the order imposed by the dependency vectors), defined by :

$$T : <L^n, E> \rightarrow <L_T^n, E_T>$$

T is partitionned into :

$$\Pi : L^n \rightarrow L_T^k, k<n$$
$$S : L^n \rightarrow L_T^{n-k}$$

k gives the dimension of $\Pi$ and S. It is such as the function results in the order $E_T$. Thus, the k first coordinates of $\underline{I}$ and $L_T^n$ depend on time, whereas the following n-k coordinates are linked to the algorithm geometrical properties. For obtaining planar results, n-k must be less or equal than 2.
In the case that the algorithm made of $n$ loops is characterized by $n$ constant dependency vectors

$$DE = \{de_1, de_2, ... , de_n\}$$

the transformation T is chosen linear, i.e. $J = T . I$

If $v_i$ is the dependency vector $de_j$ after transformation, $V_i = T. DE_j$ , the system to solve is $T.DE = \Delta$ , $DE = \{ v_1, v_2, ..., v_m \}$. Necessary and sufficient conditions for existence of a valid transformation $T$ for such an algorithm are :
- $v_i = DE_i[c_j]$ , $c_j$ being the HCF of the $d_j$ elements
- $T.DE = \Delta$ has a solution
- The first non-zero element of $v_j$ is positive

Therefore, in our exemple of matrix product, dependency vectors are defined by :

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

A linear transformation T is such as $T = \Delta$. The first non-zero element of $v_j$ being positive, we consider $\Pi .d_i > 0$ and k =1 in order to size $\Pi$ and S, with :

$$T = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{pmatrix}$$

In this case, $\Pi.de_i = t_{1i} > 0$ . Thus, we choose for $t_{1i,}$ i=1, ..., 3, the lowest positive values, i.e. $t_{11} = t_{12} = t_{13} = 1$. S is determined by taking into account that T is bijective and with a matrix made of integers, i.e. Det(T) = ±1 . Among all possible solutions, we can choose :

$$T = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

This transformation of the indexes set enables to deduce a systolic network :
- Functions processed by the cells are deduced from the algorithm mathematical expressions. An algorithm similar to (9) contains instructions executed for each point of $L^n$. Cells are thus identical, except for the peripherical ones. When loop processings are too important, the loop is decomposed in several simple loops. The corresponding network therefore requires several different cells.
- The network geometry is deduced from function S. Identification number for each cell is given by $S(I) = ( j^{k+1}, ... , j^n )$ for $I \in L^n$. Interconnections between cells are deduced from the n-k last components of each dependency vector $v_j$ after being transformed :

$$v_j^s = S(I + DE_j) - S(I)$$

When T is linear :

$$v_j^s = S.DE_j$$

For each cell, $v_j^s$ vectors indicate the identification number of the cell for the variable associated to the vector. The network temporal processing is given by :

$$\Pi : L^n \rightarrow I_T^k$$

The elementary processing corresponding to $I \in L^n$ is performed at $t=\Pi(I)$. The communication time for a data flow associated to the dependency vector $DE_j$ is given by $\Pi(I+DE_j) - \Pi(I)$, which is reduced to $\Pi(DE_j)$ when T is linear.
Using the integer k for sizes of Π and S with the lowest possible value, the number of parallel operations is increased at the expense of cells number. Thus, when considering the matrix product defined with the following linear transformation :

$$T = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

S is defined by :

$$S\begin{pmatrix}k\\i\\j\end{pmatrix}=\begin{pmatrix}0&1&0\\0&0&1\end{pmatrix}.\begin{pmatrix}k\\i\\j\end{pmatrix}=\begin{pmatrix}i\\j\end{pmatrix}$$

The network is therefore a bidimensional squared network (Fig. 1c).
Data circulation are defined by S.$DE_j$. For the $c_{ij}$ data, dependency vector is

$$de_1=\begin{pmatrix}1\\0\\0\end{pmatrix}\Rightarrow S.de_1=\begin{pmatrix}0\\0\end{pmatrix}$$

Therefore, data remain in cells.
For the $a_{ik}$ data, dependency vector is :

$$de_2=\begin{pmatrix}0\\0\\1\end{pmatrix}\Rightarrow S.de_2=\begin{pmatrix}0\\1\end{pmatrix}$$

$a_{ik}$ circulate horizontally in the network from left to right.
Similarly, we can find :

$$S.de_3=\begin{pmatrix}1\\0\end{pmatrix}$$

and deduce that $b_{kj}$ circulate vertically in the network from top to bottom.

### 3.3 Fluency graphs description

In this method proposed by Leiserson and Saxe (Leiserson & Saxe, 1983), a circuit is formally defined as an oriented graph $G$ = ($V$, $U$) which summits represent the circuit functional elements. A particular summit represent the host structure so that the circuit can communicate with its environment. Each summit $v$ of $G$ has a weight $d(v)$ representing the related cell time cycle. Each arc $e$ = ($v$, $v'$) from U has an integer weight $w(e)$ which represents the number of registers that a data must cross to go from $v$ to $v'$.
Systolic circuits are those for which every arc has at least one related register and their synchroniszation can be done with a global clock, with a time cycle equal to Max($d(v)$).
The transformation which consists in removing a register on each arc entering a cell, and to add another on each arc going out of this cell does not change the behaviour of the cell concerning its neighborhood.
By the way, one can check that such transformations remain invariant the number of registers on very elementary circuit.
Consequently, a necessary condition for these transformations leading to a systolic circuit, is that on every elementary circuit of the initial graph, the number of registers is higher or equal to the number of arcs. Leiserson and Saxe also proved this condition is sufficient.
Systolic architecture condition is therefore made in 3 steps :

– defining a simple network $w$ in which results accumulate at every time signal along paths with no registers

- determining the lowest integer $k$. Thus, the resulting newtork $w_k$ obtained from $w$ by multiplying by k the weights of all arcs is systolizable. $w_k$ has the same external behaviour than $w$, with a speed divided by $k$.
- systolizing $w_k$ using the previous transformations

This methodology is interesting to define a systolic architecture from an architecture with combinatory logic propagating in cascade. Main drawback is that the resulting network often consists of cells activated one time per $k$ time signals. This means the parallelism is limited and execution time is lenghtened.
Other methods use these graphs :
- Gannon (Gannon, 1982) uses operator vectors to obtain a functional description of an algorithm. Global functional specificities are viewed as a fluency graph depending on used functions and operators properties, represented as a systolic architecture
- Kung (Kung, 1984) uses fluency graphs to represent an algorithm. The setting up of this method requires to choose the operational basic modules corresponding to the functional description of the architecture cells.

## 4. Method based on Petri Nets

In previously presented methods, the thought process can almost be always defined in three steps :
- rewriting of problem equations as uniform recurrent equations
- defining temporal functions specifying processings scheduling in function of data propagation speed
- defining systolic architectures by application of processings allocation functions to processors

To become free from these difficulties that may appear in complex cases and in the perspective of a method enabling automatic synthesis of systolic networks, a different approach has been developped from Architectural Petri Nets (Abellard et al., 2007) (Abellard & Abellard, 2008) with three phases :
- constitution of a Petri Net basic network depending on the processing to perform
- making of the Petri Net in a systolic shape (linear, orthogonal or hexagonal) defining data propagation

### 4.1 Architectural Petri Nets

To take into account sequential and parallel parts of an algorithm, an extention of Data Flow Petri Nets (DFPN) (Almhana, 1983) has been developped : Architectural Petri Nets (APN), using Data Flow and Control Flow Petri Nets in one model. In fact Petri Nets showed their efficiency to model and specify parallel processings and on various applications, including hardware/software codesign (Barreto et al., 2008) (Eles et al., 1996) (Gomes et al., 2005) (Maciel et al., 1999) and real-time embedded systems modeling and development (Cortés et al., 2003) (Huang & Liang, 2003) (Hsiung et al., 2004) (Sgroi et al., 1999). However, they may be insufficient to reach the implementation aim when available hardware is either limited in resources or not fully adequate to a particular problem. Hence, APN have been designed to limit the number of required hardware resources while taking advantage of the chip performances so that the importance of execution time lengthening may be non problematic

(Abellard, 2005). Their goal is on the one hand to model a complete algorithm, and on the other hand, to design the interface with the environment. Thus, in addition with operators used for various arithmetic and logic processing, other have been defined for the Composition and the Decomposition in parallel of data vectors.

### 4.1.1 Defactorized operators

#### 4.1.1.1 Compose

It proceeds to the ordered regrouping of d input data $T'_1$ to $T'_d$ of a same type into an output vector $[T'_1 ... T'_d]$ (Fig. 16).

$$Co(T'_1, \ldots, T'_d) \rightarrow [\, T'_1, \ldots, T'_d \,]$$

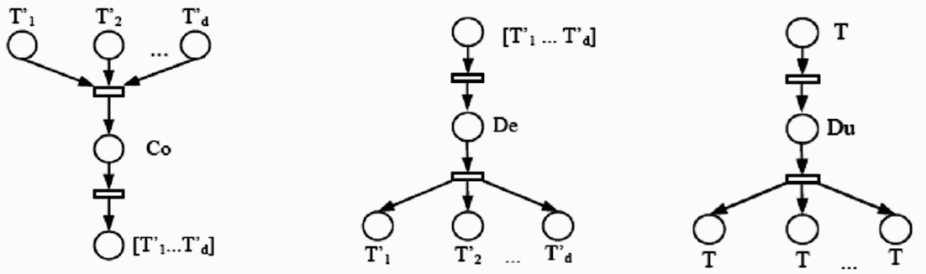



Fig. 16. Operators : Compose (left), Decompose (middle) and Duplicate (right)

#### 4.1.1.2 Decompose

It proceeds to the decomposition of a vector $[T'_1...T'_d]$ into its d elements $T'_1$ to $T'_d$ (Fig. 16).

$$De([T'1 ... T'd]) \rightarrow T'_1, \ldots, T'_d$$

#### 4.1.1.3 Duplicate

It proceeds to the duplication of input data to d subnets as in Data Flow Petri Nets, different operators can not use the same set of data (Fig. 16).

#### 4.1.1.4 Example of a Matrix Vector Product

An example of application of these operators is given on Fig. 17 with a MVP. One can easily see that the more important are the sizes of matrix and vector, the more important is the number of operators in the Net (and consequently the required hardware ressources).

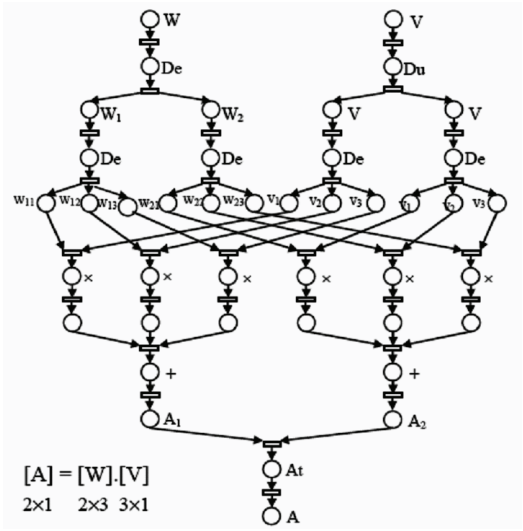



Fig. 17. Data Flow Petri Net of a MVP

The use of classic DFPN leads to an optimal solution as regards the execution time, thanks to an unlimited quantity of resources. However, a problem may appear. In fact, although these operations are simple taken separately, their combination may require relatively important amount of hardware resources, depending on the data type of the elements, and on the input matrix and vector sizes. We therefore have to optimize the number of cells prior to execution time. This is not a major drawback with a programmable component which has short execution times for real time controls. In order to limit as more as possible the resources quantity, we defined the Architectural Petri Nets (APN), that unify in a unique model Data Flow and Control Flow.

### 4.1.2 Factorization concept

The decomposition of an algorithm modelled with DFPN into a set of operations leads to the repetition of elementary identical operations on different data. So, it may be interesting to replace the repetitive operations by a unique equivalent subnet in which input data are enumerated and output data are sequentially produced. This leads us to define the concept of factorized operator which represents a set of identical operations processing differentsequential data.

Each factorized operator is associated to a factorization frontier splitting 2 zones : a slow one and a fast one. When the operations of slow zone are executed one time, those of the fast zone are executed n times during the same lapse of time.

**Definition 6.** A T-type element is represented by a vector of d1 elements, all of T'-type. Each T' type element may be also a vector of $d_2$ T''-type elements, and so on.

**Definition 7.** A Factorized Data Flow Petri Net (FDFPN) is a 2-uple (R, F) in which R is a DFPN and F a set of factorization frontiers F = {$FF_1$ , $FF_2$, ... $FF_n$}.

### 4.1.3 Factorized operators

The data enumeration needs to use a counter for each operator. An example is given on Fig. 18. Various factorized operators that are used in our descriptions are described in next sections.

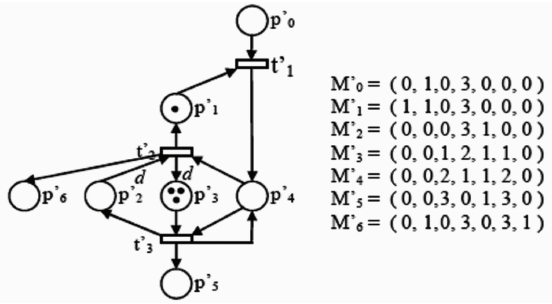



Fig. 18. Counter from 0 to n-1 (here n=3)

#### 4.1.3.1 Separate

It is identified by Se and it proceeds to the factorization of a Data Flow in an input vector form $[T'_1...T'_d]$ by enumerating the elements $T'_1$ to $T'_d$. A change of the input data value in the operator corresponds to d changes of the output data value. The Separate operator allows to go through a factorization frontier by increasing the data speed : the down speed of the input data of Separate is d times greater than the upper speed of output data. d output data (fast side) correspond to one input data (slow side) as the result of the input data elements enumeration synchronized with an internal counter (which sole $p'_0$ and $p'_6$ places are represented for graphic simplification).

Thus, a factorization frontier FF defined by a Separate operator dissociates the slow side from the fast side (Fig. 19a). A graphic simplified representation, where places coming from counter are not represented, is adopted on Fig. 19b. In a FDFPN, the operator Separate corresponds to the factorized equivalent of Decompose defined in 4.1.1.2.

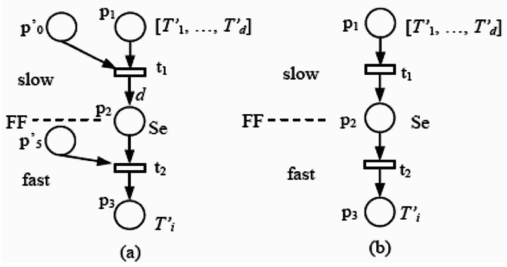



Fig. 19. Separate operator

#### 4.1.3.2 Attach

It is identified by At and it proceeds to the factorization of $d$ input data flows $T'_i$ by collecting them under an output vector form $[T'_1...T'_d]$ (Fig. 20a with $p'_0$ and $p'_6$ coming from the $d$-counter, and graphic simplified representation on Fig. 20b). d changes of input data

values in the Attach operator correspond to one change of the output data values. In a FDFPN, the operator Separate corresponds to the factorized equivalent of Compose defined in 4.1.1.1.

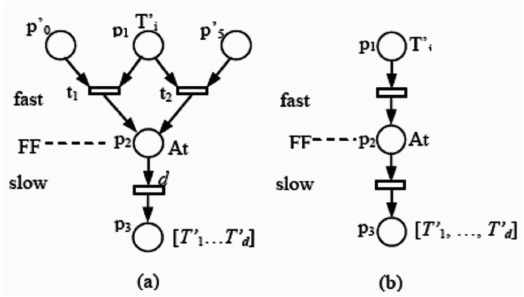



Fig. 20. Attach operator

#### 4.1.3.3 Iterate

It is identified by It and it proceeds to the iteration of a subnet which has s as input and e as output. The operator provides the specification of connexions between repetitive subnets, and appears in the FDFPN as a cycle through the "It" operator. On Fig. 21a, $p'_0$ and $p'_6$ come from the previously described *d*-counter, produced by a control operator which will be defined in section 4. (Fig. 21b being the simplified representation of the operator). in : initializing step ; fi : final step (counting completed)

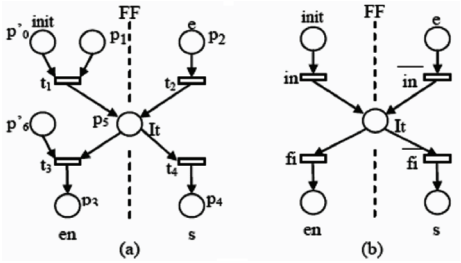



Fig. 21. Iterate operator

#### 4.1.3.4 Diffuse

This operator provides *d* times in output the repetition of an input data. Diffuse (Di) is a factorized equivalent to the Duplicate function defined in 3.2.3.3. (Fig. 22).

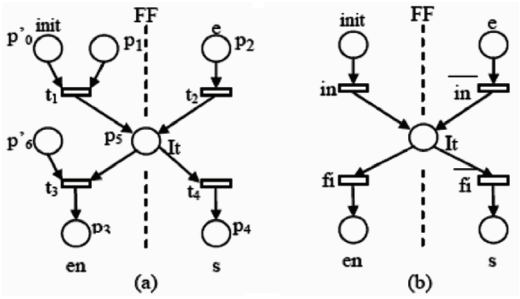



Fig. 22. Diffuse operator

#### 4.1.4 Example of a Matrix Vector Product

From the example of previous MVP, the corresponding FDFPN is given on Fig. 23a. Factorization enables to limit the number of operators in the architecture - and therefore the number of logic elements required – since data are processed sequentially. As for the validation places that enables to fire the net transitions, they come from a Control Flow Petri Nets (CFPN), which is described in the next paragraph (Fig. 23b).

Given the algorithm specification, i.e. the FDFPN, control generation of its implementation is deduced from data production and consumption relations, and neighborhood relation between all FF. Hence the generation of control signals equations that can be modelled with Petri Nets, by connecting control units related to each FF. Control synthesis of a hardware implementation consists in producing of validation and initialization signals for needed counters. Control generation of hardware implementation corresponding to the algorithm specification described by its FDFPN is thus modelled by CFPN.

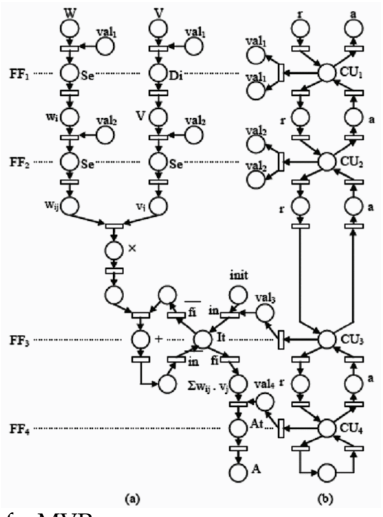



Fig. 23. FDFPN description of a MVP

### 4.1.5 Definition of Control Flow Petri Nets

A CFPN is a 3-tuple (*R*, *F*, *Pc*) in which : *R* is a 2-part places Petri Net, *F* is a set of factorization frontiers, *Pc* is a set of control places.

#### 4.1.5.1 Control synthesis

Five steps are necessary :
- Design of a FDFPN.
- Design of the PN representing neighborhood relations between frontiers.
- Definition of neighborhood, production and consumption relations using this Petri Net.
- Generation of signal control equations.
- Modelling using CFPN by connecting unit controls related to each FF.

#### 4.1.5.2 Control units

In a sequential circuit containing registers, each FF has relations on its both sides (slow and fast). Relations between request and acknowledgment signals, up and down, for both slow and fast sides, provide the design of the control unit. It is composed of a d-counter and additional logic which generate communication protocols, cpt (counter value) and val (validation signal) for transitions firing.

Functions rules : If the control unit (CU) receives aupper request (*ur*= 1) and the down acknowledge is finished (*da*=0), it validates the data transfer (*ua*=1) and sends a request to the next operator (*dr*=1) (Fig. 24). If a new request is presented while da is not yet activated, then CU does not validate a new data transfer which is left pending. CU controls bidirectional data flow.

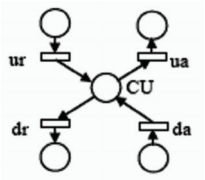



Fig. 24. Control Unit representation

### 4.2 Example of the Matrix Product

Once these operators have been defined, they can now be used in the Petri Net description of a systolic array, as it is developped in the following example. Be *C* = *A*.*B* a processing to perform, with *A*, *B* and *C* squared matrixes of the same size (n=2 to simplify). Processings to perform are :

$$c_{i,j}=\mathrm{Sum}(a_{i,k}.b_{k,j})_{k=1...2} \tag{10}$$

which require eight operators for multiplication and to propagate $a_{ik}$, $b_{kj}$ and $c_{ij}$ (Fig. 25).

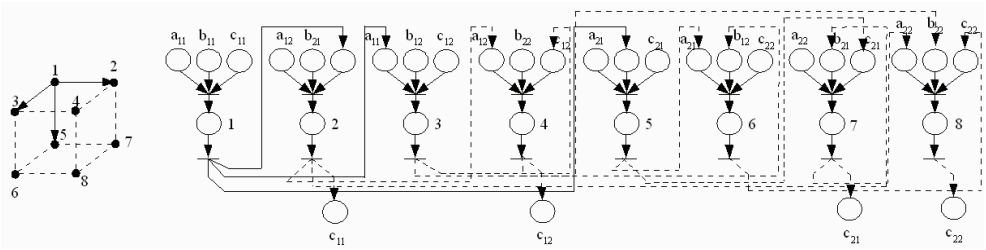


Fig. 25. First step of data propagation

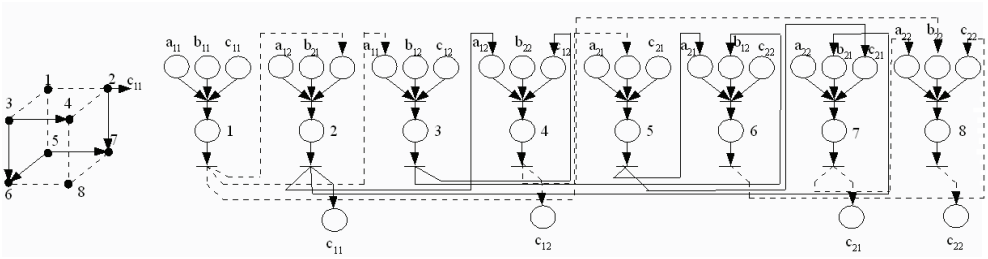


Fig. 26. Second step of data propagation

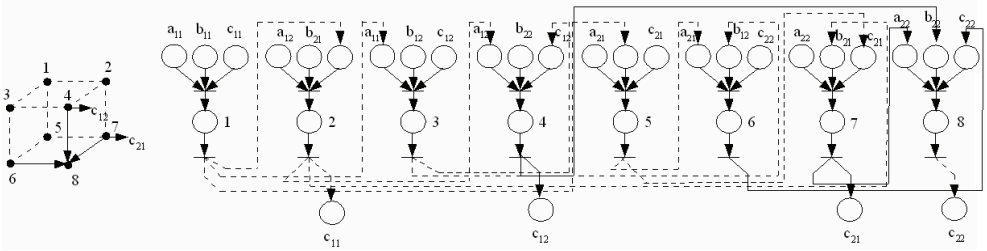


Fig. 27. Third step of data propagation

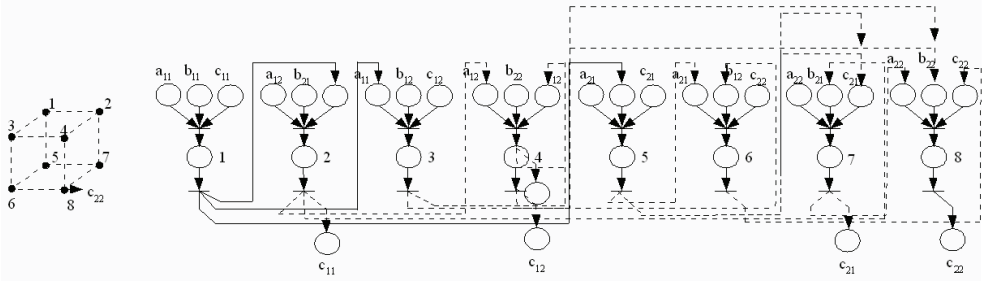


Fig. 28. Fourth step of data propagation

In the first step (Fig. 25), operator 1 receives $a_{11}$, $b_{11}$ and $c_{11}$. It performs $c_{11}=a_{11}.b_{11}$ and propagates the three data to operators 3, 5 and 2. In the second step (Fig. 26), operator 2 receives $a_{12}$ et $b_{21}$, operator 3 receives $b_{12}$ and $c_{12}$ and operator 5 receives $a_{21}$ and $c_{21}$. Operator 2 performs : $c_{11} = a_{11}.b_{11} + a_{12}.b_{21}$. Operator 3 performs $a_{11}.b_{12}$ and operator 5 processes

$a_{21}.b_{11}$. These operators are respectively connected to operators 4 and 7 on the one hand, 6 and 7 on the other hand.
In the third step (Fig. 27), operator 4 receives $b_{22}$, operator 6 receives $c_{22}$ and operator 7 receives $a_{22}$. These 3 operators are linked to operator 8. They perform : $c_{12} = a_{11}.b_{12} + a_{12}.b_{22}$ and $c_{21} = a_{21}.b_{11} + a_{22}.b_{21}$. In the final step (Fig. 28), operator 8 performs $c_{22} = a_{21}.b_{12} + a_{22}.b_{22}$.
By propagating data in the 3 directions, the processing domain becomes totally defined :

$$D = \{(i,j,k) \mid 1 \le i \le N, 1 \le j \le N , 1 \le k \le N \}$$

Classic projections are :
- $\Pi = (1,1,0)$ or $(1,0,1)$ or $(0,1,1)$ which results in the linear network in Fig. 1a.
- $\Pi = (0,0,1)$ or $(0,1,0)$ or $(1,0,0)$ which results in the squared network in Fig. 1b.
- $\Pi = (1,1,1)$ which results in the hexagonal network in Fig. 1c.

For example, with the first solution, the result is as in Fig. 1. Each cell is made of a multiplier/adder with accumulation (Fig. 29).

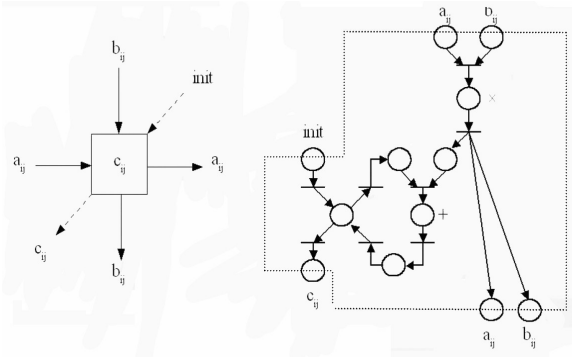
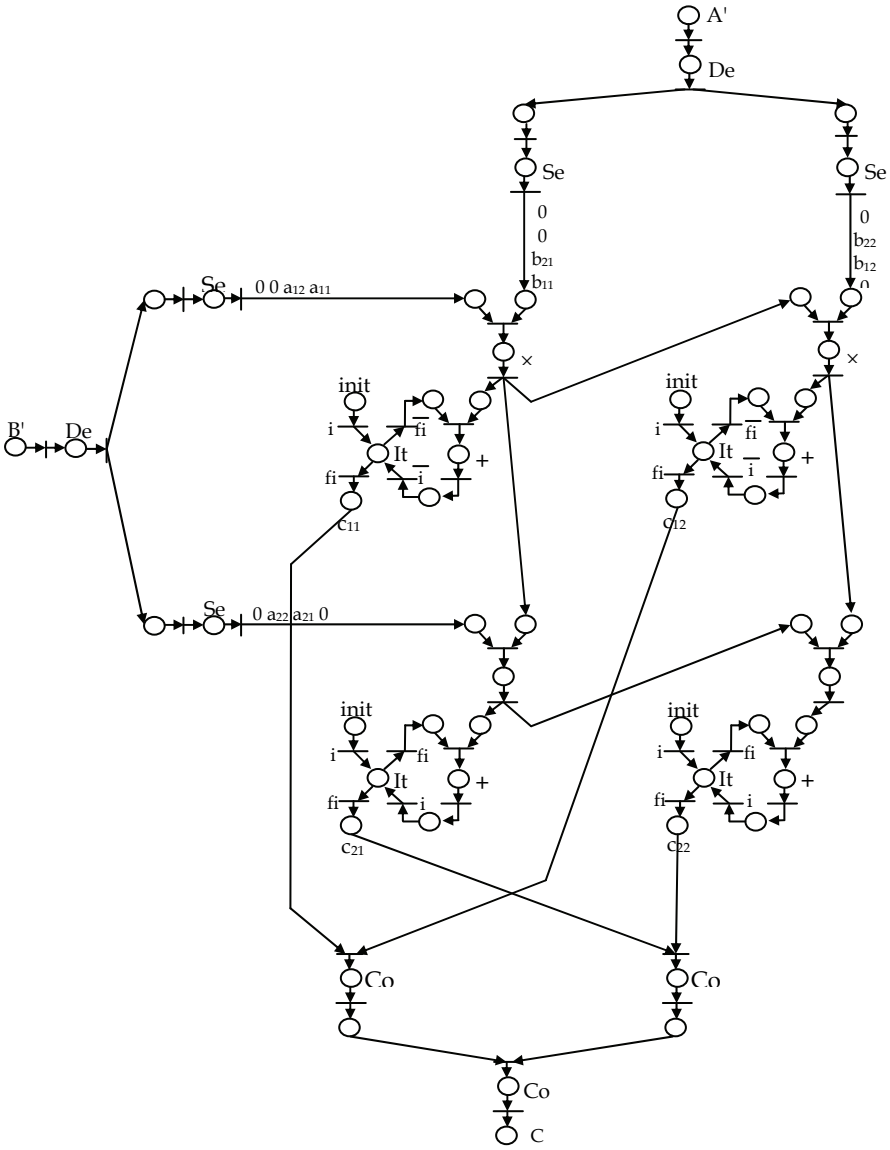



Fig. 29. Squared network of matrix product $C=A.B$

The Architectural Petri Net defining the complete systolic network is obtained by adding Decompose and Compose operators in input and output so as to perform the interface with the environment (Fig. 30). In order to be free from the related hardware problems that can occur to retrieve results in the cells, the hexagonal structure can also be used. In this type of network, a, b and c circulate in 3 directions (Fig. 31). For instance, with a 3×3 matrix product, the network operating cycle is as following :
1 - Network is reset. $a_{11}$, $b_{11}$ and $c_{11}$ come in input respectively of operators $o_5$, $o_9$ and $o_1$.
2 - $a_{11}$, $b_{11}$ and $c_{11}$ are propagated to $o_{15}$, $o_{17}$ and $o_{13}$.
3 - $a_{11}$, $b_{11}$ and $c_{11}$ come as input of $o_{19}$ in which $c_{11} = a_{11}.b_{11}$ is done. $a_{12}$, $a_{21}$, $b_{12}$, $b_{21}$, $c_{12}$ and $c_{21}$ come in input respectively of operators $o_4$, $o_6$, $o_8$, $o_{10}$, $o_2$ and $o_{12}$.
4 - $c_{11}$, $a_{12}$ and $b_{21}$ come as input of $o_6$ at the same time. $c_{11} = a_{11}.b_{11}+a_{12}.b_{21}$ is done. Other data are propagated.
5 - $c_{11}$, $a_{13}$ and $b_{31}$ come as input of $o_7$ at the same time. $c_{11}= a_{11}.b_{11}+a_{12}.b_{21}+a_{13}.b_{31}$ Other data are propagated.
Processings are done similarly for other terms until the matrix product has been completed.




$$A=\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \rightarrow A'=\begin{pmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{21} & a_{22} & 0 \end{pmatrix} \quad B=\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \rightarrow B'=\begin{pmatrix} b_{11} & 0 \\ b_{21} & b_{12} \\ 0 & b_{22} \\ 0 & 0 \end{pmatrix} \quad C=\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$$

Fig. 30. Petri Net of the systolic network for the matrix product

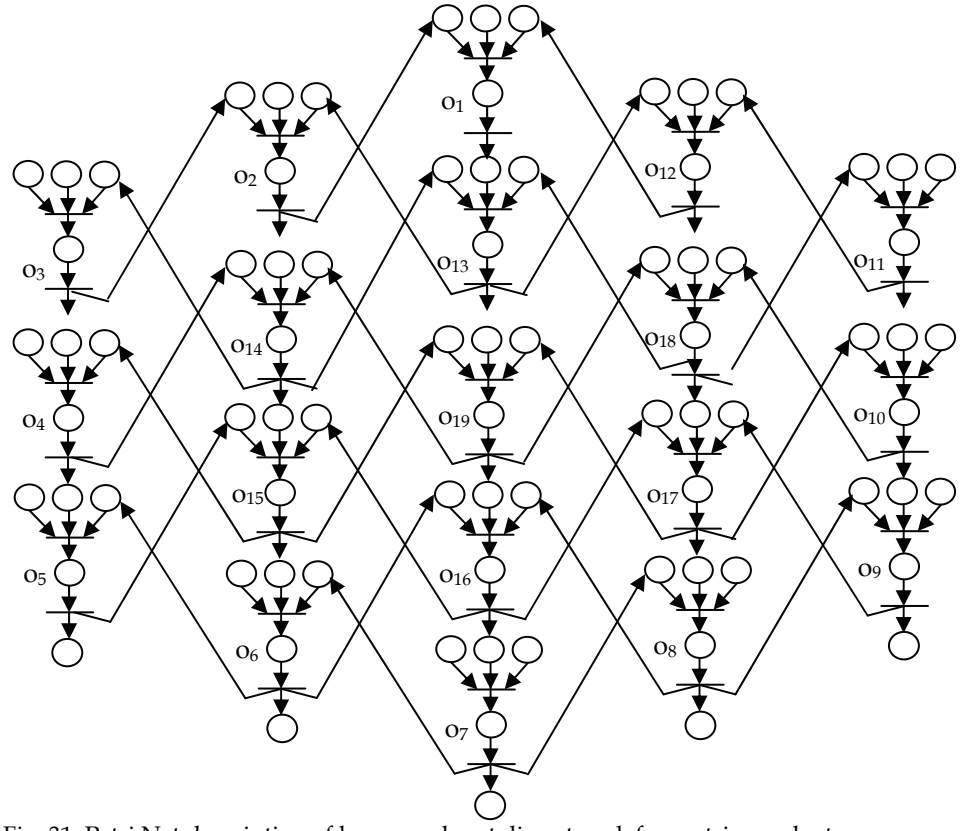



Fig. 31. Petri Net description of hexagonal systolic network for matrix product

## 5. Conclusion

The main characteristics of currently available integrated circuits give the possibility to make massively parallel systems, as long as the processings « volume » are given priority to data transfer. Systolic model is a powerful tool for conceiving specialized networks, using identical elementary cells locally interconnected. Each cell receives data coming from neighbourhing cells, performs a simple processing, then transmits the results to neighbourhing cells after a time cycle. Only cells on the network frontier communicate with the environment. Their conception is often based on methods using recurrent equations, or on sequential algorithms or fluency graphs. It can be efficiently developped thanks to a tool completely formalized, lying on a strong mathematical basis, i.e. Petri Nets, and their Architectural extension. Moreover, this model enables to do their synthesis and to ease their implementation on reprogrammable components.

## 6. References

Abellard, A., (2005). Architectural Petri Nets : Basics concepts, methodology and examples of application, *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pp. 2037-2042, Waikoloa, HI, USA, October 2005, IEEE.

Abellard, A.; Abellard, P. & Gorce, P. (2007). Architectural Petri Nets : Basics concepts, methodology and examples of application, *Proceedings of IEEE/ASME AIM International Conference on Advanced Intelligent Mechatronics*, Zurich, Switzerland, September 2007, IEEE.

Abellard, A. & Abellard, P. (2008). A Design Methodology of Systolic Architectures Based on a Petri Net Extension. Application to a Stereovision Hardware/Software Processing Improvement, *Proceedings of ICSEA - International Conference on Software Engineering Advances*, pp. 77-82, Sliema, Malta, October 2008, IARIA.

Almhana, J. (1983). *Modélisation par réseaux de Petri à flux de données. Application à la synthèse de l'opérateur de Riccati rapide*. PhD Thesis, Université d'Aix-Marseille III, France.

Barreto, R. ; Maciel, P. ; Tavares, E. ; Oliveira, M. & Lima R. (2008), A time Petri Net-based method for embedded hard real-time software synthesis, *Design Automation for Embedded Systems*, Vol. 12, pp. 31-62, ISSN 0929-5585 (Print) 1572-8080 (Online), Springer.

Blume, H. ; von Sydow, T. & Noll, T.G. (2006), A case study for the application of deterministic and stochastic Petri Nets in the SoC communication domain, *Journal of VLSI Signal Processing*, Vol. 43, pp. 223-233, ISSN 0922-5773, Springer.

Cortés, L.A. ; Eles, P. & Peng, Z. (2003), Modeling and formal verification of embedded systems based on a Petri net representation, *Journal of Systems Architecture*, Vol. 49, pp. 571–598, ISSN 1383-7621, Elsevier.

Eles, P. ; Kuchcinski, K. & Peng, Z. (1996), Synthesis of systems specified as interacting VHDL processes, *Integration-The VLSI Journal*, Vol. 21, No. 1-2, pp. 113-138, ISSN 0167-9260, Elsevier.

Gomes, L. ; Barros, J.P. & Costa, A. (2005), Structuring Mechanisms in Petri Net Models: From specification to FPGA based implementations. In: Adamski, M. ; Karatkevich, A. & Wegrzyn, M. (Eds.), *Design of embedded control systems*, pp. 153-166, ISBN 978-0-387-23630-8, Springer.

Ghavami, B. & Pedram H. (2009), High performance asynchronous design flow using a novel static performance analysis method, *Computers and Electrical Engineering*, in press, Elsevier.

Hsiung, P.A. & Gau, C.H. (2002), Formal synthesis of real-time embedded software by time-memory scheduling of colored time Petri Nets, *Electronic Notes in Theoretical Computer Science*, Vol. 65, No. 6, pp. 140-159, Elsevier.

Hsiung, P.A. ; Lin, C.Y. & Lee, T.Y. (2004), Quasi-dynamic scheduling for the synthesis of real-time embedded software with local and global deadlines, *Lecture Notes in Computer Science*, Vol. 2968, pp. 229–243, ISBN 3-540-21974-9, Springer-Verlag.

Huang, C.C. & Liang W.Y. (2003), Object-oriented development of the embedded system based on Petri-nets, *Computer Standards & Interfaces*, Vol. 26, pp. 187–203, Elsevier.

Maciel, P. ; Barros, E. & Rosenstiel, W. (1999), A Petri Net model for hardware/software codesign, *Design Automation for Embedded Systems*, Vol. 4, No. 4, pp. 243-310, Springer.

Oliveira, M. ; Maciel, P. ; Barreto, S. & Carvalho, F. (2004), Towards a software power cost analysis framework using colored Petri Net, *Lecture Notes in Computer Science*, pp. 362–371, ISBN 3-540-23095-5, Springer-Verlag.

Sgroi, M. ; Lavagno, L. ; Watanabe, Y. & Sangiovanni-Vincentelli, A. (1999) Synthesis of embedded software Using free-choice Petri Nets, *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*, pp. 805-810, ISBN 1-58133-109-7, New Orleans, LA, USA, June 1999, IEEE.

Strbac, P. ; Tuba, M. & Simian, D. (2009) Hierarchical model of a systolic array for solving differential equations implemented as an upgraded Petri Net, *WSEAS Transactions on Systems*, Vol. 8, No. 1, pp. 12-21, WSEAS.

Capello, P.R. & Steiglitz, K. (1983). Unifying VLSI array designs with geometric transformations, *Proceedings of International Conference on Parallel Processing*, pp. 448-457, Bellaire, USA.

Castro-Pareja, C.R ; Sagadeesh, J.M. : Venugopal, R. & Shekha, R. (2004), FPGA based 3D median filtering using word-parallel systolic arrays, *Proceedings of IEEE ISCAS International Symposium on Circuits and Systems*, Vol. 3, pp. 157-160, Vancouver, Canada, May 2004, IEEE.

Gannon, D. (1982). Pipelining array computation for MIMD parallelism. *Proceedings of International Conference on Parallel Processing*, Columbus, OH, USA, 1982.

Jackson, P.A. ; Chan, C.P. ; Scalera, J.E. ; Rader, C.M. & Vai, M.M. (2004). A Systolic FFT Architecture for Real Time FPGA Systems, *Proceedings of HPEC - Eighth Annual Workshop on High Performance Embedded Computing,* Lexington, MA, USA, 2004.

Johnson, K.T. & Hurson, A.R. (1993). General purpose systolic arrays. *Computer*, Vol.26, No.1, pp. 20-31.

Kung, H.T. (1982). Why systolic architectures ? *Computer*, Vol.15, pp.37-46.

Kung, H.T. (1988), Systolic communications, *Proceedings of International Symposium on Computer Architectures,* San Diego, CA, USA, 1988, IEEE.

Kung, S.Y. (1984), On supercomputing with systolic/wavefront array processors, *Proceedings of the IEEE,* Vol.72, pp. 867-884.

Lee, J.J. & Song, G.Y (2002), Implementation of the Systolic Array for Dynamic Programming, *Proceedings of ICITA International Conference on Information Technology and Applications,* ISBN 1-86467-114-9, Bathurst, Australia, 2002, IEEE.

Lee, J.J. & Song, G.Y (2003), Implementation of the super systolic array for convolution, *Proceedings of ASP-DAC Asia and South Pacific Design Automation Conference,* pp. 491-494, ISBN 0-7803-7659-5, Kitakyushu, Japan, January 2003, IEEE.

Lee, J.J. & Song, G.Y (2004), Implementation of a bit-level super systolic FIR filter, *Proceedings of IEEE AP-ASIC - Asia Pacific Conference on Advanced Systems Integrated Circuit,* pp. 206-209, ISBN 0-7803-8637-X, Fukuoka, Japan, August 2004, IEEE.

Leiserson, C.E. & Saxe, J.B (1983), Optimizing synchronous circuitry by retiming, *Proceedings of 3D CalTech Conference on VLSI,* pp. 87-116, 1983.

Li, G.J. & Wah, B.W (1984), The design of optimal systolic arrays, *IEEE Transactions on Computers,* Vol.33, No.10, 1984.

Lim, H. & Swartzlander, E.E (1996a), Multidimensional systolic arrays for multidimensional DFTs, *Proceedings of the IEEE International Conference on Acoustic, Speech and Signal Processing,* Vol.6, pp. 3276-3279, Atlanta, GA, USA, May 1996, IEEE.

Lim, H. & Swartzlander, E.E (1996b), Efficient systolic arrays for FFT algorithms, *Proceedings of the 29th Asilomar Conf. Signals, Systems and Computers,* Vol.1, pp. 141-145, ISBN 0-8186-7646-9, Pacific Grove, CA, USA, 1996, IEEE.

Lim, H. & Swartzlander, E.E (1999), Multidimensional systolic arrays for the implementation of Discrete Fourier Transform, *IEEE Transactions on Signal Processing,* Vol.47, No.5, pp. 1359-1370.

Mihu, I.Z. ; Brad, R. & Breazu, M. (2001), Specifications and FPGA implementation of a systolic Hopfield-type associative memory, *Proceedings of the International Conference on Neural Networks,* Vol.1, pp. 228-233, Washington, DC, USA, 2001.

Moldovan, D.I. (1982), On the analysis of synthesis VLSI algorithms, *IEEE Transactions on Computer,* Vol.31, No.11, pp. 1121-1126.

Mongenet, C. (1985). *Une méthode de conception d'algorithmes systoliques, résultats théoriques et réalisation*. PhD Thesis, Université de Nancy, France.

Nash, J.G. (2002), Automatic latency optimal design of FPGA based systolic arrays, *Proceedings of the 10th IEEE Symposium on Field Programmable Custom Computing Machines*, pp. 299-300, Napa, CA, USA.

Nash, J.G. (2005), Computationally Efficient Systolic Architecture for Computing the Discrete Fourier Transform, *IEEE Transactions on Signal Processing*, pp. 4640-4651, Vol. 53, No. 12, ISSN 1053587X.

Quinton, P. (1983). The systematic design of systolic arrays, *Report IRISA 193*, Rennes, France.

Quinton, P. & Robert, Y. (1991). *Systolic algorithms & architectures*, Ed. Prentice Hall, ISBN 0138807906, London.

Sousa, L.A. (1998), Bidirectional systolic arrays for digital recursive filters, *Proceedings of the International Conference on Electronics, Circuits and Systems*, Vol.3, pp. 451-502, ISBN 0-7803-5008-1, Lisboa, Portugal, September 1998, IEEE.

Yang, Y. ; Zhao, W. & Inoue, Y. (2005), High performance systolic arrays for band matrix multiplication, *Proceedings of the IEEE International Symposium on Circuits and Systems*, Vol.2, pp. 1130-1133, ISBN 0-7803-8834-8, Kobe, Japan, May 2005, IEEE.

6

# Towards Rewriting Semantics of Software Architecture Specification

Yujian Fu
*Department of Computer Science*
*Alabama A&M University*
*yujian.fu@aamu.edu*
Zhijiang Dong
*Department of Computer Science*
*Middle Tennessee State University*
*zdong@mtsu.edu*
Phil Bording
*Department of Computer Science*
*Alabama A&M University*
*phil.bording@aamu.edu*
Xudong He
*School of Computer Science*
*Florida International University*
*hex@cis.fiu.edu*

## 1. Introduction

During the past decade, architectural design has emerged as an important subfield of software engineering. This is because a good architecture can help ensure that a system will satisfy user requirements. Consequently, a new discipline emerged, which concerns formal notations for representing and analyzing architectural designs using Architecture Description Language (ADL) [25]. These notations provide both a conceptual framework and a concrete syntax for characterizing software architectures [25]. Combine with software architecture models that are not considered as ADLs, we call them software architecture specifications.

Software architecture specifications (i.e. software architecture model, software architecture description languages (ADLs) (such as Rapide [24], Wright [1] and XM-ADL [23]), etc.) allow software designers to focus on high level aspects of an application by abstraction of the details of the subsystems and components. It is precise and accurate to use formal methods to describe the abstraction that makes software architecture specifications are suitable for verification using model checking techniques. Software specifications are, in a way, domain-specific languages for aspects such as coordination and distribution. Software Architecture Model (SAM) is a formal approach based on two formal languages - Petri nets

and temporal logic for distributed concurrent software systems. Further, SAM has been used to interpret semantics of Unified Modeling Language (UML) diagrams [9, 10].

The theory of rewriting logic (RWL) has proved to be a unifying formal framework to describe concurrency formalisms [26, 27]. It is useful to specify concurrent behavior of various types of systems. In the verification aspect, the issue of using rewriting logic in the model checking group is how to convert a logic notations based on Boolean results to a state based labeled transition system. As a high performance and reflective language, the Maude specification [6] was developed to support rewriting logic based model checking on concurrent systems. Maude is a high performance declarative programming language that is based on rewriting and equational theory. Equational theory is used to describe the properties and rewriting logic is used to describe the state transition of systems.

Major concurrency formalisms have been successfully translated in rewrite theories [27]. In addition, rewriting logic has been used by several authors in the verification of architectural notations such as architectural description languages and object-oriented design formalisms. However, there are few works in the formal description of software architecture specifications using rewriting logic.

Current research on software architectures has a focus on how to express and verify functional and nonfunctional aspects statically and dynamically. It is highly desirable to precisely describe the semantics of software architecture specification. Some ADLs do not provide formal semantics [28] while many do [1, 24]. One category provides benefits for easy to understand and use, but hardly to be reasoned and analyzed in the architecture level. The other category, in contrast, has benefits of precise reasoning and verification. Analysis software architecture specification using rewriting logic aims at formally reasoning distributed concurrent systems at the architecture level.

This helps to reduce errors that are introduced to implementation during development process [17, 18].

Recent work has been done [13, 9, 12, 8] to verify the system model in both the design and implementation level with an integrated framework, which combines the formal verification (model checking) with implementation verification technique (runtime checking). However, there is no work on the logic based semantics analysis of software architecture specification – SAM. This chapter is to present a systematic translation algorithm as well as a validation approach towards rewriting logic semantics of software architecture model.

Related Works. Several related works are investigated in the software architecture specification and rewriting logic semantics. The paper in [7] presents a framework for describing global optimizations by rewrite rules with Computation Tree Logic (CTL) formulae as side conditions, which allows the generation of correct optimizations, but cannot be used for verification of (possibly incorrect) optimizations. The correctness is established in an imperative language without procedures. The work in [15] proposes a method for deploying optimizing code generation while correct translation between input program and code. They focus on code selection and instruction scheduling for SIMD machines.

Xie et al. [31] presented an approach to transforming software specification syntax to the model checking programming language. In [31], static analysis is used to validate the syntax, semantics and property translations, on which software testing strategy is adopted to validate the translation. Preserving equivalence conditions was not checked in that work. The work in [22] describes a translation from textual transition system to Petri nets. Using

bisimulation, they validated the translation by proof of the translated model are equivalent with the source model based on the step by step comparison. Although it is a stepwise proof between translated model and the source model, the soundness and completeness cannot be established because it is based on the semi-formal explanation for the comparison by the natural language.

Several works have been done for the analysis of SAM specification either using theorem proving [21] or model checking technique [20]. All these methods are based on the either partial order reduction [20] or encoded symbolic states [21]. This chapter presents for the first time an approach to convert Software Architecture Model (SAM) [30] to rewriting logic based semantics. This work has an interesting property of actually executing the SAM semantics to do the simulations, that is, rewriting a topology, and the verification, since the transformation from SAM to rewriting logic makes rewriting logic an alternative executable semantics of SAM.

The remainder of this paper is organized as follows. In Section 2, we review SAM with predicate transition nets for high-level design and rewriting logic with the tool support – Maude's syntax. A translation algorithm is presented in Section 3. Then coffee machine case study was presented in Section 4. After that, a validation approach is demonstrated in Section 5. Finally, we draw conclusions and describe future work in Section 6.

## 2. Preliminaries

### 2.1 SAM – Software Architecture Model

SAM (Software Architecture Model) [30] is hierarchically defined as follows. A set of compositions $C = \{C1, C2, \ldots, Ck\}$ represents different design levels or subsystems. A set of component $Cm_i$ and connectors $Cn_i$ are specified within each composition $Ci$ as well as a set of composition constraints $Cs_i$ , e.g. $Ci = \{Cm_i , Cn_i , Cs_i \}$. In addition, each component or connector is composed of two elements, a behavioral model and a property specification, e.g. $Cij = (Sij, Bij)$. Each behavioral model is described by a Petri net, while a property specification by a temporal logical formula. The atomic proposition used in the first order temporal logic formula is the ports of each component or connector. Thus each behavioral model can be connected with its property specification. A component $Cm_i$ or a connector $Cn_i$ can be refined to a low level composition $Cl$ by a mapping relation $h$, e.g. $h(Cm_i)$ or $h(Cm_i) = Cl$. SAM is suitable to describe large scale systems' description. However, there is no high level behavior definition of components and connectors. In our work, we only consider the flattened version of SAM specification in which each component/connector cannot be further refined.

#### 2.1.1 Predicate Transition Nets Predicate

Transition (PrT) net [14] is a high level Petri net. A Predicate/Transition net is a 9-tuple$(P, T, F, \sum, Eq, \varphi, L, G, M0)$, where:

1. P is a finite set of places, T is a finite set of transitions ($P \cap T = \Phi$, $P \cup T \neq \Phi$), and F is a set of arcs or flow relations between each pair of P and T, e.g. $F \subseteq (P \times T) \cup (T \times P)$. The tuple (P, T, F) forms a basic Petri net structure.
2. $\sum =< St, Op >$ consists of some sorts ($St$) of constants together with set of operations ($Op$) and relations on the sorts.
3. $Eq$ defines the meanings and properties of operations in $OP$.

4. $\varphi : P \rightarrow St$ is a relation associated each place $p$ in $P$ with a subset of sorts.
5. L is a labelling function on places, transitions, arcs, and variables. Given a place $p \in P$ or a transition $t \in T$, $L(p)$ returns the name of place $p$, $L(t)$ returns the name of transition $t$. Given an arc $f \in F$, the labelling function of $f$, L($f$), is a set of labels associated with the arc $f$, which are tuples of constants (*CONs*) and variables ($X$), which is best described by L($f$, $Terms_{\Sigma,X}$). We use $Terms_{\Sigma,X}$ represents the expressions on the label of arc $f$. We use L($Terms_{\Sigma,X}$) represents L($f$, $Terms_{\Sigma,X}$) when there is no confusion in context. If $f < F$, $L(f) = \Phi$.
6. $R$ is a mapping from transitions to a set of inscription formulae. The inscription on transition $t \in T$, $R(t)$, is a logical formula built from variables and the constants, operations, and relations in structure $\Phi$, variables occurring free in a formula have to occur at an adjacent input arc of the transition.
7. $M0$ is the initial or current marking with respect to sort, which assigns a multi-set of tokens to each place $p$ in $P$ with the same sort, $M0 : P \rightarrow MCONs$.

The dynamic semantics of PrT nets are defined by the transition firings. Dynamic semantics of PrT nets are described as follows [19]:

1. A marking $M$ of a PrT net $N$ is a mapping function defined from set of places $P$ to constants *MCONs*.
2. The enabling condition of a transition $t \in T$ under a marking $M$ with a substitution $\alpha = \{x_i \leftarrow ci | xi \in X, ci \in MCONs\}$ is defined as follows:
$$\forall p \in P.\ (L(f) : \alpha) \subseteq M(p) \wedge R(t) : \alpha.$$
3. If a transition $t \in T$ under a marking $M$ with a substitution $\alpha$ is enabled, a marking $M0$ is obtained after the transition $t$ is fired, then the firing condition of a transition $t$ is defined as:

$\forall p \in P.\ M0(p) = M(p) - L(p, t) : \alpha \cup L(t, p) : \alpha.$

The enabling and firing condition can also be defined by preset and postset of transition. A preset of a transition $t$, denoted by $pre(t)$ or $\bullet t$, is the set of all places that have outgoing relation from these places to the transition $t$.

If a transition is enabled, required tokens specified by the label expression of input arcs of the transition must be available in the preset of the transition. If a transition is fired, those required tokens are consumed and produce some tokens that satisfy the label expression of output arcs of the transition. Both consumed tokens and produced tokens must have the same sort of the preset of the transition and postset of the transition respectively. Let $L(f, Terms_{\Sigma,X})$ be label expression that associates with arc $f = (p, t) \in F \vee f = (t, p) \in F$. For any place $p \in P$, we can define the two functions for the consumed tokens (*consuming-token-p*($L((p, t), Terms_{\Sigma,X})$)) and produced tokens (*producing-token-q*($L(f, Terms_{\Sigma,X})$)) for a transition $t$ as follows.

$$\forall p \in P.consuming-token-p(L((p,t),Terms_{\Sigma,X})) : p \rightarrow L((p,t),Terms_{\Sigma,X}) : \alpha \text{ where } L(f,Terms_{\Sigma,X}) : \alpha \in \varphi(p), (p,t) \in F, \text{ and } M'(p) = M(p) - consuming-token-p(L((p,t),Terms_{\Sigma,X})),$$

$$\forall q \in P.producing-token-q(L(f,Terms_{\Sigma,X})) : p \rightarrow L((p,t),Terms_{\Sigma,X}) : \alpha \text{ where } L(f,Terms_{\Sigma,X}) : \alpha \in \varphi(p), (p,t) \in F, \text{ and } M'(p) = M(p) \cup producing-token-q(L((p,t),Terms_{\Sigma,X})).$$

From above functions, we can see that for any transition $t$, the tokens consumed in the preset of the transition $t$ can be described by a substitution of label expression in the function *consuming- token-p*($L(f, Terms_{\Sigma,X})$), if the substitution of label expression in the token set of preset of transition $t$; while the tokens produced in the postset of the transition $t$ can be described by a substitution of label expression in the function *producing-token-q*($L(f)$), if the substitution of label expression satisfy the sort of postset of the transition $t$. The token set of

preset and postset of transition $t$ can be described by the substitution of sorts of preset and postset, i.e., $pre(t)(\varphi(p) : \alpha)$ and $post(t)(\varphi(p) : \alpha)$. Therefore, we have enabling and firing conditions as follows:

$$Consuming\text{-}token\text{-}p(L(f, Terms_{\Sigma,X})) \in pre(t)(\varphi(p) : \alpha), \text{ and}$$

$$Producing\text{-}token\text{-}q(L(f, Terms_{\Sigma,X})) \in post(t)(\varphi(p) : \alpha).$$

We can define the interleaving semantics using the sequence of markings with the occurrence of corresponding transitions for each set of substitutions.
A sequence $\sigma = M0[t0{=}\alpha 0 > M1[t1{=}\alpha 1 > \ldots [tn\text{-}1/\alpha n\text{-}1 > Mn$ with $n \geq 0$ is called a finite interleaving execution starting with $M0$ iff $\forall i \in Nat$ and $0 \leq i \leq n$ and $Mi\text{-}1 \rightarrow_{ti\text{-}1/\alpha i\text{-}1} Mi$, where $Mi : P \rightarrow \wp(St)$, $\alpha i$ denotes a substitution for the variables in a guard condition of a transition $ti$, $St$ denotes set of sorts.

### 2.1.2 Temporal Logic

Temporal logic defines four future-time (past-time) operators in addition to the propositional logic operators. They are:

- Always in the future (past), symbolized as a box $\Box$ ($\boxdot$).
- Sometime in the future (past), symbolized as a diamond $\Diamond$ ($\diamondsuit$).
- Until for the future (Since for the past), $U$ ($S$ ).
- Next (Previous) for the future (past) , O($\odot$).

An example of a temporal logic formula $\Box(p \rightarrow \Diamond q)$ indicates that predicate $p$ implies eventually $q$ always happen.

## 2.2. Rewriting Logic

Rewriting logic [26] is a logic for concurrency. A rewrite theory R is a tuple $(\Sigma, E, L, R)$, where $(\Sigma, E)$ is an equational logic, $L$ is set of labels, and $R$ is a set of rewrite rules. A rewrite $P : M \rightarrow N$ means that the term $M$ rewrites to the term $N$ modulo $E$R, and this rewrite is witnessed by the proof term $P$. Apart from general (concurrent) rewrites $P : M \rightarrow N$ that are generated from identity and atomic rewrites by parallel and sequential composition, rewriting logic classifies its most basic rewrites as follows: a one-step (concurrent) rewrite is generated by parallel composition from identity and atomic rewrites and contains at least one atomic rewrite, and a one-step sequential rewrite is a one-step rewrite containing exactly one atomic rewrite.
We often write .

$$l : [s] \rightarrow [t] \text{ if } [\overrightarrow{u}] \rightarrow [\overrightarrow{v}] \text{ for } l : [s] \rightarrow [t] \text{ if } [u_1] \rightarrow [v_1] \wedge \ldots \wedge [u_k] \rightarrow [v_k]$$

For the unconditional rewrite, if clause is weaved and the form would be l : [s] → [t]. Where s and t are terms that may contain variables. A rule describes a local concurrent transition in a system, i.e., anywhere where a substitution instant $\alpha(s)$ of the lefthand side $s$ is found, a local transition of that state fragment to the new local state $\alpha(t)$ can take place.
Rewriting logic therefore extends equational logic with rewrite rules, allowing one to derive both equations and rewrites. Deduction remains the same for equations, but the symmetry rule is dropped for rewrite rules. Rewriting logic is a framework for true concurrency [27]:

the locality of rules allows multiple rules to apply at the same time provided they don't modify the shared part.
The operational semantics of rewrite specification extends the operational semantics of membership equational specification by applying computational equations $E$R and rewrite rules $R$R modulo the structural equations $E$R. The process net for the behavior model – Petri nets – can be matched into the rewrite theory $R$. For instance, signatures and rewrite rules can be mapped into the net specification and places, and transitions with guard conditions and their pre- and post-sets. Instead of using operational semantics, the work [26] demonstrated the isomorphism between category Petri nets and category Rewrite theory.
Rewriting logic and membership logics are supported by Maude [3]. In Maude [2] the basic units are functional modules and system modules. A functional module is an equational style functional program with user-definable syntax in which a number of sorts, their elements, and functions on those sorts are defined. A system module is a declarative style concurrent program with user-definable syntax.
A functional module is declared in Maude using the keywords

```
fmod <ModuleName> is
        <DeclarationsAndStatements>
endfm
```

The <DeclarationsAndStatements> includes signatures (e.g. sorts, subsorts, kinds etc.), operations, and equations. In Maude, functional modules are equational theories in membership equational logic satisfying some additional requirements. Computation in a functional module is accomplished by using the equations as rewrite rules until a canonical form is found. This is the reason why the equations must satisfy the additional requirements of being Church-Rosser, terminating, and sort decreasing.
A system module is declared in Maude using the keywords

```
mod <ModuleName> is
        <DeclarationsAndStatements>
endm
```

The <DeclarationsAndStatements> includes sorts and subsorts, operation, equation, rules, etc.. declaration.
Conditional rules has the form of

```
crl [label] : <left term> => <right term> if <condition or set of conditions> .
```

While unconditional rules has the form of

```
rl [label] : <left term> => <right term> .
```

The system modules specify the initial model T of a rewrite theory R in the membership equational logic variant of rewriting logic. These initial models capture nicely the intuitive idea of "rewrite systems" in the sense that they are transition systems whose states are equivalence classes $[t]$ of ground terms modulo the equations $E$ in R, and whose transitions are proofs $\alpha : [t] \rightarrow [t0]$ in rewriting logic, that is concurrent rewriting computations in the system described by the rules in $R$.

## 3. Translation from SAM Specification to Rewriting Logic

```
*** This is the algorithm to translate the SAM model to Maude program

*** M–SIGNATURE
for all  B ∈ C_k, where k ∈ Nat
    for all  P ∈ B, St ∈ B
        declare sort s,    s_i ∈ St where i ∈ Nat
        sort Marking .
        for all   p ∈ P and s_i ∈ St
        *** declare an operation
        f_p : s_i → Marking .

*** M–SYSTEM
    including M–SIGNATURE
    for all   B ∈ C_k, where k ∈ Nat
    for all   T ∈ B and G ∈ B,
        rl [t] : f_1 → f_2
            where f_1 is the set of operations mapped from . t,
                  f_2 is the set of operations mapped from t..
        crl[t] : f_1 → f_2 if g
            where g is associated w t, and g ∈ G and t ∈ T

*** M–PREDICATE
    including M–SYSTEM .
    including SATISFACTION .
    sort Marking < State .
    for all   B ∈ C_k, where k ∈ Nat
        for all  P ∈ B, St ∈ B
        ***declare an operation
        Pf_p : s_i → Prop .
        m_1, m_2 : Marking .
        *** define semantics using equations
        eq m_1  f_p m_2 ⊨ Pf_p = true .
            where i, j ∈ Nat

*** M–MODEL–CHECK
    including M–SYSTEM .
    declare initial markings
    declare properties to be checked .
```

Fig. 1. Translation Algorithm from SAM to Maude

The SAM specification allows formal verification of a component system against system constraints and property specified on its abstraction. Here, verification means that the developer can animate the specification by providing initial markings and checking if the responses meet the expected results. Verification of SAM is based on the precise syntax and semantics of Petri net formal language and temporal logic.

In this work, we choose Maude [2] – a high level and high performance declarative programming language – as a model checking tool. Maude is based on rewriting logic and membership equational logic. During the verification, we found a seamless matching

between SAM specification and rewriting logic. We use *Nat* to denote the set of natural numbers. The translation algorithm is summarized in Fig. 1. The translation covers the topology and dynamic behavior of the SAM specification.

In Fig. 1, *B* denotes the set of behavior models, *Ck* denotes the set of components, connectors or compositions; *St*, *P*, *T*, and *R* the set of sorts, places, transitions, and constraints associated with transition in the behavior model *B*, respectively; each element of the set *St*, *P*, *T*, and *G* is represented by lower case letters *si*, *p*, *t*, *g* respectively. Following the rewriting logic notation in [2], each operation is denoted by *f* or *fi* where $i \in Nat$. We use *fpre* to denote the left side operations and *fpost* denote right side operations.

SAM specification is a hierarchical structure that provides the high level organization of the system topology. When no behavior refinement defined, the components and connectors are taken off so that a flattened architecture description with one behavior model (Petri net) is obtained. Petri nets of all components/connectors form an integrated Petri net through merging connected ports between components/connectors. This flattened SAM structure simplifies the analysis and verification. Therefore, the translation algorithm presented in Fig. 1 is based on the flattened version of SAM specification. In this section, we first present the translation algorithm and the corresponding Maude program in two groups – signature translation and system description translation. Comments in Maude syntax are started with three stars and ended at the end of the line.

In Maude syntax, there are two fundamental blocks defined – function module and system module. Where functional module in Maude denotes the signature of the system, system module describes the state changing under certain set of rules (conditions). The purpose of the translation algorithm is to map each building block in SAM specification to a corresponding one in Maude specification.

In Fig. 1, the signature of the system is described in a function module named *M-SIGNATURE* where *M* represents the system. To describe the system, three system modules, named *M-SYSTEM*, *M-PREDICATE*, and *M-CHECKING* must be defined. *M-SYSTEM* describes the system topology with structure and interactions. In this module we describe the state transition of the system using rewriting rules and the equations. *M-PREDICATE* describes the semantics of the operations and state transitions. *M-CHECKING* is a verification module of the system. This module specifies the initial conditions that are required for the model execution and properties to be checked. We summarize the translation algorithm in Fig. 1.

### 3.1 Mapping to Signature

Signature (sorts, operations) and equations of the behavior model (*B*) of SAM – Petri nets – can be mapped to the _-signature in the rewriting theory. In addition, each sort of a place and a port is considered as a local state, and defined as a sort *Marking*. The translated sorts in the rewriting logic include the set of sorts in Petri nets *St* as well as a new sort *Marking*.

Maude currently supports limited basic data types (int, bool, String) and operations on them in predefined module. The module for the basic sorts and operations can be loaded when the core Maude is running. To implement all user defined sorts in Petri nets, we need following two restrictions: a) specifying them as Cartisian product using the predefined sorts in Maude; and b) trying to specify the system using the basic sorts and avoid the complicated user defined sorts. We do not define any more equations in the signature translation. The equations are those used for the basic sorts defined in rewriting and membership equational logic for basic types.

### 3.2 Mapping to System Module

In the behavior model Petri nets, the state changings are described by marking updating through transition firing. Each time a transition $t$ is fired, some tokens consumed in the preset ($pre(t)$) of the transition $t$, new tokens are generated to the postset ($post(t)$). This state changing can be described by a(n) (un)conditional rule *rl* in the rewriting theory R.

Let $id(t)$ denote the name of transition $t$. A transition $t \in T$ with its preset and postset of places can be mapped to an unconditional rule with following format:

$rl[id(t)] : fpre(t) \rightarrow fpost(t)$ if guard $g \in G$ associated with this transition is *true* or empty.

Otherwise, the transition $t \in T$ is mapped to a conditional rule

$$crl[id(t)] : fpre(t) \rightarrow fpost(t) \text{ iff } g \text{ for } g \in G.$$

This translation is shown in Fig. 1 from line 11 to 17.

In Maude program, we can express the above translation as follows:

```
including M-SIGNATURE .

vars vi : si .

rl [id] : <operations of pre(transition)> => <operations of post(transition)> .

crl [id] : <operations of pre(transition)> => <operations of post(transition)> if <guard of
transition> . ,
```

where *id* is the identifier of the transition, $pre(t)$ and $post(t)$ denote the preset and postset of transition $t$.

### 3.3 Mapping to State Predicates

After translating the signature and system model, to do model checking, the things left is for the state predicates and initial markings. Maude provides the conversion from system model to the Kripke Structure with internal modules based on rewriting theory. To specify state predicates what we need is to associate the kind of sorts of the system to the formulae in the Kripke structure. In Maude, the predefined module *LTL* converts the Linear Temporal Logic (LTL) formulae to the Kripke structure [2]. Module *SATISFACTION* associates the state to formula by declaring a sort *State* and a boolean operation on the sort *State*.

Sort *Marking* is defined as a subsort of predefined sort *State*. Based on the pre-exiting module definitions of Maude, we can have following translation. Each state predicate is defined as an operator of sort *Prop*. Then defines their semantics by means of a set of equations that specify for what states a given state predicate evaluates to true. This is also expressed the algorithm Fig. 1 in line 18 – 29. We have following code template in Maude:

```
protecting M-SIGNATURE .

including SATISFACTION .

subsort Marking < State .

vars m1 m2 : Marking .

op Pp : si -> Prop .
```

eq m1 Pp m2 |= Pid = true . ,

where *si* denotes the set of sorts of the place *p*, .

### 3.4 Mapping to Initial Markings & Properties

Finally, to define the initial marking and the properties to be verified (Fig. 1), upon previous translation algebra, we can simply define initial markings as equations of operations to initial conditions, and properties are equations from property operations to properties. Implementing in Maude, we have following template:

In Maude, the predefined module *MODEL-CHECKER* convert the system model defined by the rewriting logic to the Kripke structure so that we can apply model checking on the system [2]. The module *LTL-SIMPLIFIER* is used to define the linear temporal logic in rewriting logic. Based on these two modules, we define the following template Maude program:

```
protecting M-PREDICATE .

including MODEL-CHECKER .

including LTL-SIMPLIFIER .

*** declare initial markings as operations that output sort Marking.

op initn : -> Marking .

*** declare operation for each property that output sort Formula.

op propertyn : -> Formula .

*** declare equations for each place that has initial markings.

eq init = pi .

*** declare needed variables.

eq propertyn = <propertyn> .
```

### 3.5 Discussion

The ports synchronization is a challenging issue. The semantics of each pair of merged ports are described as interface places that are shared by component and connector in SAM. Although ports are visible for communicated components/connectors, only when tokens are produced in the outgoing ports, the incoming ports of corresponding components/connectors can be executed. In the translation (Fig. 1) for the shared places of ports, the outgoing ports are translated to right side of a rule, while the corresponding incoming ones are the left side of the corresponding rule. On the other side, if the overlap rewriting rules can fire concurrently, the executions of the concurrent rule is nondeterministic.

This translation algorithm (Fig. 1) is applied on a case study – coffee machine and experimented using Maude program. The experimental result is demonstrated in Section 4.

## 4. Case Study

Fig. 2 shows the SAM specification of a simple coffee machine [29], which accepts requests and then either serves a cup of coffee or returns back money. Behavior models of all three

components *CoinHandler*, *BrewingFacility*, and *CMInter f ace* are demonstrated in Appendix A (Fig. 3(a)-Fig. 3(c)).

A request with a tuple of money and coffee type is accepted in the component *CMInter f ace*. A simplification of this model is that we assume money and coffee type are input at the same time instead of modelling input sequentially. Then the request flows into component *CoinHandler* through connector *CH-CMI*. After checking the price table kept in the *CoinHandler*, either a money return or a coffee request is issued through the corresponding port *coin_back_ch* or *coffee_request_ch*, respectively. The place *price* (Fig. 3(a)) outputs a token either to transition *enough* or *not enough*, which is used to check whether or not the input money is enough for a certain type of coffee. If there is enough money, coffee request is issued by port *coffee_request_ch*, and *coin_back_ch* gets a token when the money is more than that kept in the place *price*. If there is not enough money, *coin_back_ch* will eventually get a token to return the money.

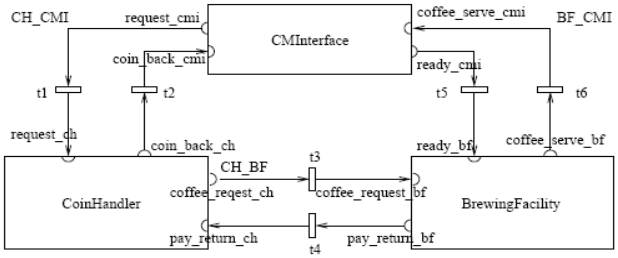



Fig. 2. SAM specification of Coffee Machine

The component *BrewingFacility* (Fig. 3(b)) keeps a coffee storage table and is responsible for cooking and providing coffee to *CMInterface*. *BrewingFacility* (Fig. 3(b)) serves coffee whenever two conditions are satisfied – a token with *true* is obtained in place *ready_bf*, and a token flows into place *coffee_request_bf* when requested coffee is not out of stock in place *storage*. Place *storage* keeps a table to count how many cups of coffee the machine has currently. If there is enough coffee, a user receives a cup of coffee in the *CMInterface*. If there is not enough coffee, then an exception is issued to *CoinHandler* through place *pay_return_bf*.

### 4.1 Translated Maude Code

In this section, we use a simple example to illustrate the experimental results of the translation algorithm as follows.

#### 4.1.1 Mapping to Signature

Based on the above algorithm, we first translate the coffee machine example to obtain the function module. We consider one component *CoinHandler* as the example here. In the SAM specification, we define ports *request_ch* and *coin_back_ch* in the component *CMHandler*. The sorts of the port *request_ch* is defined as < *int*, *int* >, while the sorts of the port *coin_back_ch* is < *int* >. After the translation, we have following Maude program segment:

```
fmod CM-SIGNATURE is
protecting INT .
```

```
sort Marking .
op request-ch : Int Int -> Marking .
op coin-back-ch : Int -> Marking .
*** other operations are ignored.
endfm
```

### 4.1.2 Translation of System

In the coffee machine example, port *request_cmi*(*money*, *type*) is translated to an operation on sorts *int* and *Marking*. We want the data in this port go to the port *request_ch*(*money*, *type*) without condition. This behavior is controlled by the transition *enough* specified in the Petri net, and specified in the rewriting logic as the following formula:

$r1$ : *request_cmi*(*money*, *type*) $\rightarrow$ *request_ch*(*money*, *type*) if $money \geq cost$,

where operations *request_cmi*(*money*, *type*) and *request_ch*(*money*, *type*) are defined as follows:

*request_cmi*(*money*, *type*) : $int \times int \rightarrow Marking$ and

*request_ch*(*money*, *type*) : $int \times int \rightarrow Marking$.

In Maude, we have the following rule for this state changing:

```
mod CM is
including CM-SIGNATURE .
vars money type cost : Int .
crl [enough] : request-ch(money, type) price(type,cost) => coffee-request-ch(type)
save(money, type)
price(type,cost) if money >= cost .
*** other rules are ignored.
```

### 4.1.3 Mapping to State Predicates

After we have the translated function module and system module, we can define the state predicates with their semantics. In the Maude program, concerning the translation of the port *request_ch*(*money*, *type*) and *coin_back_ch*(*money*), we can have the following program segment by applying the translation algorithm:

```
mod CM-PREDS is
protecting CM .
including SATISFACTION .
subsort Marking < State .
***(CMHandler)
op Prequest-ch : Int Int -> Prop .
```

```
op Pcoin-back-ch : Int -> Prop .
vars money type cost capacity : Int .
vars m1 m2 : Marking .
eq m1 request-ch(money,type) m2 |= Prequest-ch(money,type) = true .
eq m1 coin-back-ch(money) m2 |= Pcoin-back-ch(money) = true .
*** others are ignored.
```

#### 4.1.4 Mapping to Initial Marking & Property

Following is the segment of translated model checking module:

```
mod CM-CHECK is
including CM-PREDS .
including MODEL-CHECKER .
including LTL-SIMPLIFIER .
vars money type : Int .
op init1 : -> Marking .
eq init1 = request-ch(75,2) price(1, 50) storage(1, 50) .
op property1 : -> Formula .
eq property1 = [] (Prequest-ch(money,type) -> <>(Pcoffee-request-ch(type) /\ Pcoin-back-
ch(money))) .
*** others properties are ignored
*** due to space limitation.
```

### 4.2 Discussion

We have examined several properties for this example, the running results evaluated from translate Maude program follow the expected results of the SAM model. This algorithm had been applied on several other examples such as cruise controller [5] and online shopping example [11]. All the experimental results demonstrated its efficiency and portability.

## 5. Translation Validation

In our translation, we simply flattened the SAM specification so that all compositions are dereferenced and all components and connectors are visualized by a Petri net. Petri nets of dereferenced compositions are connected together through merging shared ports. Then the whole system connects to one Petri net. This is shown in the M-SIGNATURE and M-SYSTEM module of the translation algorithm (Fig. 1). We translate the flattened SAM specification, i.e. its Petri nets into the rewriting logic. Therefore, we consider an individual flattened component/connector in the translation validation. To validate the translation, we

first build a Petri net based on the rewriting logic. The semantics of the rewriting based Petri net can be expressed by a sequence of rewriting steps. Then we define the correspondence results between the operational semantics of Petri nets and the logic one, lifting it to the proved semantics.

### 5.1 Rewriting Theories and Deduction

The mapping of a Petri net model into a rewriting theory requires fairly sophisticated algebraic techniques. In the following we first recall the rewriting logic and membership equational logic [16].

#### 5.1.1 Basic

Definition An equational logic is a pair $(\Sigma, E)$, where $\Sigma$ is a set of operations, also called its "syntax", and $E$ is a set of equations of the form $\forall X.t = t'$ constraining the syntax, where $X$ is some set of variables and $t$, $t'$ are well-formed terms over variable set $X$ and operations in $\Sigma$. Equational logics can be many-sorted (operations in $\Sigma$ have arguments of specific sorts), or even order-sorted (sorts with a partial order on them). Equations can be conditional or unconditional. Conditions is a typically finite set of pairs $u = u'$ over set of variables $X$.

Term rewriting is an approach related to the equational logic $(\Sigma, E)$ in which equations are oriented from left to right, with the form $\forall X.p \rightarrow q \text{ if } \wedge_i u_i \rightarrow u'_i$, and called rewrite rules. A rewrite rule can be applied to a term $t$ at any position where $p$ is matched. A pair $(\Sigma, R)$, where $R$ is a set of rewrite rules, is called a rewrite system.

Rewriting logic [26] is a logic for concurrency. A rewrite theory is a tuple $(\Sigma, E, L, R)$, where $(\Sigma, E)$ is an equational logic, $L$ is set of labels, and $R$ is a set of rewrite rules. Rewriting logic therefore extend equational logic with rewrite rules, allowing one to derive both equations and rewrites. Deduction remains the same for equations, but the symmetry rule is dropped for rewrite rules. Rewriting logic is a framework for true concurrency [27]: the locality of rules allows multiple rules to apply at the same time provided they don't modify the shared part. A formal definition of a labeled rewrite theory is given in the following.

**Definition 1** An many-sorted labeled rewrite theory R is a 4-tuple $< \Sigma, E, L, R >$ where $\Sigma$ is an many-sorted signature, E is a set of $\Sigma$-equations, L is the set of labels, and $R \subseteq L \times T_{\Sigma,E}(X)^2$ is the set of labeled rewrite rules.

We often write.

$$l : [s] \rightarrow [t] \text{ if } [\vec{u}] \rightarrow [\vec{v}] \text{ for } l : [s] \rightarrow [t] \text{ if } [u_1] \rightarrow [v_1] \wedge \ldots \wedge [u_k] \rightarrow [v_k].$$

Rewrite rules in $R$ may be understood as the basic *rewriting steps* of a theory, the building blocks of the actual rewrite relation. More complex deductions can be obtained by a finite number of applications of inference rules. We introduce a suitable signature for building an algebra of labels, each element of the term algebra encoding a justification of a rewrite. Out of all possible different ways to introduce such a signature, we follow the lines of [4].

**Definition 2 (rewriting step)** Let $R =< \Sigma, E, L, R >$ be a rewrite theory, let  be the signature containing all the labels r as suitable operators, with the corresponding arity and sort given by the variables in R(r) A proof term $\alpha$ is a term of the algebra $T_R(X) = T_{_[}(X)$ (We assume that there are no clashes of names between the sets of operators).

A rewriting step is a triple < α, [s], [t] > (usually written as α : [s] → [t]) where α is a proof term and [s], [t] 2 TΣ,E(X).

| | |
|---|---|
| (reflexivity) | $\frac{[t] \in T_{\Sigma,E}(X)}{t : [t] \rightarrow [t]}$ |
| (transitivity) | $\frac{\alpha : [s] \rightarrow [u], \beta : [u] \rightarrow [t]}{\alpha;\beta : [s] \rightarrow [t]}$ |
| (congruence) | $\frac{f \in \Sigma_n, \alpha_i : [s_i] \rightarrow [t_i]}{f(\alpha_1, ..., \alpha_n) : [f(s_1, ..., s_n)] \rightarrow [f(t_1, ..., t_n)]}$ |
| (replacement) | $\frac{\alpha_i : [w_i] \rightarrow [z_i], R \vdash r : [s] \rightarrow [t]}{r(\alpha_1, ..., \alpha_n, \alpha) : [s(\overrightarrow{w}/\overrightarrow{x})] \rightarrow [t(\overrightarrow{z}/\overrightarrow{x})]}$ |
| (equality) | $\frac{E \vdash s = u, \alpha' : u \rightarrow u', E \vdash u' = t}{\alpha : s \rightarrow t}$ |
| (composition) | $\frac{\alpha : [s] \rightarrow [u], \beta : [u] \rightarrow [t]}{\alpha;\beta : [s] \rightarrow [t]}$ |
| (associativity) | $\frac{-}{\alpha;(\beta;\gamma) \equiv (\alpha;\beta);\gamma}$ |
| (distributivity) | $\frac{f \in \Sigma}{f(\alpha_1;\beta_1, ..., \alpha_n;\beta_n) \equiv f(\alpha_1, ..., \alpha_n);f(\beta_1, ..., \beta_n)}$ |

Table 1. Inference Rules

As argued in [26], "a rewrite theory is just a static description of 'what a system can do'; the behavior of the theory is instead given by the rewrite relation induced by the set of rules of deduction". Given a set of rewriting rules, we can derive a series of rewriting theorems about the system. This procedure is called *entailment* and defined in the next definition.

**Definition 3 (Entailment)** Let R =< ∑, E, L, R > be a rewrite theory. We say that R entails the rewriting step α : [s] → [t], written as R ⊢ : [s] → [t], if and only if it can be obtained by a finite number of applications of the inference rules in Table 1.

### 5.2 A Theory for Petri nets

In Section 3, to describe local state represented by a place, we introduce a new sort *Marking* to rewriting logic. The sort *Marking* lifts the local state on sort description and relates to global state described by marking in Petri nets. Therefore, we use *Marking* as a sort of ∑ in the rewriting theory of Petri nets in the following.

Given a PrT net $N = (P, T, F, \Sigma, Eq, \varphi, L, G, M0)$ [19], we define a rewriting theory R$_{PrT}$ = < ∑R, $E$R, $L$R, $R$R > for each ingredient of PrT nets as follows. If there is no confusion in the context, we refer R simply as R$_{PrT}$ . A rewriting theory of Petri nets can be defined as:

- $S\mathrm{R} = St \cup \{Marking\}$ for $St \in \Sigma$ and $Op\mathrm{R} = Op \cup Opp$, where $Opp = \{\varphi(p) \rightarrow Marking\}$ and $p \in P$. Then we have signature $_\mathrm{R} = (S\mathrm{R}, Op\mathrm{R})$.
- $E\mathrm{R} = Eq$.
- $L\mathrm{R}$ is set of labels, which is defined by a function such that $L\mathrm{R} : P \cup T \rightarrow f\Sigma \cup R\mathrm{R}$, where $f\Sigma$ denotes the operators on the signature $\Sigma$, if no confusion, we simply refer $f$ as $f\Sigma$.
  - $\forall p \in P.\ L\mathrm{R}(p) = f\Sigma$, where $f$ is defined as a mapping from sorts of $p$ to the sort *marking*, i.e., $f : s \rightarrow marking$, where $s$ is the sort of the place $p$.
  - $\forall t \in T.\ L\mathrm{R}(t) = L(t)$, where $L \in N$.
  - $\forall x \in X.\ L\mathrm{R}(x) = L(x)$, where $L \in N$.
  - $\forall a \in F.\ L\mathrm{R}(a) = \Phi$. Label expression on each flow relation is used to instantiate tokens in a place.
  - $\forall cons \in CONS.\ L\mathrm{R}(cons) = CONS$ , where $CONs$ is a set of constants.
- $R\mathrm{R}$ is the set of rewriting rules defined by constraint set $G$ that is associated with transitions $t \in T.\ \forall g \in G,\ t\ \forall\ T$,
  - if $g = \{true\} \vee g = \Phi$ of the transition $t$, we have an unconditional rule with the form $l : [p] \rightarrow [q]$, where $p, q \in Op\mathrm{R}$, $Op\mathrm{R} \in \Sigma\mathrm{R}$, with $p = Oppre(t)$ and $q = Oppost(t)$.
  - otherwise, $g = \{TermOp, bool(X)\}$, then $l$ is conditional rule with the form $l : [p] \rightarrow [q]$ *if* $\wedge i\ Ci$ where $Ci \in TermOp, bool(X)$ and $\forall vi \in Ci.\ vi \in X$, with $p = Oppre(t)$ and $q = Oppost(t)$ respectively.

It is worth to note that $E\mathrm{R}$ can be empty. We define a rewriting step for a Petri net based rewriting theory R*PrT* as follows.

**Definition 4 (rewriting step)** Let RPrT = < ∑R, ER, LR, RR > be a rewrite theory of a PrT, a proof term α is a term of the algebra TR(X) = T∑(X) (We assume that there are no clashes of names between the sets of operators).

A rewriting step is a triple < Δ, [s], [t] > (usually written as Δ : [s] → [t]) where α is a proof term and [s], [t] 2 T∑,E(X), s = pre(t)(φ(p) : α), and t = post(t)(φ(p) : α).

With the rewriting theory of Petri nets, we can give the following interleaving correspondence based on the defined rewriting step.

### 5.3 An Interleaving Correspondence

We state a claim that relates the semantics of PrT nets based on the rewriting theory R*PrT* . First we define a series of execution status of R*PrT* .

**Definition 5** Let Δ be a closed proof term over the rewriting theory RPrT . Then we have Δ as

- *initial, if it is element of sort SPrT .*
- *enabled, if it is initial and contains occurrence of an operator in Op*R *and E*R ≠ Φ.
- *one step, if it is enabled and contains occurrence of M*(*p*) *with p ∈ P and no occurrence of composition operator ;.*
- *fired, if it is one step.*
- *many steps, if* Δ = Δ1, …, Δ*n with* 1 ≤ *n* ≤ *Nat and each* Δ*i is one step.*
- *rewriting step, if it is many steps and all the one-step rewriting step is enabled.*

In the following, we define the relation between an occurrence for a transition $t \in T$ of a PrT net and a rewriting step $\Delta$.

**Definition 6** An occurrence for a transition $t \in T$, a guard $g \in G$ of a PrT net N is a rewriting step $\Delta t : [p] \rightarrow [q]$ if $\wedge i$ Ci with the following conditions:

- *a transition t is initial, if* $g(t) \in Term(X)$ *and* $\Delta t$ *is initial.*
- *a transition t is enabled, if t is initial and* $\Delta t$ *is enabled.*
- *a transition t is fired once under marking M*1 *and reach marking M*2*, if* $\Delta t$ *is one step.*

This characterization is needed to prove the correspondence between PrT nets and rewriting theory R*PrT* : given a PrT net *N*, there is a one-to-one correspondence between the computation sequence of PrT net *N* and the rewriting theory R*PrT* .

**Proposition 1 (One-step Correspondence)** Let $t \in T$ be a transition in PrT net N, M1 and M2 be two markings before and after t is fired, then a one step computation sequence for PrT net N is M1tM2.

If RPrT entails an initial, one-step rewriting step $\Delta : pre(t)(\varphi(p) : \alpha) \rightarrow post(t)(\varphi(p) : \alpha)$, then there is a computation sequence M1tM2 with $pre(t)(\varphi(p) : \alpha) \in M1$ and $post(t)(\varphi(p) : \alpha) \in M2$ defined for PrT nets.

*Proof.*

The proof has two steps on the definition of rewriting step. For any transition $t \in T$, from Subsection 2.1.1, with a one step computation sequence for PrT net *N M*1*tM*2,
$\exists$consuming-token-p(L((p, t), Terms$\Sigma$,X)) such that consuming-token-p(L((p, t), Terms$\Sigma$,X)) $\in$ M1, and $\exists$producing-token-q(L((t, q), Terms$\Sigma$,X)) such that producing-token-q(L((t, q), Terms$\Sigma$,X)) $\in$ M2.
Based on the definition 1, we can conclude the above proposition.

*endProof.*

We already observed in previous section that each place defines an _-signature on the set of sorts and the fixed sort *marking* as the operation/function with arguments. The markings are connected by transition firing defined by the Petri net semantics [19]. Thus we can conclude this section by the correspondence result between Petri net computational sequence semantics and the rewriting theory semantics.

**Corollary 1** Let Mi and Mj be two markings of a PrT net N. Then a computation sequence Mi…Mj entails a transition Mi to Mj iff RPrT entails a rewriting step $\Delta : Mi(p) \rightarrow Mj(p)$.

Finally, next result lifts the correspondence to computations.

**Proposition 2 (Computational Correspondence)** Let $M_{c0}$ and $Mc_j$ be two markings of a PrT net N. Then there exists a proved computation $\sigma = M_{c0tc0}\ldots t_{c\ j}\text{-}1M_{cj}$ with source $M_{c0}$ and target Mcj iff RPrT entails an initial one step sequential rewriting step $\Delta = \Delta 1, \ldots, \Delta m$ with $\Delta i : pre(tk)(\varphi(p) : \alpha) \rightarrow post(t)(\varphi(p) : \alpha)$ where $i \leq k \leq j$.

*Proof.*

This can be proved inductively on the computation sequence.

Base case: $k = c0$ we have the computation sequence *Mc*0*tc*0*Mc*1, from Proposition for the One-step Correspondence, we have $\Delta c0 : pre(tc0)(\varphi(p) : \alpha) \rightarrow post(tc0)(\varphi(p) : \alpha)$.

Hypothesis Assumption: Suppose it is true for $k = n$ where $i \le n \le j$ the proposition for computational correspondence holds. Then we have $\Delta = \Delta 1, \ldots, \Delta m$ with $\Delta i : (pre(tk))(\varphi(p) : \alpha) \rightarrow (post(tk))(\varphi(p) : \alpha)$ holds. Because R*PrT* entails an initial one step sequential rewriting step $\Delta = \Delta 1, \ldots, \Delta m$, there exists one rewriting step $\Delta I : \Delta 1, \ldots, \Delta i$ where $i \le k \le j$ and $\Delta i : pre(tk)(\varphi(p) : \alpha) \rightarrow post(tk)(\varphi(p) : \alpha)$. We need to show for $k = n + 1$ it still holds.
If $k = n + 1$, we need to consider two situations:

1) one is that $post(tn) = pre(tn+1)$;
2) $post(tn)$ is not $pre(tn+1)$. Because we use interleaving semantics for the PrT nets, and in the rewriting theory, each rewriting rule can be executed concurrently, $tn$ and $tn+1$ may be in a causal relation.

Based on the one-step correspondence, for each transition firing, we can have a rewriting step, so for $tn+1$ under some marking $Mn$, we have $\Delta cn+1 : pre(tcn+1)(\varphi(p) : \alpha) \rightarrow post(tcn+1)(\varphi(p) : \alpha)$.
For case (1), under the marking $Mn$, based on the hypothesis assumption, we know that $(post(tn)) = (pre(tn+1))$. From transition rule, the proposition holds.
For case (2), there must be some transition $th$ fired and make the proposition holds, i.e., let $th$ be the transition in the hypothesis assumption, that is $th = tn$. Similarly, the proposition holds.
*endProof.*
The characterization above is summarized as follows: for each firing transition, there is in fact a one-to-one correspondence between proved computations starting from the preset of the transition to the postset of transition and rewriting step with substitution _. Furthermore, for a firing sequence under a series of markings, there is a correspondence between the execution of the constructed Petri net and the rewriting sequence. The proofs of the correspondence validate the correctness of translation in Section 3.

## 6. Conclusion

We discussed a rewriting semantics of a software architecture specification (SAM) by a translation from specification to a rewriting logic based declarative programming language Maude. In addition, to validate the translation, we have shown the correctness of the translation by a stepwise proof. A case study of a coffee machine modeled in SAM is applied on the translation algorithm. This paper provides an alternative concurrent semantics of a formal architecture specification (SAM) using rewriting logic and equational logic implemented on Maude. As a high performance declarative programming language platform, Maude is efficient in different types of systems.
Maude provides modules that convert model in rewriting logic to Kripke structure so that it is able to do model checking using Maude. Moreover, Maude supports the specification of nested processes, which is hard accomplished in other model checking systems such as SMV or SPIN. It is known that nested processes are the natural way to specify distributed systems, which are supported by SAM. The Maude can model check systems whose states involve data types in any algebraic data types [6]. Other model checkers only support limited data types in terms of which all other data must be encoded. However the resulting Maude program after translation is kind of large, which is hard to debug even with the support of Maude tool.

An important aspect of our work is the validation of translation. In addition to the fact that translation validation is an important pragmatic procedure, we believe it is quite interesting in the context of architecture specification verification and automatic code generation. Given a SAM component or connector, it can be fully automatically translated to a Maude system module on rewriting logic without the need of any information from other components in a given architecture description. As the only visible entity in the architecture model, ports are mapped to the operations in the signature from sorts to the local state representation *Marking* and proposition sort *Prop*.

The transitions are mapped to rules in the rewriting logic with the variable substitution mapped to the rewriting steps. The semantics consistency between Petri nets' and rewriting logic's is seamlessly established through these mappings.

We are also interested in a deeper treatment of specification and verification features with a special focus on model checking techniques to tackle the state space problem. We will also work on validating the usability and expressiveness of SAM specification by implementing encodings of various process calculi that can be developed using Petri net and temporal logic.

## 7. References

[1] R. J. Allen. A formal approach to software architecture. PhD thesis, 1997. Chair-David Garlan.

[2] M. Clavel, F. Dur´an, S. Eker, P. Lincoln, N. Mart´ı-Oliet, J. Meseguer, and C. Talcott. Maude manual, 2004.

[3] M. Clavel, S. Eker, P. Lincoln, and J. Meseguer. Principles of Maude. volume 4 of Electronic Notes in Theoretical Computer Science. Elsevier Science Publishers, 1996.

[4] A. Corradini and F. Gadducci. Rational term rewriting. Lecture Notes in Computer Science, 1378:156–171, 1998.

[5] Z. Dong, Y. Fu, and X. He. Automated runtime validation of software architecture design. In Proceedings of Second International Conference of Distributed Computing and Internet Technology (ICDCIT 2005), volume 3816 of Lecture Notes in Computer Science, pages 446–457. Springer, 2005.

[6] S. Eker, J. Meseguer, and A. Sridharanarayanan. The maude LTL model checker. In Proceedings of the 4th International Workshop on Rewriting Logic and Its Applications (WRLA'02), volume 71 of Electronic Notes in Theoretical Computer Science, Amsterdam, September 2002. Elsevier.

[7] C. C. Frederiksen. Correctness of classical compiler optimizations using CTL. In Proceedings of theETAPS 2002 :European Joint Conference on Theory and Practice of Software, pages 41–55, 2002.

[8] Y. Fu, Z. Dong, G. Argote-Garcia, L. Shi, and X. He. An Approach to Validating Translation Correctness From SAM to Java. In Proceedings of The Nineteenth International Conference on Software Engineering and Knowledge Engineering (SEKE2007), 2007.

[9] Y. Fu, Z. Dong, and X. He. A Methodology of Automated Realization of a Software Architecture Design. In Proceedings of The Seventeenth International Conference on Software Engineering and Knowledge Engineering (SEKE2005), 2005.

[10] Y. Fu, Z. Dong, and X. He. An Integrated Runtime Monitoring Framework for Software Architecture Design. In Proceedings of the Software Engineering and Applications (SEA'05), 2005.

[11] Y. Fu, Z. Dong, and X. He. An Approach to Web Services Oriented Modeling and Validation. In Proceedings of the 28th ICSE workshop on Service Oriented Software Engineering (SOSE2006), 2006.

[12] Y. Fu, Z. Dong, and X. He. A method for realizing software architecture design. In Proceedings of the Sixth International Conference on Quality Software(QSIC'06), pages 57–64, Washington, DC, USA, 2006. IEEE Computer Society.

[13] Y. Fu, Z. Dong, and X. He. A translator of software architecture design from sam to java. International Journal of Software Engineering and Knowledge Engineering, 17(6):1–54, 2007.

[14] H. J. Genrich. Predicate/Transition Nets. Lecture Notes in Computer Science, 254, 1987.

[15] S. Glesner, R. Gei¨s, and B. Boesler. Verified code generation for embedded systems. In Proceedings of the COCVWorkshop (Compiler Optimization meets Compiler Verification), volume 65. Electronic Notes in Theoretical Computer Science (ENTCS), 5th European Conferences on Theory and Practice of Software (ETAPS 2002),, April 13 2002.

[16] J. A. Goguen and J. Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. Theoretical Computer Science, 105(2):217–273, 1992.

[17] K. Havelund. Using runtime analysis to guide model checking of java programs. In Proceedings of the 7th International SPIN Workshop on SPIN Model Checking and Software Verification, pages 245–264, London, UK, 2000. Springer-Verlag.

[18] K. Havelund and G. Rosu. Synthesizing monitors for safety properties. In Proceedings of Tools and Algorithms for Construction and Analysis of Systems, pages 342–356, 2002.

[19] X. He. A formal definition of hierarchical predicate transition nets. In Proceedings of the 17th International Conference on Application and Theory of Petri Nets, pages 212–229, London, UK, 1996. Springer-Verlag.

[20] X. He, J. Ding, and Y. Deng. Model checking software architecture specifications in sam. In Proceedings of the 14th international conference on Software engineering and knowledge engineering (SEKE'02), volume 27 of ACM International Conference Proceeding Series, pages 271–274, New York, NY, USA, 2002. ACM Press.

[21] X. He, H. Yu, T. Shi, J. Ding, and Y. Deng. Formally analyzing software architectural specifications using sam. Journal of Systems and Software, 71(1-2):11–29, 2004.

[22] K. Korenblat, O. Grumberg, and S. Katz. Translations between textual transition systems and petri nets. In Proceedings of the Third International Conference on Integrated Formal Methods (IFM'02), pages 339–359, London, UK, 2002. Springer-Verlag.

[23] L. Lu, X. Li, Y. Xiong, and X. Zhou. Xm-adl: An extensible markup architecture description language. In IEEE, pages 63–67. IEEE Press, 2002.

[24] D. Luckham, J. Kenney, and L. A. et al. Specification and analysis of system architecture using rapide. volume 21 of IEEE Transactions on Software Engineering, pages 336–355, 1995.

[25] N. Medvidovic and R. N. Taylor. A classification and comparison framework for software architecture description languages. Software Engineering, 26(1):70–93, 2000.

[26] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. volume 96, pages 73–155, 1992.

[27] J. Meseguer. Rewriting logic as a semantic framework for concurrency: a progress report. In Proceedings of International Conference on Concurrency Theory, pages 331–372, 1996.

[28] R. N. Taylor, N. Medvidovic, K. M. Anderson, E. J. W. Jr., J. E. Robbins, K. A. Nies, P. Oreizy, and D. L. Dubrow. A component- and message-based architectural style for gui software. IEEE Transactions on Software Engineering, 22(6):390–406, 1996.

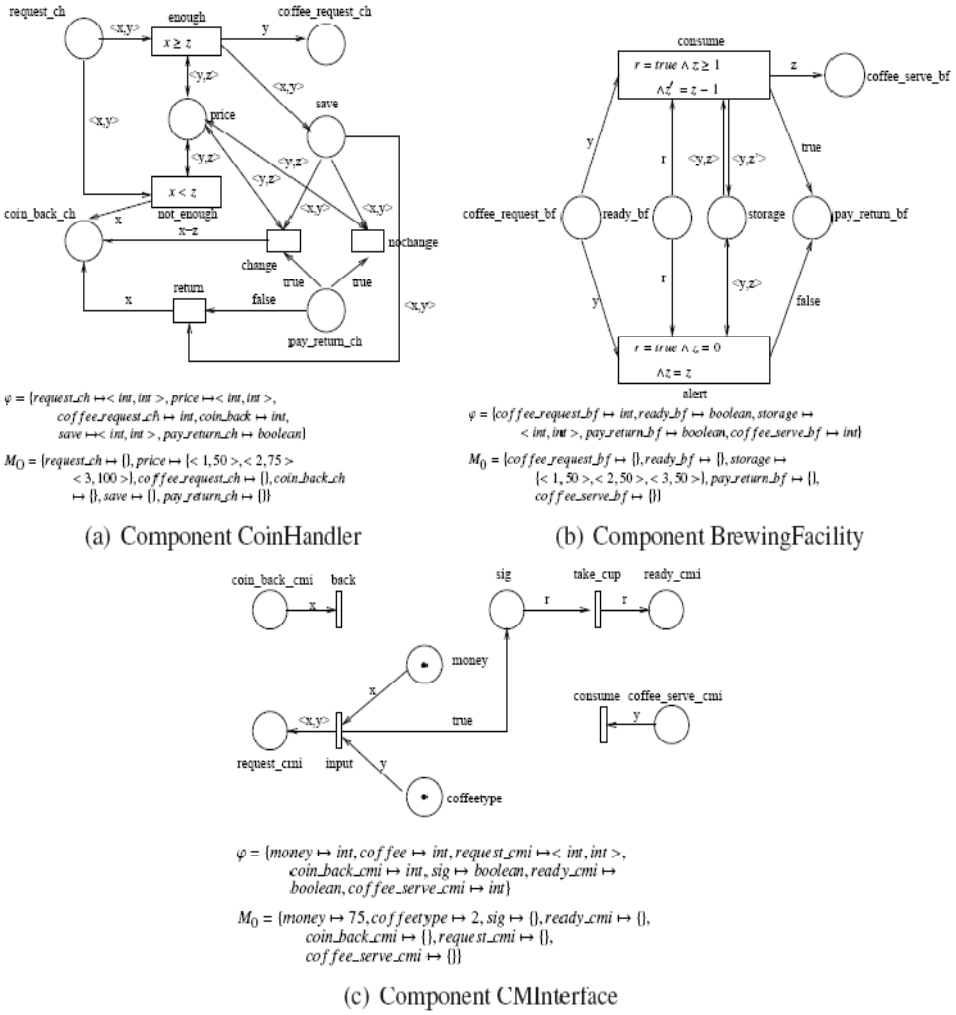
[29] W. M. P. van der Aalst, K. M. van Hee, and R. A. van der Toorn. Component-based software architectures: A framework based on inheritance of behavior. Science of Computer Programming, 42(2-3):129–171, 2002.

[30] J. Wang, X. He, and Y. Deng. Introducing Software Architecture Specification and Analysis in SAM through an Example. Information and Software Technology, 41(7):451–467, 1999.

[31] F. Xie, V. Levin, R. P. Kurshan, and J. C. Browne. Translating Software Designs for Model Checking. In Proceedings of 7th International Conference Fundamental Approach to Software Engineering (FASE), volume 2984, pages 324–338. Springer-Verlag, 2004.

**A. Behavior Model of Components in Coffee Machine**

The behavior model of each component in the coffee machine example is shown in the Fig. 3.




$\varphi$ = {*request_ch* ↦< *int*, *int* >, *price* ↦< *int*, *int* >, *coffee_request_ch* ↦ *int*, *coin_back* ↦ *int*, *save* ↦< *int*, *int* >, *pay_return_ch* ↦ *boolean*}

$M_0$ = {*request_ch* ↦ {}, *price* ↦ {< 1, 50 >, < 2, 75 > < 3, 100 >}, *coffee_request_ch* ↦ {}, *coin_back_ch* ↦ {}, *save* ↦ {}, *pay_return_ch* ↦ {}}

(a) Component CoinHandler

$\varphi$ = {*coffee_request_bf* ↦ *int*, *ready_bf* ↦ *boolean*, *storage* ↦ < *int*, *int* >, *pay_return_bf* ↦ *boolean*, *coffee_serve_bf* ↦ *int*}

$M_0$ = {*coffee_request_bf* ↦ {}, *ready_bf* ↦ {}, *storage* ↦ {< 1, 50 >, < 2, 50 >, < 3, 50 >}, *pay_return_bf* ↦ {}, *coffee_serve_bf* ↦ {}}

(b) Component BrewingFacility

$\varphi$ = {*money* ↦ *int*, *coffee* ↦ *int*, *request_cmi* ↦< *int*, *int* >, *coin_back_cmi* ↦ *int*, *sig* ↦ *boolean*, *ready_cmi* ↦ *boolean*, *coffee_serve_cmi* ↦ *int*}

$M_0$ = {*money* ↦ 75, *coffeetype* ↦ 2, *sig* ↦ {}, *ready_cmi* ↦ {}, *coin_back_cmi* ↦ {}, *request_cmi* ↦ {}, *coffee_serve_cmi* ↦ {}}

(c) Component CMInterface

Fig. 3. Behavior of Subcomponents in CoffeeMachine

7

# Transfer-Resource Graph and Petri-net for System-on-Chip Verification

Xiaoxi Xu and Cheng-Chew Lim
*The University of Adelaide*[*]
*Australia*

## 1. Introduction and Background

Verification of integrated circuits is an inherently difficult problem and the popular system-on-chip (SoC) design paradigm has brought about additional challenges. This section discusses these difficulties and challenges and gives an outline of our concurrency oriented solution to the SoC verification problem.

### 1.1 The VLSI Verification Problem

Verification of integrated circuit design is a process that checks the implementation of the design against its specification and identifies design bugs. The view that design *verification* is subservient to design *implementation* has soon become invalid when designs become just moderately complex. It has been claimed that the verification complexity is growing at a double-exponential rate (Saleh, 2004), i.e., exponential with respect to Moore's law. Consequently, nowadays, about 50%-80% of the design time and efforts are spent in very-large-scale-integration (VLSI) design verification. It becomes well known that verification is the biggest single bottleneck (Goering, 2008) in VLSI design.

There are two categories of verification methods.

- **Simulation-based methods**. In this category, the verification engineers develop a set of *tests* to stress a given design; the design is therefore often called the *design-under-test* or DUT. A test can be an abstract description about the control and observation applied upon the DUT. To actually apply the control and observation, a structure called test-bench (TB) needs to be constructed and simulated together with the DUT. The TB concretizes the abstract tests into "0/1" signals and directly interact with the DUT in these signals.
- **Formal methods**. In formal methods, verification engineers don't provide tests, but *design properties* – properties that a correct design should or should not have. Meanwhile, the design usually needs to be represented as a finite-state-machine (FSM) model. Then some model-checking tool computes whether the model abides by the properties. Formal methods form useful supplement to simulation in verifying control-intensive and FSM-based designs.

[*] This work was supported by the Australian Research Council under Linkage Project LP0454838.

Despite these approaches, verification of non-trivial design remains a major challenge due to its inherent difficulties.

• **The correctness issue**. Although a correct design specification is regarded as the foundation for verification, a real-world specification cannot be perfectly correct or complete. An SoC specification cannot foresee all corner-cases, especially when concurrency is concerned, and determine what behavior should be regarded as correct or incorrect.
• **The completeness issue**. For simulation, we have no absolute criteria to decide whether sufficient tests have been applied, because *passed tests do not indicate the absence of bugs*. Formal methods may exhaustively prove or disprove a given property, but the completeness problem turns into whether enough properties have been provided (Katz et al., 1999).

### 1.2 System-on-Chip (SoC) and Challenges for SoC Verification

The system-on-chip (SoC) solution – designing a whole system ready for application on a single chip – has become a popular VLSI design paradigm. Although there is no official definition of SoC, many would agree that a VLSI design that features *multiple components, including at least one processor, connected by on-chip interconnection* falls into the category of SoC. For example, the Nios SoC (Altera, 2004) in Figure 1 used in our research shows SoC features. The Nios CPU is a pipelined processor; the DMA can perform data transfer between any slaves on the bus; the full-duplex capable UART is the communication interface; ROM and SRAM store instructions and data respectively; RAM and FLASH are additional memories. The on-chip Avalon bus is actually a cross-bar interconnection made of a set of arbitrators, which communicate with each other. There are also a number of interrupt sources.

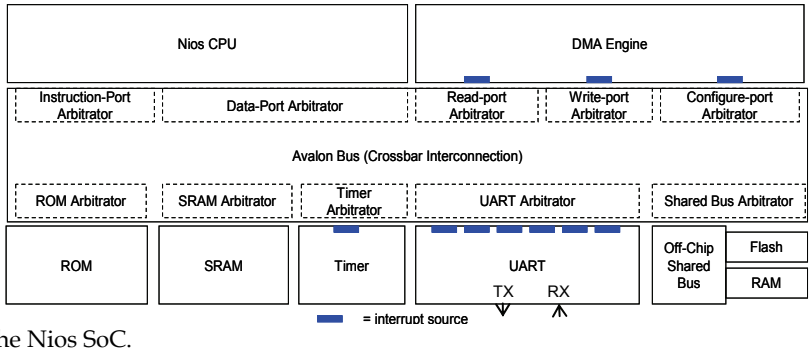



Fig. 1. The Nios SoC.

In addition to many advantages including higher performance, the SoC paradigm has practically reduced the process of designing a complex system into integrating some predesigned and reusable components. However, the verification of an SoC becomes even more difficult due to its features.
First, SoCs feature large scale of hardware (HW) integration. On one hand, the integration introduces *emerging properties* that are not present in any individual component and thus need *additional* control and observation effort; on the other hand, these components are not guaranteed to work together in all circumstances since they have been designed and verified separately under different sets of principles and assumptions.

Second, software (SW) running on the SoC processor(s) also contributes to the full SoC functionality, and makes an SoC behave much more intelligently than an FSM. Therefore checking hardware-software interactions becomes an important part of the system-level verification and introduces another new dimension to the verification problem.
Even if these two issues, i.e., HW-HW interaction and HW-SW interaction, are addressed *separately*, full system-level verification is still not achieved, because a system capable of running heterogeneous forms of interactions *concurrently* could be subject to various unforeseeable implementation bugs, including (Mosensoson, 2002):

- Interactions between blocks that are assumed verified;
- Conflicts in accessing shared resources;
- Arbitration problems and dead locks;
- Priority conflicts in exception handling;
- Unexpected hardware/software sequences.

These bugs are all related to interactions especially to concurrent ones with resource competitions. *Concurrency*, together with the associated *resource competition*, is the central characteristic of a system; therefore constructing concurrency is the key to system-level verification. However, current verification practices are not dealing with this concurrency problem effectively.
Formal methods are simply not at the position to do system-level verification. They work best with moderately complex, single-component and FSM-based designs; while a typical SoC shares none of these features. Moreover, formal methods, which require the user to provide properties, face a fundamental dilemma in dealing with SoC verification. An SoC, which is capable of heterogeneous HW-HW and HW-SW interactions, has *unforeseeable* failure modes; therefore, the user cannot postulate those properties they are yet to know.
System-level verification has to substantially rely on simulation. However, existing simulation-based methodologies such as VMM (Bergeron et al., 2005) put substantial focus on test-bench (TB) construction. A TB tends to stimulates and observes a DUT from its *exterior*; so it has inherent issues in controlling and observing the concurrency *internal* to the DUT. Moreover, in the mainstream TB-centric methodologies, software's position and roles are not given sufficient consideration to be well integrated in the verification framework.

### 1.3 Outline of Concurrency-Focused Solution to SoC Verification

To deal with the SoC verification challenges, particularly concurrency, we present two ideas: **Software-Centric Verification** (Xu et al., 2008)**.** This means that the software native to the SoC, instead of the test-bench (TB), should take more proactive roles in verification. In the sequel, we call software also as "test-program" (TP). TP takes high level *control* roles, especially in *concurrency management*; while TB only takes relatively passive *observation* roles. Software-centric verification reshapes the traditional verification framework, which heavily relies on complex TB for all control and observation roles. Figure 2 shows the new framework. The rationale, implementation and advantage of software-centric verification are detailed in (Xu et al., 2008)**.**

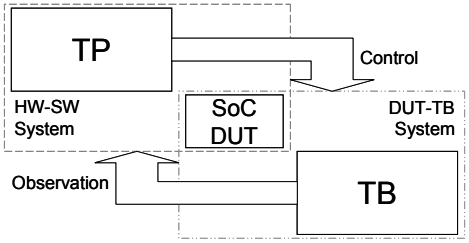



Fig. 2. Software-centric verification. The SoC and the test-bench (TB) form a DUT-TB system, which is under the active control of the test-program (TP); meanwhile, the SoC and the TP form a HW-SW system, which is under the passive observation of the TB.

**Interaction-Oriented Verification** (Xu & Lim, 2007). We also emphasize that objects-under-test are actually *interactions among components*, rather than the *components* themselves. Focusing on interactions (or communications) is already a common practice, particularly in transaction-level-modeling (TLM) (Cai & Gajski, 2003). TLM emphasizes the separation of components' *communication capabilities* with their *communication capabilities*. However, the verification community has not yet proceeded to migrate from the view in which interactions are treated as the properties, or "capabilities", *attached* to hardware components, to the view in which interactions themselves become a set of objects *independent* from hardware components.

Treating interactions as objects-under-test is philosophically attractive: the terms "tests" and "objects-under-test" are now referring to *one* category of entities – interaction-objects, which can naturally model "concurrency".

- As "tests", interaction-objects stress the communication mechanisms. Moreover, they can be combined to run *concurrently* to exercise the system more vigorously.
- As "objects-under-test", interactions bring their own properties to be tested, for instance, their temporal relations when they run *concurrently*.

The rest of this chapter handles some critical issues in our approach and they share the same theme – concurrency. Particularly, since *test-generation* and *coverage* are two fundamental aspects of simulation-based verification, we need to provide i) a method to generate test-cases of concurrency, and ii) a method to quantify the concurrency completeness. The first issue is addressed by a model called transfer-resource graph (TRG) and the second by a Petri-net derived from the TRG. We start our treatment by introducing the interaction model – *transfer*. A transfer is a *software-controllable interaction-object*, linking the ideas of software-centric and interaction-oriented verification.

## 2. The Transfer Model

This section discusses how to identify and characterize interactions, which appear to be abstract and shapeless, as valid objects that can be further combined to form concurrency.

### 2.1 Overview: Proper Abstraction Level

For system-level verification, interactions must be modeled at a proper abstraction level. Interactions in a system come in different levels, such as signal-level handshakes, logical level frames/packets/tokens, and also application-level threads/processes. The abstraction

level should not be too low, for we are to implement tests in software known as test-program (TP), which hardly have direct control and observation on signal-level events (e.g., Bus-Request and Bus-Acknowledge). However, the abstraction level should not be too high either, since after all a TP is supposed to vigorously stress hardware devices.
To trade-off the above considerations, the interaction model should be readily comprehended by a device level programmer, who understands hardware functionalities and performances, but may have little knowledge about hardware implementation. "Transfers" represent interaction-objects at this specific abstraction level.

### 2.2 Transfer Definition

While interactions need to be treated as objects-under-test, there is a challenging issue associated with modeling them; that is, interactions come in various forms, requiring different techniques to stimulate and observe them. Some interaction examples in the Nios SoC are the following.

(i) A Flash-to-RAM DMA transfers. It is a series of read/write operation driven by the dedicated hardware – the DMA engine.
(ii) The execution of a sort subroutine. It can be viewed as a pattern of memory access performed by the CPU; this kind of interaction is driven by the execution of SW.
(iii) An incoming bit stream via the UART receiver. The stream finally reaches a memory buffer. This process is mostly driven by the interrupt mechanism.

Note that these three examples are data flows driven by heterogeneous mechanisms. While checking each of them is common sense, checking their concurrent execution will greatly improve the test quality – we are able to observe not only interactions but also *interferences between interactions*. If the above three interaction examples take place in parallel, we will observe how the DMA engine and the CPU compete with each other for the bus access, how the UART will *interfere with* their competition by frequently interrupting the sort subroutine, and how the UART interrupt would be nested in the DMA interrupt.
The key to effectively constructing such parallelism is to generalize heterogeneous interaction forms into a common model, which we call *transfer-type*.
***Definition 1:*** A transfer-type is a set of software-controlled and data-intensive interaction patterns among SoC components. Its *software-controlled* feature means that a transfer-type has the following properties.

(i) **Configuration:** a transfer-type has its own parameters, which can be configured by some instructions. An important part of the configuration is the resources to be used.
(ii) **Invocation:** a configured transfer-type can be activated by some instructions. Invocation instructions are allowed to have side effects of configuration.
(iii) **Notification:** the completion of invoked transfer-type can be notified to software in some way (e.g., via interrupt), so that some software flag can indicate the event.

A closely related concept is the *instance* of a transfer-type called transfer-instance.
***Definition 2:*** A transfer-instance is a transfer-type bounded with a specific configuration.
We may view a transfer-type as a set of transfer-instances. When the discrimination between these two concepts is insignificant or can be inferred, we use the term *transfer*.
Figure 3 shows the life-cycle of a transfer, which includes a data-phase and a control-phase. Note that the configuration, the invocation and the notification are the overhead of a transfer and that the main body of a transfer is the data-flow.

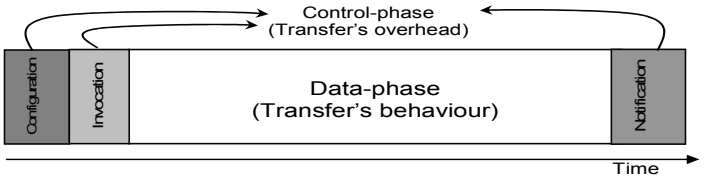



Fig. 3. The life cycle of the transfer model.

Notice that the differentiation between "control" and "data" is relative to the abstraction level. One control-operation at the transfer level, typically a register-write, is also a data-operation at instruction level; similarly, the data-flow of a transfer has already implicitly included physical-level control.

### 2.3 Transfer's Expression Power

In the early stage of SoC design/verification, the abstraction level should be high enough to hide the differences between hardware behaviors and software behaviors (Keutzer et al., 2000). Our transfer-type model meets this requirement. All three examples of interaction in Section 2.2 can be expressed as transfer-types.

(i) Transfer-type "Flash-to-RAM-DMA":

- **Configuration**: source-address, destination-address, transaction width/length;
- **Invocation**: set DMA engine control register 'go' bit;
- **Notification**: DMA finish interrupt.

(ii) Transfer-type "Sorting":

- **Configuration**: address, data type(signed/unsigned integer,etc), length, reverse;
- **Invocation**: call subroutine sort(address, type, length, reverse);
- **Notification**: the return of the subroutine.

(iii) Transfer-type "UART-Rx-by-Interrupt":

- **Configuration**: end-of-packet character, max-length, finish-mode (by max-length and/or end-of-packet char), error-detection-mode (parity, frame);
- **Invocation**: a STORE instruction to a special address – the test-bench/test-program interface; when this address is written, the test-bench starts to feed the SoC with a bit stream.
- **Notification**: the UART interrupt handler detects the Rx-finish conditions.

More generally, the *transfer* model can model three categories of data-intensive interactions.

(i) **Hardware Behaviors (Hard-transfers).** The read/write operations on the bus are driven by master devices, whose behaviors are mostly hardwired. So these operations are categorized as hard-transfers.

(ii) **Software Behaviors (Soft-transfers).** A processor in SoC is a valid master device, whose behaviors are programmable rather than hardwired. So its behaviors are called soft-transfers. There is a subtle but important difference between the code in a soft-transfer and the code in configuring a transfer. The former should be regarded as the *payload* code and subject to verification; and the latter is treated as the *overhead* for verification. One guideline to build soft-transfers is to compose read/write intensive subroutines to stimulate the interactions between CPU and slaves. However, soft-transfers do not have to transfer data *literally*; they can also be computation-intensive operations to apply stress to different types

of physical resources. For example, a deep recursive subroutine can apply stress to the register window mechanism in the Nios CPU architecture.

(iii) **HW/SW Cooperation (Virtual-transfers).** In the Nios SoC, the incoming UART byte-stream is formed by the cooperation between the UART, the interrupt subsystem and the UART-receiver-ready interrupt-service-routine (ISR). Although the byte-stream is physically performed by the CPU in the ISR, from a higher level of abstraction, it is functionally equivalent to perceive that a virtual master (also see Section 3.2) is conducting the stream between the UART receiver and a memory buffer, *independently* from the CPU, which may be involved in another task (at a reduced performance). Transfers conducted by virtual masters are called virtual-transfers. Unlike a soft-transfer, which explicitly requires a real CPU as its resource, a virtual transfer just requires a virtual master; therefore, we can arrange multiple virtual transfers (and one soft-transfer) to work "concurrently" on a single CPU. This concurrency is actually the parallelism between the CPU and peripherals. In a virtual transfer, the primary forms of interactions are interrupt request and response, while the read/write traffic on the bus may be secondary.

Table 1 lists the three transfer categories and summarizes how to implement their configuration, invocation and notification.

| Category | Transfer Examples | Configuration | Invocation | Notification |
|---|---|---|---|---|
| **Hard-tansfer** | Any DMA transfer | Setting control-registers | Setting control-registers | Hardware interrupt |
| **Soft-transfer** | UART polling; sorting; recursive subroutines; processor self-testing | Setting control-registers; Passing arguments to subroutines | Calling subroutine | Subroutine's return |
| **Virtual-transfer** | UART trdy/rrdy ISRs; Timer time-out ISR | Setting control-registers; Setting global variables | Enabling interrupt sources | The virtual master (ISR) itself |

Table 1. Implementation of hard-, soft- and virtual-transfers.

### 2.4 Transfer Complexity

To identify transfer-types in a given system, we need to discuss the complexity of the transfer model.

One transfer-type's complexity is caused by its configuration. We use **T** to denote the set of transfer-types in a system, and denote $T_i$ as each transfer-type member. For $T_i$, each of its parameter has a set of values to select from. $T_i$'s parameter-space is very application-oriented because parameters could be either dependent of or coupled with one another. Therefore, $T_i$ requires an operation $T_i.P(\cdot)$ – denoted from the object-oriented programming point of view – to perform its parameterization. The complexity of $T_i.P(\cdot)$ can represent the complexity of $T_i$. To let $T_i.P(\cdot)$ *deterministically* traverse its parameter-space seems neither necessary nor practical; therefore, we implement $T_i.P(\cdot)$ using weighted and constrained randomization.

Our transfer model is quite flexible in the sense that defining transfer-types allows for the trade-off between the number of transfer-types in a system and the complexity of their P(·)s. To one extreme, we could model only one single transfer-type to represent all possible interaction patterns in a system, but its P(·) needs to deal with a very large but also very artificially constrained parameter-space. To the other extreme, we could create a transfer-type for every possible interaction pattern of concrete parameters; then, we would have a huge number of transfer-types, while their P(·)s all have trivial complexity. In other words,

given an SoC, the more generalized each transfer-type is, the fewer transfer-types are required, but at the cost of more complex P(·)s.
In practice, it is natural to adopt this strategy: to generalize interaction patterns with *similar* parameterization style as one transfer-type. Taking the example of the Nios SoC, we initially modeled 12 transfer-types to represent DMA transactions among four source memory modules (ROM, RAM, FLASH, SRAM) and three destination memory modules (RAM, FLASH, SRAM). However, we have later decided to merge them into one transfer-type called "memory-to-memory DMA", with a single but stronger P(·) capable of assigning source and destination among all memory modules. Whereas, we consider it more appropriate to model "UART-Rx-by-DMA" and "UART-Tx-by-DMA" as separate transfer-types, which have very different parameters.

### 2.5 Transfer Temporal Granularity

To further characterize transfers, we give an estimation of their life-expectancy.
First, we discuss the necessity of comparable life-expectancy of all transfers. The transfer model enables us to generalize data-flows driven by various mechanisms, which could operate in a wide spectrum of data-rates. In our Nios SoC, transfer-type "ROM-to-RAM-DMA" has a rate of 33.3MB/sec; while the transfer-type "UART-Rx-by-interrupt" is operating at 14.4KB/sec. Now the question is: how to "match" concurrent transfers in order to achieve the desired verification quality, i.e., the parallelism and resource-contention? For example, does it make sense to create a test-case in which a 1000-byte-long transfer $T_1$ at the speed of 10MB/sec runs alongside another 1000-bytelong transfer $T_2$ at 10KB/sec? It appears to be a poor match, since $T_1$'s life is only 1/1000 of $T_2$'s, meaning that the parallelism and resource contention exist only 0.1% of the simulation. Therefore, it makes sense to configure all transfers to have *comparable* life-expectancies, say, within one order of magnitude of difference.
We now consider how to estimate the optimal life-expectancy. Common sense tells us that the life-expectancy should not be too long. This is because simulation is a very time-consuming process. In the shortest time possible, we not only need to cover most configurations for each given transfer-type, but also should try its concurrent running with other transfers. On the other hand, neither can life-expectancy be too short. We regard the data-phase of a transfer as its main body, in which parallelism and resource-competition are supposed to happen; whereas the transfer's control-phase (i.e., its configuration, invocation and notification) is the overhead. So it is natural to require the data-phase to be at least one order of magnitude longer than the control-phase; otherwise, a considerable portion of simulation time will be spent on the overhead.
Fortunately, the length of control-phase is predictable because all transfer-types' configuration/invocation/notification appear to be made up of instruction sequences of similar length. Hence, we assume that the following quantities are available:

- the average execution time of transfer configuration, C;
- the average execution time of transfer invocation, I;
- the average execution time of transfer notification, N.

Then we can reasonably conclude that the optimal transfer life-expectancy is simply in the range of (10 ~ 100)×(C+I+N), which makes the overhead well under 10%.

In the Nios SoC example, (C + I) requires 25 assembly instructions, or 100 SoC clocks. Transfer notification is typically by interrupt, which includes the time spent in context switching and ISR execution, so the average N is about 350 SoC clocks. Therefore the optimal transfer life-expectancy is in the range of (10 ~ 100) × (C + I + N), or 4500 ~ 45000 SoC clocks. This is also the temporal granularity of our proposed "system-level" tests. It guides us on how to model transfer-types, especially on how to bias their P(·)s' behaviors.

Now we have a quantitatively feel of the "abstraction level" of transfers to understand its relationship with other identifiable interaction-objects. *Transfers*, whose granularity is around $10^4$ cycles, is formed by the aggregation of signal-level *transactions* and *instructions*, whose granularities typically range around $10^1$ cycles; in turn, *transfers* themselves can be aggregated to become *processes*, whose granularity can range well beyond $10^6$ cycles. Transfer's unique granularity helps us to understand its features and limitations.

- It appears impractical and also unnecessary to consider concurrent transfers' temporal relations at clock-level accuracy.
- It is reasonable to assume that concurrent transfers do *not* need to communicate each other (as processes do), and that transfers do *not* use resources dynamically.

## 3. Resource and Transfer-Resource Graph

### 3.1 Overview: Resource-Contentions and Resource-Conflicts

The focus of system-level verification is concurrency. The main purpose of constructing concurrency is to observe interesting resource-competitions. Resource-competitions could happen in various domains, including the on-chip interconnection subsystem, interrupt mechanism, CPU-time and memory locations. Even more intriguing situation is that competitions in various domains can interfere with each other, as discussed in Section 2.2.

The strength of the transfer model is that it allows these heterogeneous competitions to be built naturally – we simply arrange multiple transfers to run concurrently. However, there should exist some principles to prevent unchecked or meaningless randomness. Our principle is to distinguish between resource-contentions and resource-conflicts. *Resource-contentions* represent physical level competitions that are supposed to be resolved by hardware mechanisms (e.g. bus protocol, interrupt handling scheme). These competitions are not just legal but also desirable. In contrast, *resource-conflicts* are competitions at the logical level and require programmer's discretion to *avoid*. For example, we should allow the DMA engine to compete with the CPU for a physical memory module, but we require that the DMA transfer should never access the memory addresses that are *currently* involved in a sort subroutine; because otherwise the results of both transfers will not be predictable from their configurations.

***Definition 3:*** Given a set **t** of transfer-instances $t_1$, $t_2$, ..., $t_n$, which are respectively instantiated from transfer-types $t_1$.T , $t_2$.T , ..., $t_n$.T , we assume that each $t_i$.T is associated with a pass/fail Boolean function

$$t_i.T.Check(t_i.configuration, MemRegSpace_{start}, MemRegSpace_{end}),$$

which, according to $t_i$'s configuration, checks if the running of $t_i$ has caused the expected changes in the memory/register space. If, for all i, $t_i$.T.Check() is constant regardless of $t_i$'s temporal relations (sequential, overlapping, etc) with all other transfers in **t**, we say **t** is free of resource-conflicts (with respect to those Check() functions).

This definition requires the result of each $t_i$ be *deterministically* predicted; but as a trade-off the temporal relations between conflict-free transfers are allowed to happen in an *indeterministic* manner; that is, if there are n conflict-free transfers, each having a "start" and an "end" event, then we shall allow for $(2n)!/2^n$ possible event sequences, all of which shall yield the same results in the memory/register space.

To avoid resource-conflict is reasonable – if each transfer's result can be predicted by its configuration and the contents in memory/register space, high level functional checkers, i.e., T.Check(), can be easily implemented in the test-bench. Not enforcing this restriction on resource-conflicts is still an option; in that case, the test-generator simply has more freedom, but it loses the capability to predict correct results, therefore the burden of predicting correct test results is left to the user.

Once resource-conflicts are avoided, no other restrictions are preventing a test-generator from constructing parallelism. In this way, resource-contentions at physical level are constructed implicitly.

### 3.2 Logical Resources

Since resource-conflict is a logical concept, we only need to model the local resources in the system. With this simplification, we only model three categories of resources: masters, registers and memory-ranges. We will see that this modeling is not as ad hoc as it may seem.

(i) **Master.** A master is defined as any device that can conduct a transfer-type. Examples of master in our Nios SoC include the read-master and the write-master of the DMA engine, and the data-master of the Nios CPU. Once modeled, a master is a trivial resource – the test-generator only needs a single bit to indicate its status: available or unavailable. However, the concept of *virtual-master* requires a little more insight into how to interpret system's behaviors. A virtual-master (also see Section 2.3) is an interrupt-service-routine (ISR) that cooperates with hardware to perform data-intensive operations, e.g. the UART receiver-ready-ISR is a virtual-master performing transfer-type "UART-Rx-by-Interrupt". A virtual master is usually capable of only one transfer-type, but we can model as many virtual-masters as necessary for an SoC, independent from the number of physical CPUs. Other examples in our Nios SoC include UART transmitter-ready-ISR and timer-ISR. Once modeled, the test-generator does not distinguish virtual and real masters. In this way, the resource-contention on CPU-time can be constructed implicitly.

(ii) **Register**. Registers are also simple resources. We only need to model data-intensive registers visible to programmers. Examples are the UART rxdata and txdata registers. Since control/status registers across an SoC are not suitable to be treated as data, they are not modeled as register resources. However, in fact, many control/status bits are already implicitly abstracted as masters.

(iii) **Memory-range.** Memory-ranges are flexible resources dynamically maintained by the test-generator. A memory-range is an object with properties of base-address, size, sub-word granularity and R/W mode. From within one free memory-range, test-generator can allocate sub-ranges of suitable size/location to some transfers; meanwhile, the unused fragments become free memory-ranges. The entire real memory space can be treated as the sole initial memory-range (if some minor constraints are resolved such as ROM cannot be written), so that an allocated sub-range can naturally cross boundaries between physical memory modules. Allocated memory-ranges can overlap if they are used as read-only. With these

arrangements, the test-generator naturally constructs resource-contentions on physical memories.

Just like that transfer-types are the generalization of similar transfer-instances (see Section 2.4), the above discussed logical resource types (master/register/memory-range) are the generalization of the bit-resources, namely, all bits in memory and registers accessible by a programmer. A bit, regardless of data-, control- or status-bit, is the finest logical resource object to a programmer; masters, registers and memory-ranges are simply different aggregations of bits. For instance, a physical master device's behavior is controlled and observed by the bits in its control/status registers; it is actually those control/status bits that are abstracted as one logical "master" resource. Therefore, the granularity of a master resource is a few control/status bits. Similarly a register's granularity is several data bits; and a memory-range's granularity is a lot of continuous data bits.

### 3.3 Definition of Transfer-Resource Graph

Transfers and resources interlink to form transfer-resource graph (TRG). TRG can be formally defined in terms of *transfer-instances* and *bit-resources*.

***Definition 4:*** A flat TRG is a triple G = (**t**, **r**, *u*), where:

**t** is a set of concrete transfer-instances in a system;

**r** is a set of bits accessible to a programmer; and

function *u*: $\mathbf{t} \times \mathbf{r} \rightarrow \{\mathsf{n}, \mathsf{s}, \mathsf{e}\}$, where n, s, and e respectively represent *no-use*, *shared-use* and *exclusive-use*. Notation "$u(\mathrm{t}, \mathrm{r}) = \mathsf{n/s/e}$" respectively means that transfer t will not use, share or exclusively use bit r.

Then the term "concurrency" can be defined in terms of *scenarios*.

***Definition 5:*** Given a flat TRG G = (**t**, **r**, *u*),

a) if $\mathrm{t}, \tau \in \mathbf{t}$, we say that t and τ *conflict with each other* if and only if
$$\exists\ \mathrm{r} \in \mathbf{r},\ (u(\mathrm{t}, \mathrm{r}), u(\tau, \mathrm{r})) \in \{(\mathsf{s}, \mathsf{e}), (\mathsf{e}, \mathsf{s}), (\mathsf{e}, \mathsf{e})\}.$$

b) A *scenario* **s** is a subset of **t** satisfying
$$\mathrm{t} \in \mathbf{s}, \tau \in \mathbf{s} \Rightarrow \mathrm{t} \text{ and } \tau \text{ do not conflict with each other.}$$

c) A *maximum scenario* is a scenario $\mathbf{s_M}$ satisfying
$$\forall \mathrm{t} \in \mathbf{t} \setminus \mathbf{s_M},\ \exists \tau \in \mathbf{s_M}, \mathrm{t} \text{ and } \tau \text{ conflict with each other.}$$

However, to implement a flat TRG is impractical due to the huge number of concrete transfers and bit-resources in a system. In order to visualize a TRG and generate scenarios practically, we use a different TRG definition based on transfer-types and logical resource models (masters/registers/memory-ranges).

***Definition 6:*** A TRG is G = (**T**, **R,** *U*), where

**T** is a set of transfer-types in a system; each transfer-type is a set of transfer-instances;

**R** is a set of logical resources; each resource is a set of bits; and

function *U*: $\mathbf{T} \times \mathbf{R} \rightarrow \{\mathsf{n}, \mathsf{s}, \mathsf{e}\}$. For each pair $(\mathrm{T}, \mathrm{R}) \in \mathbf{T} \times \mathbf{R}$, if all instances of T exclusively use all bits in R, then $U(\mathrm{T}, \mathrm{R}) = \mathsf{e}$; if no instances in T use any bit in R, then $U(\mathrm{T}, \mathrm{R}) = \mathsf{n}$; otherwise, $U(\mathrm{T}, \mathrm{R}) = \mathsf{s}$.

Figure 4 visualizes an abridged TRG for the Nios SoC. Arrows represent the transfer-types, the blocks represent the resources, and the letter e or s represents the access mode. Note that some ISRs are treated as master resources.

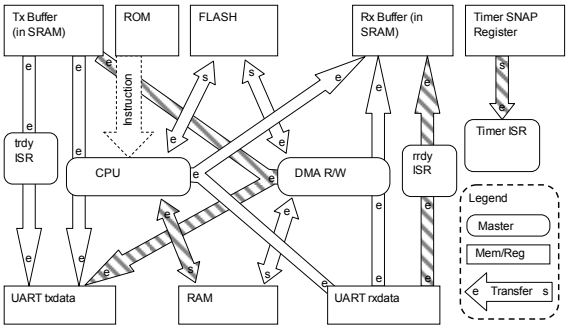



Fig. 4. The abridged transfer-resource graph for the Nios SoC. The shaded transfers form a scenario.

### 3.4 Implementing TRG for Scenario Generation

We implement TRG as a couple (**T**, **R**), where members in **T** and **R** are all intelligent objects aware of resource usage. A transfer-type T in **T** has a resource-allocation operation within its parameterization operation T.P(·); therefore it is denoted as T.P.A(·), which will set the allocated exclusive and total resource-usages respectively as $T.\mathbf{U}_e$ and $T.\mathbf{U}_t$, where $T.\mathbf{U}_e \subseteq T.\mathbf{U}_t \subseteq \mathbf{R}$. To construct a parameterized scenario, we need to search for any subset **S** of **T** and perform $T_i.P(\cdot)$ and $T_i.P.A(\cdot)$ of each $T_i$ in **S**, so that for any distinct $T_j$ and $T_k$ in **S**, $T_j.\mathbf{U}_e \cap T_k.\mathbf{U}_t = \varnothing$ and $T_j.\mathbf{U}_t \cap T_k.\mathbf{U}_e = \varnothing$.

Before we give a scenario generation algorithm, we need to introduce another internal operation of transfer-type T. Once T.P(·) has decided the concrete parameter-values, the test-generator needs an interpretation operation, denoted as T.I(·), to interpret the parameter-values into actual configuration/invocation instructions.

Given a TRG G = (**T**, **R**), let $\mathbf{R}_S$ and $\mathbf{R}_E$ respectively represent the current resources available for shared and exclusive access. The following algorithm constructs a maximum scenario.

(1) $\mathbf{R}_S = \mathbf{R}$; $\mathbf{R}_E = \mathbf{R}$;
(2) Randomly select a transfer-type T from **T**;
(3) Issue T.P(·), which in turn issues T.P.A(·), to parameterize/allocate resources to T (now T is actually a transfer-instance) so that $T.\mathbf{U}_e \subseteq \mathbf{R}_E$, and $(T.\mathbf{U}_t \setminus T.\mathbf{U}_e) \subseteq \mathbf{R}_S$;
(4) Issue T.I(·) and output the configuration/invocation instructions in a test-program;
(5) $\mathbf{R}_S = \mathbf{R}_S \setminus T.\mathbf{U}_e$; $\mathbf{R}_E = \mathbf{R}_E \setminus T.\mathbf{U}_t$;
(6) In **T**, drop any transfer-type that cannot obtain sufficient resources from the reduced $\mathbf{R}_E$ or $\mathbf{R}_S$.
(7) If **T** = ∅, one maximum scenario has been generated; otherwise repeat from step (2).

The four shaded transfers in Figure 4 form a maximum scenario. Although they appear loosely distributed in the TRG, the test quality is high because all hardware components are supposed to behave concurrently in simulation: the CPU is busy sorting data in the RAM, and will be interrupted by all peripherals; the DMA is transferring the tx data from a buffer to the UART; the Timer is counting down; and the UART is working in full duplex mode. Meanwhile, the instruction flow also contributes to the concurrency. The instruction flow is not as manageable as data-flows, but can be viewed as the "noise" to transfers. (However, we can extract rich information from it about the quality of TP execution (Xu et al., 2008).)

Therefore, high degree of resource-contentions will be achieved on various physical resources such as the bus, the slave interfaces, the interrupt mechanisms and CPU-time.
In our implementation, users can intervene with the test-generation by specifying a bias file, which biases most randomization operations in the test-generator, including:

- the behavior of transfer-type selection, i.e., step (2) in the above algorithm;
- the behavior of transfer-type parameterization, i.e., T.P(·);
- some control variables called environment parameters, which globally affect concurrent transfers, e.g. UART baud-rate and CPU data/instruction cache mode.

The bias file will also be used in test-generation with feedback information from Petri-net based post-simulation analysis. Section 5.3 provides further information.

### 3.5 Generation of Event-Driven Test-Program

Here we briefly introduce a scheme for step (4) in the above algorithm that allows the TP to be executed in an *event-driven* manner. This is illustrated in Figure 5.

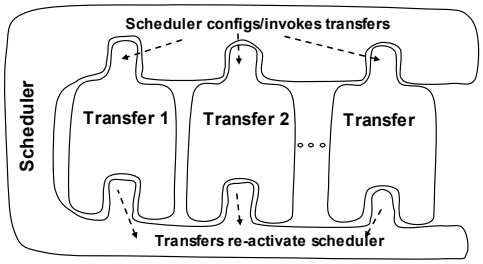



Fig. 5. Event-driven test-program.

The parallelism-management part of the TP, shown as Scheduler in Figure 5, configures and invokes some concurrent transfers. Any of these transfers, when finished, re-activates the Scheduler (e.g., for hard- and virtual-transfers, call-back Scheduler; for soft-transfer, return to Scheduler). In turn, the Scheduler can submit or resubmit transfer(s) because some resource must have been released by the finished one. In this way, the execution of a TP is actually driven by transfers' notification events. The majority of the Scheduler can be generated automatically by step (4) in the scenario generation algorithm.
Under this scheme, transfers – software-controllable interaction-objects – are no longer *passive* building blocks subject to arrangement, but become *active* ones which can stress the system rigorously, reflecting the rationale of our software-centric and interaction-oriented verification methodology. The comparison between two event-driven implementations and a polling-based scheme can be found in (Xu & Lim, 2007).

### 3.6 Features and Limitations

As a model at high abstraction level, TRG has the following features and limitations:
• TRG decouples two levels of complexity for test-generation – the complexity of each transfer-type and the complexity of their parallelism. The former is system-specific while the latter is relatively independent from a specific system, making TRG applicable to a wide range of designs.

• Modeling TRG does not require extensive knowledge about hardware implementation. Most effort is required in modeling each transfer-type, e.g., composing T.P (·).
• TRG is a method independent from simulation platforms or hardware abstraction levels. Tests generated from a TRG can be used as performance tests as well as functionality tests. SoC designers can use TRG to plan test-cases of parallelism to evaluate performance penalties due to resource-contentions long before a system is actually integrated; TRG may also find its applicability in generating manufacturing tests.
• The target bugs are not those bugs obviously local to each hardware component, but hard-to-detect bugs buried in close resource competitions. Therefore, hardware components are preferably free of obvious internal bugs.
• Result-checking is by means of checking the contents in memory and registers, which can be easily implemented as functional checkers in test-benches. However, a failed result gives limited indication of the nature and location of the bug. Therefore, other error-detection mechanisms (e.g., assertions) should also be implemented in test-benches.

## 4. TRG and Petri-net for Coverage

### 4.1 Overview

The simulation-based verification of a complex VLSI like SoC requires multiple coverage models. Each model measures simulation effectiveness from a specific perspective. At system level, since a system's behaviors can be described as concurrent interactions, one coverage model is needed to enumerate all concurrent interactions and the *temporal relations* between them. However, the widely used statement-based (line, toggle, conditional, etc) and local state machine based coverage measures cannot give such information. The temporal relations open up an enormous coverage space, requiring a mathematical model to deal with it. We choose Petri-net (Peterson, 1981; Zhu & He, 2002) as the model because its semantics can describe is resource-constrained concurrency.

***Definition:*** A Petri-net is a directed graph represented by a quintuple (**P**, **T**, **F**, $W$, $M_0$), where,

• **P** is a set of nodes known as *places*; each place can hold *tokens*. Tokens are all identical; ■
• **T** is a set of nodes known as *transitions*; ■
• **F** is a set of directed arcs (known as *flows*) connecting places and transitions, i.e., ■

$$\mathbf{F} \subseteq (\mathbf{P} \times \mathbf{T}) \cup (\mathbf{T} \times \mathbf{P});$$

• Function $W$: $\mathbf{F} \rightarrow \mathbf{N}^+$ ($\mathbf{N}^+$ : positive integers.); $W$(f) is called the *weight* of flow f; ■
• Function $M_0$: $\mathbf{P} \rightarrow \mathbf{N}$, (**N**: non-negative integers,) known as *initial marking*. $M_0$(p) is the number of initial tokens in place p. ■

A transition t is said *enabled* when the number of tokens in each input places is no less than the weight of the corresponding input flow. When enabled, t can (but does not have to) *fire*, consuming $W(f_i)$ tokens from the input place connected via $f_i$, and producing $W(f_o)$ tokens in the output place connected via $f_o$. Firing does not consume time. The firing sequence is called the *execution* of the net. The *state* of a Petri-net can be described in terms of its *marking*, i.e., the distribution of the tokens. A Petri-net has its *reachability graph*, whose nodes are the states and whose directed arcs represent the transitions between states.

### 4.2 TRG and Petri-net

TRG and Petri-net share some similarities in describing a system. Both formally define concurrency and conflict.

• In TRG, concurrency can be defined as a scenario with more than one transfer. In Petri-net, concurrency means a transition has multiple incoming or outgoing flows.

• In TRG, conflict means that the same resource is accessed by multiple transfers, and at least one access is exclusive. In Petri-net, conflict means a place has multiple incoming or outgoing flows.

Although capable to specify system-level concurrency and conflict, TRG lacks the capability to describe the *dynamics* of the system. As a high level *test-generation tool*, TRG cannot and does not need to deterministically specify temporal relations between concurrent transfers. The rich possibilities of temporal relations can only be, and *are meant to be*, realized in simulation. For example, TRG does not and cannot specify that, at which moment in transfer $T_1$'s life, another running transfer $T_2$ will finish. The timing that $T_2$ finishes is a complex function of its configuration, its submission timing and the resource-contentions between $T_1$ and $T_2$.

While a scenario in a TRG only represents a snapshot of data-flows, an execution of a Petri-net captures the temporal aspect of the system's behavior. Its reachability graph (readily obtained from a Petri-net tool) can enumerate the possible executions. Therefore, a Petri-net is suitable for *post-simulation* analysis of the temporal aspects of a system.

A desirable feature of TRG is that it can be readily converted to a Petri-net. Assuming that any transfer in TRG contributes two transitions in Petri-net, start and end, we can construct a Petri-net from a TRG by the following steps:

(1) **Converting Resources:** for each resource R in the TRG, create a place $P_R$ in the Petri-net;

(2) **Converting Transfers:** for each transfer T in the TRG,
- create two transitions $T_{start}$ and $T_{end}$;
- create a state-place $T_{running}$ (cf. resource-place $P_R$.);
- create two flows of weight 1, one from $T_{start}$ to $T_{running}$ and the other from $T_{running}$ to $T_{end}$;

(3) **Connecting Transfers and Resources:**
- First, for each transfer-resource pair (T, R) that satisfies $U(T, R) = \mathbf{s}$:
  - add one token into $P_R$;
  - create one flow of weight 1 from $P_R$ to $T_{start}$;
  - create one flow of weight 1 from $T_{end}$ to $P_R$.
- Then, for each transfer-resource pair (T, R) that satisfies $U(T, R) = \mathbf{e}$:
  - if $P_R$ has no token, put one token in it;
  - create one flow of weight n(R) from $P_R$ to $T_{start}$, where n(R) is the number of tokens in $P_R$;
  - create one flow of weight n(R) from $T_{end}$ to $P_R$.

Figure 6 shows a Petri-net constructed from the TRG of the Nios SoC.

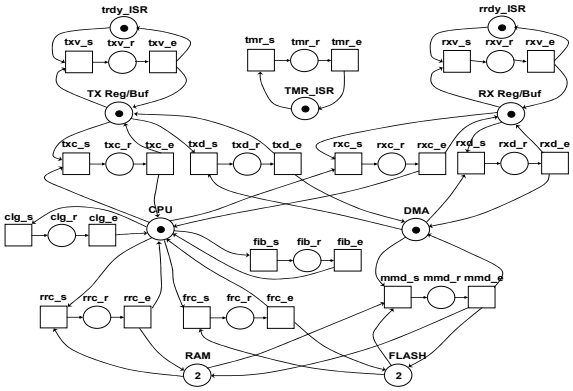



Fig. 6. Petri-net converted from a TRG.

The complexity of the above algorithm is linear to the size of TRG. Given a TRG ($\mathbf{T}_{trg}$, $\mathbf{R}$, $U$), the sizes of the resulting Petri-net ($\mathbf{P}$, $\mathbf{T}_{pn}$, $\mathbf{F}$, $W$, $M_0$) are:

- $|\mathbf{P}| = |\mathbf{T}_{trg}| + |\mathbf{R}|$;
- $|\mathbf{T}_{pn}| = 2|\mathbf{T}_{trg}|$, and
- $|\mathbf{F}| = 2|\mathbf{T}_{trg}| + 2|\{(T, R) : (T, R) \in (\mathbf{T}_{trg} \times \mathbf{R}), U(T, R) \in \{e, s\}\}|$.

### 4.3 Use of Petri-net

Once a Petri-net is obtained from a TRG, we can use the net in a number of ways. For instance, we could prove the liveness and boundedness of the Petri-net; we can simplify the Petri-net (but keep the reachability graph isomorphic), then we are able to map the simplification back onto the TRG. However, these rather theoretical topics are beyond the scope of this chapter.

The most practical use of the Petri-net is to define the coverage space. The coverage space is based on the reachability graph associated with the net. For instance, the reachability graph can indicate the total number of (unparameterized) scenarios in the TRG, because each state in the reachability graph simply represents a scenario in the TRG. This size contributes to the total complexity of scenario generation algorithm in Section 3.4. Generally, there are several options to define the space:

- All states in the graph (i.e., markings);
- All state-state transitions in the graph (not the transitions in the Petri-net);
- All paths in the graph;
- All cycles in the graph.

These options represent the different levels of temporal details. In (Zhu & He 2002), a number of coverage-space definitions based on the reachability graph are proposed. These definitions roughly fall into: (1) state-based category, (2) transition-based category, and (3) flow-based category. Some coverage spaces could be just too enormous to be practically covered due to the graph size and connectivity, in such cases, some restrictions can be applied to bound the coverage space, such as limiting the length of the path and the size of the cycle.

To check the coverage, we need to collect transfers' start/finish event history from the simulation trace. This history can be easily collected, because each transfer has a software flag indicating whether it is running. The Petri-net reads the event history to re-play the transition firing sequence. Its reachability graph is traversed in this manner. The traversed states, transitions and other coverage points (cycles/paths) are counted and compared with the coverage space size, and then the percentages are reported.
Besides indicating the completeness of temporal relations, the coverage information can be further used to guide test-generation. We have implemented test-generation with feedback at state and transition level. See Section 5.3 for details.

### 4.4 Discussion

It should be noted that both TRG and the Petri-net converted from a TRG are high level abstraction of an SoC (with its application). Most resource-contentions at physical level are invisible in the TRG and the derived Petri-net, simply because physical resources are not present in these models. This is a feature as well as a limitation. If we feel the resulting Petri-net is too "coarse" for coverage purpose, we can consider the following approaches.

- Adjust the size of the TRG by dividing a transfer-type into some sub-types (see Section 2.4), so that the resulting Petri-net has more states and state-transitions. This is in effect to take transfer parameterization into coverage consideration.
- Include more aggressive coverage spaces, such as path coverage and cycle coverage. This is effectively to take more detailed temporal relations into coverage consideration.
- Take system-specific knowledge into coverage consideration. We could model a transfer as an FSM with more internal states and transitions in addition to the simple "start", "running" and "finish", and convert it to a state-machine typed Petri-net (i.e., a net with only one token) and embed it in the backbone net derived from the TRG. It can be seen that TRG-derived Petri-net provides a reasonable start-point and backbone for a more accurate coverage model.

## 5. Experiments

### 5.1 Statement Based Coverage Measures

We have observed that reasonable statement coverage measures can be achieved by test-programs generated from a TRG. Since another software-based test generation methodology called SALVEM (Cheng et al., 2005) is demonstrated on the same Nios SoC, we compare the results of SALVEM tests with the test results of TRG. Figure 7 shows the comparisons of the statement (toggle and conditional) coverage. The TRG method has higher coverage on some components but is lower (but comparable) on the CPU, which has 11,000 lines of code and is the most complex component in the system. The lower coverage on CPU using TRG method may be attributed to the fact that we have not put too much effort in manually creating subroutines stressing the processor itself, which we believe need another level of automation beyond the scope of this article. We believe that the TRG method imposes no restrictions on achieving reasonably high statement-based coverage.

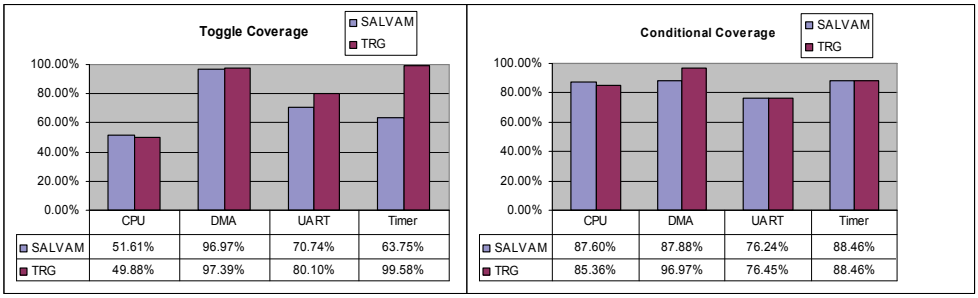



Fig. 7. Toggle and conditional coverage comparisons.

### 5.2 State Space Traversing

The statement-based coverage measures give little information regarding system-level concurrency and resource-contention. Therefore we attempt to indicate this information using "system state space". We define the state space as the space made by the concatenation of the major control/status registers in the SoC components (CPU, DMA, UART and Timer). The concatenation is 64-bit long; the theoretical space of size $2^{64}$ is so large that even its reachable sub-space is impractical to be traversed exhaustively by any set of real-world tests. However, we can statistically measure how fast states can change and how fast new (i.e., unprecedented) states will emerge. These values are useful since system states can give information regarding concurrency. For example, from the traversed states, we can tell if all peripherals have requested interrupt simultaneously.

We compare the capabilities to traverse state-space between two sets of TPs. One set contains TPs filled with scenarios holding one or two transfers, and the other contains TPs filled with maximum scenarios. Figure 8 shows the rate of state-change. The high concurrency TPs has a state-change rate roughly two times the rate of the low-concurrency TPs. Faster state-change rate implies that more events are happening simultaneously.

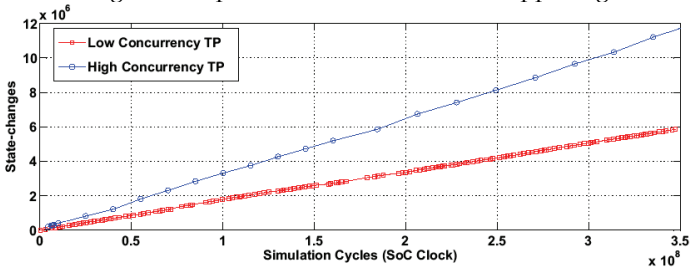



Fig. 8. State-changes against simulation cycles.

However, faster state-change rate does not necessarily mean efficient state-space traversing, for any state can recur many times. We further compare how fast *unprecedented* states emerge in simulation. Our experiments show that low-concurrency TPs have traversed about $10^5$ distinct states in 420 million SoC clocks (12 computing hours on a 3+GHz workstation); in comparison, high-concurrency TPs can traverse $10^6$ distinct states in the same simulation duration. In Figure 9, each data point represents one simulation of a TP. We observe that high-concurrency TPs produce new states at a much faster speed; and the

speed is less sensitive to the number of known-states. This encouraging comparison implies that concurrency is the key to efficiently exploring the state space.

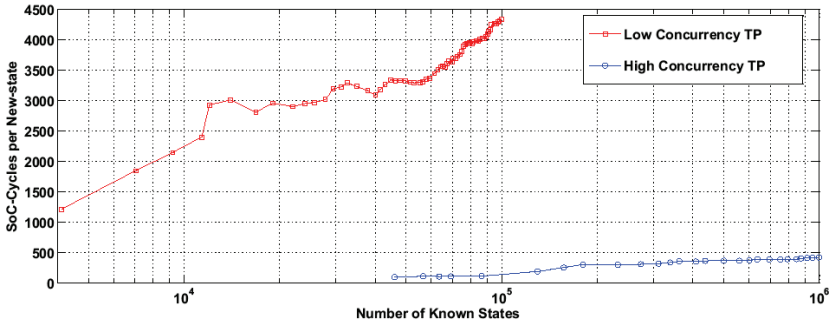



Fig. 9. New-state emergence rate against the number of known-states.use

### 5.3 Test Generation with Feedback

We modeled a TRG with 12 major transfer-types for the Nios SoC. Our generator can exhaustively (but randomly) produce 139 transfer-subsets to be the (unparameterized) scenarios. This is well predicted by the reachability graph, which has 140 states, with the additional state representing the empty scenario. The reachability graph also contains 772 transitions.

We have achieved test-generation with feedback at state-level and transition-level. A simulation-trace analyzer is developed. The analyzer is responsible for the following tasks:

- Count states and transitions in the trace log file;
- Compare the counts with the total states and transitions in the reachability graph;
- Identify the target (i.e., uncovered or less frequent) states/transitions;
- In a bias file (see Section 3.4), adjust the randomization arguments about transfer-selection and transfer parameterization.

The state-level feedback is straightforward because a state in the reachability graph simply represents a scenario in the TRG. Once a target scenario is identified, in the bias file, we simply increase the selection weights of the transfer-types which make up the target scenario. Thus the test-generator will be more likely to generate the target scenario.

The transition-level feedback requires additional consideration. A transition in the reachability graph is a transfer-start event $T_s$ or a transfer-end event $T_e$, which separates two scenarios $S_1$ and $S_2$, i.e., $S_1 \xrightarrow{T_s} S_2$ or $S_1 \xrightarrow{T_e} S_2$. Thus the analyzer needs to manage both target scenario ($S_1$ or $S_2$) and target event ($T_s$ or $T_e$).

First, we identify the target scenario:

- In case of $S_1 \xrightarrow{T_s} S_2$, the target scenario is $S_2$;
- In case of $S_1 \xrightarrow{T_e} S_2$, the target scenario is $S_1$;

Once the target scenario is identified, we can apply the same mechanism as that used for state-level feedback in order to make the target scenario more likely to happen.

Second, we need to make the target event happen earlier in the current scenario (in order to enter or leave the target scenario, otherwise the current scenario changes). For each transfer-type, we define one of its parameters as its life-expectancy, which controls how long a

transfer will be running. For example, for a transfer-type "RAM-to-Flash DMA", the parameter "DMA length" is the life-expectancy parameter. The analyzer then adjusts the randomization ranges of the life-expectancy parameters in the bias file: it reduces the life-expectancy of T and/or extends the life-expectancy of the rest transfers in the target scenario. Therefore, in simulation, the target event has more chance to fire earlier to enter or leave the target scenario.

Figure 10 includes the accumulative state-coverage and transition-coverage comparisons between two sets of 20 simulation-runs, one with feedback and the other only with scenarios generated randomly. (Each set needs approximately 15 computing hours on a 3G+Hz 1G RAM workstation.) The figure shows that, with feedback, all states and transitions are covered in the first several runs. For the 20 runs without feedback, state-coverage space is traversed 5 times slower, and the transition coverage space cannot be traversed in 20 runs.

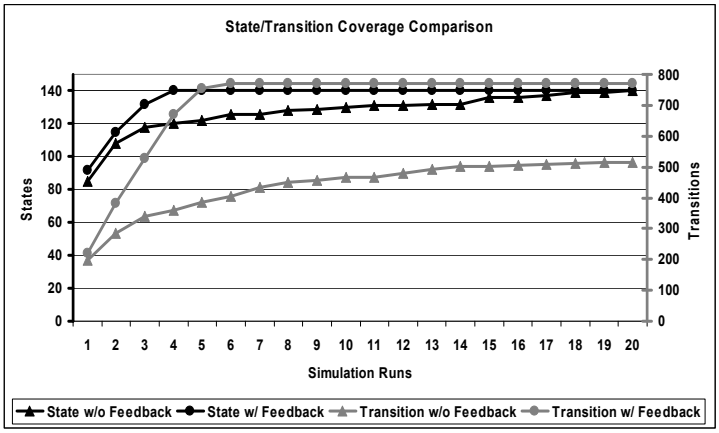



Fig. 10. State coverage and transition coverage with and without feedback.

It should be noted that the fast traversing on states and transitions does not mean that the whole verification process is complete. If more detailed temporal relations (e.g., path/cycle) and other variations such as transfer parameterization are taken into account, more scenarios are needed. The fast traversing does give us a chance to focus on other coverage areas. Like all feedback techniques, our feedback scheme only targets at one type of coverage.

## 6. Conclusion

Software-centric and interaction-oriented verification presents a natural interpretation of a system: a system is made up of programmer-controlled data-flows, i.e., transfers, which are constrained by programmer-controlled resources. As the result, the central part of the system-level verification problem is now modeled as a concurrency problem and can be dealt with concurrency models. Particularly, the test-generation problem is handled by the TRG model and the coverage measures can be treated by the Petri-net model, whose backbone can be converted from the TRG model.

Our method has been successfully demonstrated on the single processor Nios SoC. But the basic idea of combining data-flows with resource contentions is generic, making it applicable to a wide range of SoCs. In our future work, we will apply the model to verifying more sophisticated SoCs with multiple processors and multiple bus hierarchies. Another research area is the coverage model of parallelism and resource-contention. While the current Petri-net model derived from a TRG can represent certain level of resource-constrained concurrency, we may need to incorporate the domain knowledge about a real world system to capture enough information regarding fine-grained resource competitions. More research in this area is necessary on problems such as to include how much domain knowledge to trade-off between accuracy and scalability.

## 7. References


Altera, Inc. (2004). *Nios Hardware Development Tutorial ver 1.2*, 2004, http://www.altera.com/literature/tt/tt nios hw.pdf

Bergeron, J.; Cerny, E.; Hunter, A. & Nightingale, A. (2005). *Verification Methodology Manual for SystemVerilog*. Springer Science Business Media, Inc., ISBN: 978-0387255385, NY

Cai, L. & Gajski, D. (2003). Transaction Level Modeling: An Overview, *Proceedings of International Conference on Hardware-Software Codesign and System Synthesis*, pp. 19-24, ISBN: 1-58113-742-7, Newport Beach, CA, Oct 2003

Cheng, A.; Parashkevov, A. & Lim, C.-C. (2005). Verifying System-on-Chips at the Software Application Level, *Proceedings of IFIP-WG Conference on Very Large Scale Integration System-on-Chip*, pp. 586-591, ISBN: 0729806103, Perth, Australia, Oct 2005

Goering, R. (2008). Ten 2008 Trends in System and Chip Design. *SCD Source Online Article,http://www.scdsource.com/article.php?id=68, Feb 2008.*

Katz, S.; Grumberg, O. & Geist, D. (1999). "Have I written enough Properties?" – A Method of Comparison between Specification and Implementation. *Proceedings of the 10th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, pp. 280–297, ISBN:3-540-66559-5, London, UK, 1999

Keutzer, K.; Malik, S.; Newton, R.; Rabaey, J. & Sangiovanni-Vincentelli A. (2000). System Level Design: Orthogonalization of Concerns and Platform-Based Design, *IEEE Transactions on Computer-Aided Design*, Vol 19, No. 12, (Dec 2000), pp. 1523-1543, ISSN: 0278-0070

Mosensoson, G. (2002). Practical Approaches to SoC Verification. *Proceedings of DATE User Forum*, 2002.

Peterson, L. (1981). *Petri-net Theory and the Modeling of Systems*, Prentice Hall, ISBN: 978-0136619833, Upper Saddle River, NJ, USA

Saleh, R.; Wilton, S.; Mirabbasi, S.; Hu, A.; Greenstreet, M.; Grecu, C.; & Ivanov, A. (2006). System-on-Chip: Reuse and Integration. *Proceedings of the IEEE*, Vol 94, No.6, pp. 1050–1069, ISSN: 0018-9219

Xu, X.; Lim, C-C. & Liebelt, M. (2008). Positioning Test-Benches and Test-Programs in Interaction-Oriented System-on-Chip Verification, *Proceedings of IEEE International Workshop on High Level Design Validation and Test*; pp. 3-10, ISBN: 978-1-4244-2922-6, Nevada, Nov 2008

Xu, X. & Lim, C.-C. (2007). Using Transfer-Resource Graph for Software-Based Verification of System-on-Chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 27, No. 7, (July 2008), pp. 1315-1328, ISSN: 0278-0070

Zhu, H. & He, X. (2002). A Methodology of Testing High Level Petri-nets, *Information and Software Technology*, Vol 44, (June 2002), pp. 473-489,

8

# Using Petri nets for modeling and verification of Hybrid Systems

Ricardo Rodriguez[1,2], Otoniel Rodriguez[3],
Gerardo Reyes[4] and Vianey Cruz[4]
[1]*Technological University of Ciudad Juarez*
[2]*Technological Institute of Ciudad Juarez*
[3]*Autonomous University of Morelos State*
[4]*National Center of Research and Technological Development*
*Mexico*

## 1. Introduction

Artificial Intelligence (AI) is a science that came out in the decade of 50s. It emerges having as main objectives the understanding of intelligence and the building of intelligent behaviour systems that carry out complex tasks with a superior or equal level competence to a human performing. This science uses theoretical and experimental Computational techniques in order to emulate human intellectual activities: learning, perception, reasoning, creation or manipulation (Negnevitsky, 2005; Nikolopoulos, 1997).

The AI counts on several disciplines that have arisen in some application fields and it is approached to problems in industrial and commercial sectors. Such is the case in Knowledge-Based Systems (KBS) that have received great attention and have became to be essential tools in business, science, engineering, manufacturing and many others fields.

A KBS is a computational system, it has knowledge, ability and experience that only belong to a person or specialist people group in some area of the human knowledge, in such way, this system can solve specific problems in intelligent and satisfactory way (Negnevitsky,2005).

KBS provide assistance to humans in decision making and normally can explain the reasoning used to obtain a diagnostic or suggestion, and why the questions were inferred as well. These systems can manage large knowledge amount established in declarative form to be used in the intuition and experience. Such knowledge can be integrated with some knowledge representation: rules, facts, heuristics, and in some cases includes uncertainty.

A KBS allows the user to introduce facts or information to the system and obtain as results, advices or experiences. The relevant components of a KBS are: the knowledge base and the inference engine. KB represents the knowledge in application domain. Inference engine gives reasoning capability to solve a particular problem.

The knowledge obtained of a human expert is transformed into a representation frame and the inference is done using it. However, knowledge is not always totally incorporated because of its nature. It causes incompleteness problems in the KB.

The knowledge base is the heart of a KBS and it is coded in some representation language. A particular case of KBS is the Neural-Symbolic Hybrid System (NSHS) that allows interaction of the connectionist and symbolic knowledge contained in it (Nikolopoulos, 1997; Cruz, 2004; Santos, 1998). Hybrid systems allow integrating two or more knowledge representations in one system. It is in order to obtain a knowledge integrating that allows improving the global efficiency of the system. NSHS use an Artificial Neural Network to improve the knowledge that provides a symbolic system. Each one of these subsystems maintains its own representation language and a different mechanism to infer its solution.
When a NSHS is bought, there is a specialist that provides his experience in the application field. Therefore, it is coherent and without contradictions. However, in these systems mistakes can potentially arise as a result of the modeling complexity and the knowledge among to represent.
These systems have been used mainly in classification problems. The knowledge base of a NSHS and any knowledge-based system can contain mistakes since several experts provide their experiences in the field of application, which can be contradictory. Therefore, the verification and validation of knowledge in this kind of systems are critical processes in their construction, and may be focused on the knowledge base or the inference engine.
As the NSHSs gain acceptance, it increases the necessity of ensuring the automatic validation and verification of the knowledge contained in them. Verification has as a principal objective to ensure the consistency and completeness of the KB. Although, it does not warrant that its behaviour corresponds with the human expert knowledge. In the verification processes defined characteristics of a NSHS are evaluated. Such analysis can be restricted to one specific element of the system. As a matter of fact it can be in the inference machine, KB, or user interfase, and it can be focused on system specific stages, for example, in its deductive behaviour (Negnevitsky, 2005).
When the verification process has been finished, the validation process takes place analysing the proper of KB and the possibility of obtaining right solutions to the domain problems.
In some application environments, a NSHS might not be easily accepted nether be started up, at least, it can be conveniently and meticulously proved that works according to expectations (Nikolopoulos, 1997).
The present chapter presents an enhanced Petri net model for the detection and elimination of structural anomalies in the knowledge base of the NSHS. In addition, a modeling process is proposed to evaluate the obtained results of the system versus the expected results by the user. The validation and verification method is divided in two stages: 1) it consists of three phases: rule normalization, rule modeling and rule verification. 2) It consists of three phases: rule modeling, dynamic modeling and evaluation of results. Such method is useful to ensure that the results of a NSHS are correct. A set of tests is presented to demonstrate the effectiveness of the results obtained with this method. The cases of study were obtained from KBs extracted from Neural-Symbolic Hybrid Systems.
The chapter is organized in the following form: section 2 explains some related background in knowledge verification and validation; section 3 describes a Hybrid System Framework with combined knowledge; section 4 presents some aspects of error detection and system modeling; section 5 gives some information about Petri nets modeling; section 6 describes our verification method applied to error detection; section 7 presents our knowledge validation approach; experimental results and discussions are provided in section 8; finally, in section 9 we conclude the chapter and point out the direction of future research.

## 2. Related Work

A number of works dealing with knowledge verification and modeling have been proposed in the past. Such works have been based on the comparison of rule pairs. However, recent proposals use techniques such as Petri nets, directed graphs and directed hipergraphs (He et al., 2003). In these approaches, nodes are used to represent simple clauses of a rule and directed arcs to represent causal relations.

Petri nets have been used in the study of RBS due to the possibility of capturing dynamics and structural aspects of the system. The rule base can be verified by Petri net analysis techniques. Those techniques have been used in several works. In Yang et al. (1998) an error verification method in a Rule Base (RB) based in an incidence matrix is proposed. This method does not admit negated propositions. It makes a previous ordering of the RB for the verification and it does not need an initial marking of the net for the verification. In He et al. (2003) a reachability graph of a Petri net (PN) for structural anomalies detection in a knowledge base (KB) is presented. This technique is known as $w$-Net, where $w$ indicates the amount of tokens existing in each place. Nevertheless, in this technique it is necessary to know the initial marking of the net to detect errors. In Wu & Lee (1997) a variant of classic Petri net named high level extended Petri net is proposed. This model allows the logical negation and the use of variables and constants in the antecedent as well as consequence of the rules. Execution of the model is made by means of input conditions. It uses a reachability approach based on a color scheme for validation.

## 3. Knowledge Representation in Hybrid Systems

Neural-Symbolic Hybrid Systems are computer programs based on artificial neural networks that also interact with symbolic components (Cloete & Zurada, 2000). These types of systems integrate the connectionist and symbolic knowledge, in such a way that the knowledge contained in each one of these is complemented (Cruz et al., 2005; Cruz et al., 2006; Negnevitsky, 2005; Santos, 1998).

The symbolic knowledge is a set of theoretical knowledge from a particular domain. This knowledge should be translated into a formal representation in order to be used in a computer system. Some knowledge representations are: semantic networks, predicates logic, proposition logic, etc. The representation mostly used is the production rules. A disadvantage of the symbolic representation is that sometimes the characteristics of the objects can not be totally described. It is due to such representation can not make an exhaustive description of the object in all its modalities or contexts.

On the other hand, a different type of knowledge is known as "practical", integrated by a group of examples about an object or problem in different contexts or environments. For example in object recognition, an image base of the objects can be used to describe it in different contexts, positions, environments and with different focus of external quality. In some cases, a numerical representation of RGB colour, high or wide can be important. According to the above mentioned a hybrid approach can be the solution to object recognition problems.

NSHS are a type of Knowledge -Based Systems that can be used in many applications where failures can be expensive in services, properties, or even the life (Cruz, 2004; Tsai et al., 1999). It is important the verification and validation of these systems before their implementation to ensure this way their reliability. The necessity to develop knowledge

verification and validation systems will increase to guarantee the quality and reliability of such systems.
The extracted knowledge of the system is composed of the production rules. Users of a NSHS can interact with the knowledge base because of its representation is extracted and understandable.

### 3.1 Development Stages of a NSHS

NSHS transfers knowledge represented by a set of symbolic rules toward a connectionist module. The obtained neural network permits a supervised training, starting from a base of examples. In the next step, an extraction algorithm is developed to obtain the knowledge of a neural network into production rule form. Finally, the rules extracted must be verified and validated to be sure that the knowledge obtained in the extracting process is suitable to solve the problem (Cruz et al., 2006; Santos, 1998; Villanueva et al., 2006). The stages of a NSHS are shown in the figure 1.

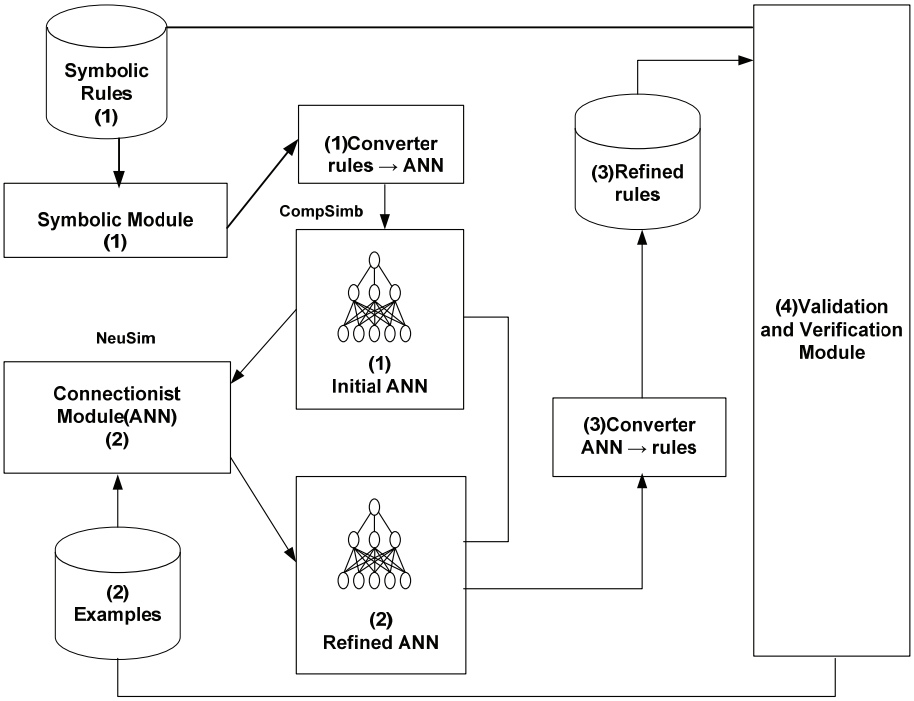



Fig. 1. Stages of a NSHS.

*1) Insertion*

In this stage, the knowledge extracted from a human expert is symbolically represented (Symbolic Module). This symbolic representation is in form of IF…THEN rules. Subsequently, it is converted to an Artificial Neural Network (ANN) named "initial" ANN. This stage is known as "rules compilation in a neural network" (Cruz, 2004).

*2) Refinement*
In this stage, a module that receives the initial ANN is implemented, which is subjected to a learning process starting from a base of examples. This module is named connectionist module; at the end of this stage, an artificial neural network is obtained, which is named "refined" ANN.
*3) Extraction*
In this stage, the extraction of the knowledge contained in the refined neural network is done. At the end of the module, symbolic rules called refined rules are obtained. This process is carried out due to the necessity to interpret and to evaluate such knowledge.
*4) Verification and validation*
In this stage, the coherence of extracted knowledge in symbolic rules is verified. Just later, tests to analyze the operation of the system as a whole are done.

## 4. Object of Knowledge Base Verification and Validation

The extracted knowledge of a NSHS is represented in production rules and stores the accessible experiences for the system. Different relations exist in the RB, for example: the conclusions of a rule can act as conditions for other rules and different rules can share common conditions. The production rules describe an IF-THEN relationship of the form $CC \Rightarrow CA$. Where CC is a collection of conditions, CA is a collection of actions or conclusions and the "$\Rightarrow$"symbol acts as a logical implication. The propositions CC can be joined by $\wedge / \vee$ that represent the logical connectives AND/OR respectively. Propositions CA can be joined only by the connective $\wedge$. A negative proposition $\neg p$ in CC is true if $p$ does not exist in the work memory. A negative proposition in CA causes elimination of $p$ in work memory.

### 4.1 Knowledge Verification Aspects

The main goal of verification is to obtain the consistent, complete and correct system. Because of that, anomalies on KB must be detected (Ramaswamy et al., 1997; Ramirez & De Antonio, 2001; Tsai et al., 1999; Yang et al., 2003). An anomaly is referred to common fault patterns according to an analysis technique (Ramirez & De Antonio, 2007; Tsai et al., 1999). This KB can contain errors due to:

1) The existence of several human experts providing their experiences in the application field.
2) The inserted knowledge can not be represented properly because of communication problems between human experts and the knowledge engineer.
3) The information may be missed during knowledge insertion due to matching of same one to the neural network,
4) The base of examples might be redundant due to a bad selection of samples.
5) Information may be missed or gained during the integration process of numerical and symbolic knowledge (Cruz et al., 2006; Santos, 1998; Villanueva et al., 2006).

It is necessary to consider some verification principles in order to make a suitable definition of the anomaly types that can be found into an extracted knowledge base of a NSHS.

a) Anomalies are defined according to the declarative meaning of the KB, instead of any other procedural meaning.
b) Anomalies are detected by means of analyzing the KB syntax, and should be semantically understood as well.
c) Anomalies are considered as symptoms of possible errors into the KB. Not all anomalies are errors.
d) Anomaly detection methods are just applied to the KB of the KBS and consider some properties of the inference engine.
e) The syntax and semantics of the anomalies should be defined in terms of syntax and semantics of the knowledge representation language used to expressed the KB.

The anomalies that can be found in a KB are shown in figure 2.

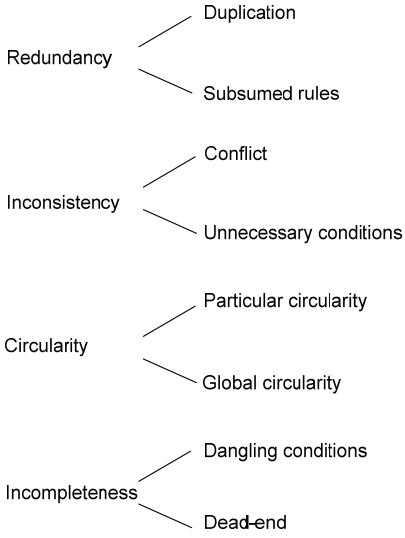



Fig. 2. Anomalies that can be found in a NSHS.

*Redundancy:* It occurs when there are unnecessary rules in a rule base. Redundant rules increase the size of the rule base and may cause additional inferences. There are two kinds of redundancies:

a) Duplication (equivalent rules). If $R1: a \wedge b \rightarrow c$ and $R2: b \wedge a \rightarrow c$, R1 and R2 are totally equivalents.

b) Subsumed rules (rule contained in another one). If $R1: a \wedge c \rightarrow b$ and $R2: a \rightarrow b$, R1 subsumes to R2.

*Inconsistency:* It occurs in conflictive facts. Here two types of inconsistencies are approached:

a) Conflict. A set of rules are conflicting if contrary conclusions are derived under a certain condition. An example of rules in conflict is the following: $R_1 : a \rightarrow b$ and $R_2 : a \rightarrow \neg b$.

b) Unnecessary conditions. A pair of rules $R, R' \in \Re$ has an unnecessary condition if they have the same consequent and the antecedents are only different in that some of the

propositions in $R$ are negated in $R'$. An example of rules with unnecessary conditions: $R_1 : a \wedge b \rightarrow c$ and $R_2 : a \wedge \neg b \rightarrow c$.

*Circularity:* It occurs when several inference rules have circular dependency. Circularity can cause infinite reasoning that must be broken. The following are examples of circularity.

a) Particular circularity. If $R1 : a \rightarrow b$ and $R2 : b \rightarrow a$, a cycle is formed in pairs of rules.

b) Global circularity. If $R1 : a \rightarrow b$, $R2 : b \rightarrow c$, $R3 : c \rightarrow a$ then chaining of circular rules appears.

*Incompleteness*: It occurs when there are missing rules in a rule base. If $R1 : \rightarrow a$, $R2 : a \wedge c \rightarrow b$, $R3 : a \rightarrow d$ and $R4 : b \rightarrow$ Here we present two types of Incompleteness.

a) Rule with dangling conditions. It occurs if the condition will never be matched by some conclusion. In R2 the condition *c* is never matched with any conclusion of the rest of the rules, therefore *c* is a dangling condition.

b) Rule with dead-end. It appears when a conclusion is not a goal and is not used as condition in any other rule. In R3, *d* is never matched by any condition of the rest of the rules; therefore *d* is a conclusion with a dead-end.

### 4.2 Knowledge Validation Aspects

The validation process allows analyzing the quality of the KB and the possibility of obtaining right solutions to the problems of the domain (Knauf et al., 2002; Tsai et al., 1999). This process is done in order to evaluate a system during or at the end of the development process to establish if it satisfies the specified requirements.

An important feature of the KBS is that its specifications and requirements are dynamically changing. It is more difficult to develop the verification and validation in systems which requirements or some other elements are frequently changing than in systems with static characteristics. There are different validation approaches such as: reachability, reliability, safety, completeness, consistency, robustness and usability.

A NSHS is only accepted when is convincingly and meticulously verified to work according to expectations.

The general evaluation frame consists of the following steps:

a) A testing criterion must be established, as reachability, reliability, safety, completeness, consistency, robustness and usability.

b) Test cases (inputs) and awaited outputs according to the selected inputs must be generated.

c) A test method to exercise the software must be applied.

d) The outputs of the software must be evaluated.

Tests are in general an intense work and a process prone to faults. Difficulties arise from different test criterions, great input and output space and legal input case generation (Knauf et al., 2002; Nikolopoulos, 1997; Vermesan, 1998).

*Test criterion.* It defines the objective of comparing a system versus a specification. Different types of tests are defined according to different types of test criterions.

*Test case generation.* Proper test inputs can be specified according to the type of problem that we can solve. It is possible to use the literature of the domain to generate the inputs as test cases.

*Expected output generation.* An expected output consists of a response. It is possible to ask to the expert of the domain to predict the expected outputs and generate them. Furthermore, expected outputs can be generated according to explicit solution specifications, saved test cases or literature of the domain.
*Test method to exercise the software.* The cost of the test method not only depends on the cost of the test case generation and on the evaluation cost. It also depends on the fact that a valid result is not found.
*Evaluation of software outputs.* It consists on evaluating if the generated output set belongs to the expected outputs in the solution of the problem (Nikolopoulos, 1997).

## 5. Petri nets as a modeling tool

Petri nets came in the literature on 1962 with the phD thesis of Carl Adam Petri (Murata, 1989). Petri nets are mathematic and graphic modeling tools that can be used in several types of dynamic systems. Petri nets allow describing and studying information from concurrent, asynchronous, distributed, parallels, nondeterministic, and/or stochastic systems (Nikolopoulos, 1997; Murata, 1989; David et al., 2005).

### 5.1 Petri nets graphic representation

A Petri net is a particular type of directed graphic and it is represented graphically by bipartite graph (there are two types of nodes and the arcs just can connect nodes of different types).Two types of Petri nets nodes are named places and transitions. Places represent variables that define the system state and transitions represent the transformer of such variables. Places are represented by circles, transitions by bars and marks are represented by a point into the circle which defines the place that contains it. Places and transitions are connected by directed arcs (Murata, 1989; Wu et al., 2000). The figure 3 shows the graphic representation of the Petri net elements.

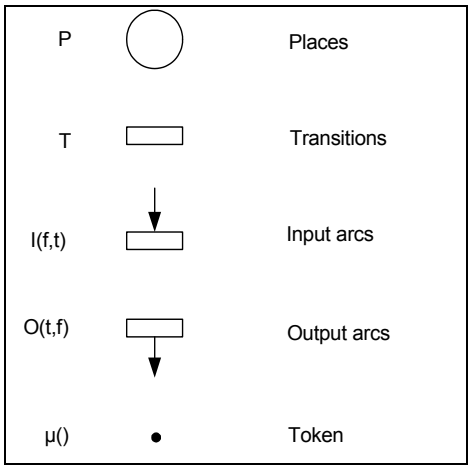



Fig. 3. Graphic Representation of the Petri net elements.

Figure 4 shows a Petri net with places: P= {P1, P2, P3, P4} and transitions: T = {T1, T2, T3, T4}.

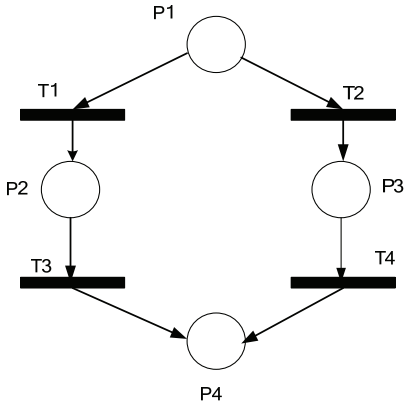



Fig. 4. Places, transitions and arcs of a Petri net.

An arc directed from a place $P_i$ to a transition $T_j$ defines an input place of the transition. Multiple inputs to the transition are indicated by multiple arcs from the input place to the transition. An output arc of a transition is indicated by an arc directed from the transition to the place. Multiple outputs are represented by multiple arcs (Murata, 1989).
Places can be used as "tokens" containers. The number of tokens contained in a place is named mark. Therefore, the net marking is defined as a column vector where the elements are the number of tokens contained in the places (David et al., 2005). Figure 5 shows the net marking: *M={m1, m2, m3, m4}*, in other words: $M=[1,1,0,0]^T$.

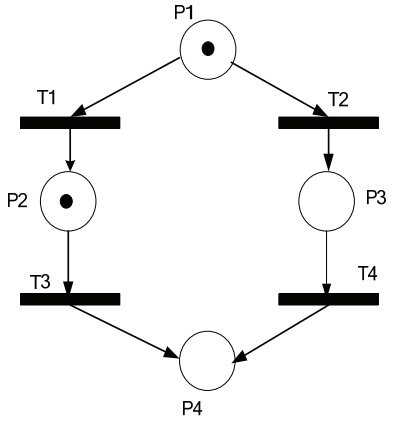



Fig. 5. Net marking.

Tokens can move inside the net, changing the state of the same one. In order to move the token, transitions must be fired (David et al., 2005).
A transition can be fired if it is enabled. A transition is enabled if into each one of its input places there is, at least, as many tokens as the arc weight that connects with them. A source

transition is always enabled. This type of transitions does not have input places (David et al., 2005).
Figure 6 shows three enabled and disabled transitions examples. Figure 6a presents t1 enabled due to its places P1 and P2 are in compliance with the enable conditions. In figure 6b, t2 is not enabled due to the place p5 has less tokens than the weight of the arc that connect it with the transition. In the figure 6c the source transition t3 is enabled.

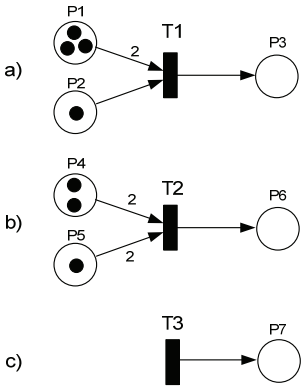



Fig. 6. Enabled and disabled transitions.

Firing of an enabled transition eliminates from each one of its input places as many tokens as the weight of each arc connecting such places with the transition. Also deposits in each output place as many tokens as the weight of each arc connecting them with the transition (David, et al. 2005; Murata, 1989).
Figure 7 shows a Petri net with two enabled transitions $t_1$ and $t_3$. Figure 8 shows the Petri net after the firing of the transition $t_1$.

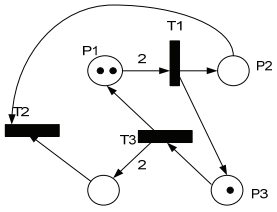



Fig. 7. Petri net with t1 and t3 enabled.

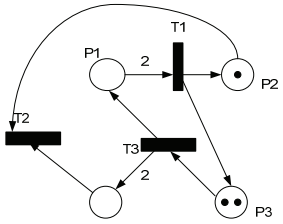



Fig. 8. Petri net after the firing of t1.

### 5.2 Petri nets formal definition

*Definition:* The structure of a Petri net is: $N = \{P, T, F, W, M_0\}$.

where:

1) $P$ is a set of places, $T$ is a set of transitions and $F$ is a set of flow relations. It means: $P \cap T = 0, P \cup T \neq 0, F \subseteq (PxT) \cup (TxP)$.
2) W is a mapping of $F \rightarrow \{1,2,3...\}$, it is a weight function.
3) $M_0$ is the initial marking $M_0 : P \rightarrow \{0,1,2,3...\}$, it is the initial marking.

### 5.3 Types of Petri nets

Nowadays, several types of Petri nets have been developed to be able of being applied to solve specific problems, and there are several classifications (David et al., 2005; Nazareth et al., 1991). Table 1 shows some types of Petri nets, as well as some of its characteristics. In this classification, Low level Petri net (Place/Transition) is emphasized as well as the most common extensions (Colored, Stochastic and Hierarchical).

| *Name of net* | *Type of net* | *Characteristics* |
|---|---|---|
| Pure Petri net | Classic net | Do not contain self cycling. Input places of t are not at the same time output places of T. |
| Ordinary Petri net | Classic net | If the weight of an arc is 1. |
| Finite Capacity Petri net | Classic net | Each place limits the number of tokens that it is able to contain. |
| Regular Petri net | Extended net with time | It has associated a determinate firing time |
| Stochastic Petri net | Extended net with time | It has associated a stochastic firing time. |
| Hierarchical Petri net | Extended net with hierarchy | A hierarchy of subnets is provided. |
| Colored Petri net | Extended net with color | It is very linked to its modeling language and allows working with tokens of different colors to represent values and types of data which the modeling system works with. |

Table 1. Petri nets clasification.

Classic Petri nets are also known as Low Level Petri nets, due to the modifications and restrictions made to the net are just arc connection conditions and the number of tokens that can store the places. However, Classic Petri net does not allow data and time modeling. Therefore, several extremely big and complex extensions for modeling real world have been proposed.

Extensions with time came up due to the necessity of describing the temporal behavior of the system. This type of nets can be divided into two classes: Deterministic time nets (regular Petri net) and Stochastic time net (Stochastic Petri net). The first one, include Petri nets that have associated a determined firing time in its transitions, arcs or places. The second one, include Petri nets that have associated a stochastic firing time in its transitions, arcs or places.

Hierarchical extensions have been used to manage the size problem that Petri nets face to model real systems. It is based on a restructuring mechanism of two or more processes that can be represented by subnets. In one level a simple description of the processes is done, while in another one it is done a detailed description of its behavior. This is the case of the client/server schemes where subnets are conveyed with each other using a place type graph.

Extensions with colors are known as colored Petri net. This type of nets combines the Classic Petri nets advantages and the high level programming languages, because of that this type of Petri nets allows the representation of different data types in the model by means of tokens that flow inside the net. In a colored Petri net the concept of token color is used. This allows having tokens of different colors, where each token color represents a piece of information. The model of a colored Petri net is more compact than the equivalent model in a classic Petri net (Chavarría & Li, 2006; David, et al. 2005; Nikolopoulos, 1997).

There are other Petri net extensions that have been developed, such as: hybrid and continuous Petri nets, as well as fuzzy Petri nets. Continuous Petri nets are a model where the mark number in the places is a decimal point number instead of an integer. On another hand, hybrid Petri nets are characterized by having a discrete part and a continuous part.

### 5.4 Enhanced Petri net model

Traditional Petri nets have inherent disadvantages. Some of these disadvantages are: deficiency of flexible descriptions for negative relations and necessity to formalize the original KB before beginning the verification process, such is the case of the example of a logical system or production systems, due to the difficulty of expressing logical disjunctions (He et al., 2003; Nazareth, 1993; Wu & Lee, 1997; Tsai et al., 1999; Yang et al., 1998). In order to overcome these problems an enhanced Petri net model is proposed by us.

*Definition: An enhanced Petri net* is a sextuple.

$$N = \{P, T, F, C, I_-, I_+\}, \quad F = F_b + F_r \tag{1}$$

Where:

1) $P$ is a set of places, $T$ is a set of transitions and $F$ contains the set of inhibitor or activator relations between *CC* and *CA*. Therefore, $F = F_b + F_r$.

2) $C$ means that for any $p \in P, C(p)$ is a collection of possible colors in $P$. For any $t \in T, C(t)$ is the collection of possible colors in the transition $T$.

3) $I_-$ and $I_+$ are negative and positive functions of $PxT$ respectively. For any $(p,t) \in PxT$. $I_-$ and $I_+$ are the previous and later transition matrices respectively.

The elements *P, T, $F_b$, $F_r$* are named predicate symbols (places), implications (transitions), activator arcs and inhibitor arcs, respectively. For a transition $t \in T$, a positive place $p_b \in P$ of $t$ is a place that connects to $t$ with an activator arc and presents a positive relation between $p_b$ and *t*. A negative place $p_r \in P$ of *t* is a place that connects to *t* with an inhibitor arc and presents a negative relation between $p_r$ and *t*. The elements of the EPN modeling are shown in the figure 9.

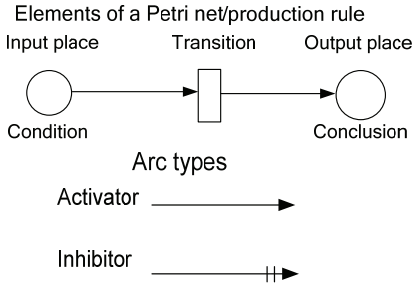



Fig. 9. Elements of the EPN modeling.

For the rule $R_i$, $t_i$ is its transition. The premises $C_1(r_i) \wedge C_2(r_i) \ldots \wedge C_n(r_i)$ are *t and the conclusion $A_1(r_i)$ is t*. The representation of the rule is shown in figure 10.

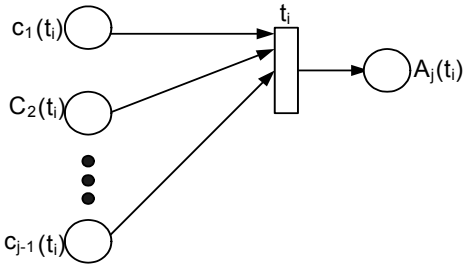



Fig. 10. EPN modeling Example.

As a PN, EPN can be mathematically represented by its incidence matrix which shows the interactions between places and transitions. In an incidence matrix $A_{mxn}$ the *n* columns represent places and the *m* rows represent transitions of N. The table 2 shows values that can have the incidence matrix of the enhanced Petri net.

| *Value* | *Means* |
|---|---|
| -1 | The place $p_j \in P$ is an input place to the transition $t_i \in T$. |
| 0 | There is not an arc connecting the place $p_j \in P$ with the transition $t_i \in T$ and vise versa. |
| 1 | The place $p_j \in P$ is an output place of the transition $t_i \in T$. |

Table 2. Values that can have the EPN incidence matrix.

## 6. Improper Knowledge Detection

In this section we propose a PN based mechanism to detect and eliminate structural errors of a KB extracted from a NSHS, which consists of three phases: *rule normalization*, *rule modeling* and *rule verification*. For the rule base verification, a static analysis of the EPN model is done by means of obtaining its incidence matrix.

*Rule normalization*

This step is done in order to simplify the rule base analysis. In this phase rules are translated into an *atomic form*:

$$R_i = C_1(r_i) \wedge C_2(r_i) \ldots \wedge C_n(r_i) \Rightarrow A_1(r_i) \tag{2}$$

It guarantees that each of its parts (CC/CA) cannot be decomposed into subparts. In an atomic rule the left condition only permits the conjunction of zero or more conditional clauses and just one element as action is permitted. The kinds of rules possible to normalize are:

1) $P_1(r_i) \wedge P_2(r_i) \wedge ... \wedge P_{j-1}(r_i) \rightarrow P_j(r_i) \wedge P_{j+1}(r_i) \wedge ... \wedge P_k(r_i)$

2) $P_1(r_i) \vee P_2(r_i) \vee ... \vee P_{j-1}(r_i) \rightarrow P_j(r_i) \wedge P_{j+1}(r_i) \wedge ... \wedge P_k(r_i)$

In order to obtain an atomic rule, first, each one of the elements of production rule is transformed into disjunctive form. An element in this form consists of one or more disjunctions, each one of these being a conjunction of one or more propositions. The rules are converted to disjunctive form using the distributive property of AND over OR, the idempotency and the contradiction. It means we have to use logical equivalences. The logical equivalences used in this work are shown in table 3.

| | |
|---|---|
| *Idempotency* | $A \wedge A \equiv A$<br>$A \vee A = A$ |
| *Complementary element* | $A \wedge \neg A \equiv F$<br>$A \vee \neg A \equiv V$ |
| *Distributive* | $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$<br>$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ |
| *Commutative* | $A \vee B \equiv B \vee A$<br>$A \wedge B \equiv B \wedge A$ |

Table 3. Logical equivalences.

Next, each subpart of the rule corresponding to each disjunction is separated. Also, a rule can have conjunctions in CA, which indicates it has multiple actions. In this way, separated rules are obtained corresponding to each action with the same set of premises that the original rule has.

In this chapter, we do not discuss the rules normalization with disjunctions in CA because the conclusion disjunctions do not make an explicit implication.

*Rule modeling to EPN*

In this step, rule mapping of the KB to EPN modeling is shown in the section Enhanced Petri net model. The rule modeling to EPN is made once the rules were normalized to their atomic form. Mathematical representation of the EPN is also obtained.

*Rule verification*

The following notations will be used during the rule verification process: *CC(i)* is a set of conditional clauses of *i-th* transition. *CA(i)* is a set of conclusion clauses of *i-th* transition. Also:

$C_{-1,i}$ = { $P_k$ | $P_k$ *is the place in the j-th column in matrix A such that* $A_{i,j} = -1$}

$C_{1,i}$ = { $P_k$ | $P_k$ *is the place in the j-th column in matrix A such that* $A_{i,j} = 1$}

$C_{0,i}$ = { $P_k$ | $P_k$ *is the place in the j-th column in matrix A such that* $A_{i,j} = 0$}

*Property 1. Redundancy.* For $t_i$ and $t_k$ $i \geq 1, k \leq m, i \neq k$; two transitions represent the rules *i* and *k* of the rule base, respectively, and execute the same action. There are two kinds of redundancy. If $t_i$ and $t_k$ satisfy any of the following conditions. $1) CC(i) \subset CC(k)$ or $2) CC(i) = CC(k)$ then the rules *i* and *k* are redundant. Duplication and subsumtion, are two kinds of redundancy.

Proof:

1) If $CC(i) \subset CC(k)$ then $|CC\ (i)| < |CC\ (k)|$. If the rows of $A_{mxn}$ represent transitions and the set of conditions represents the places, and such places $p_b \in CC(i)$ and $p_r \in CC(i)$ are equivalent in *CC(k)* then the rule *i* is subsumed in the rule k, i.e., $A_{i,j}$=1 y $A_{k,j}$=1, $1 \leq j \leq n$ and for the rest of places *j*, $A_{i,j} < A_{k,j}$.

2) $CC(i) = CC(k)$ indicates both rules *i* and *j* have the same conditions. In such a way, places $p_b \in CC(i)$ and $p_r \in CC(i)$ are equivalent in *CC (k)*, although the conditions are presented in different order, *i* and *j* are equivalent rules, i.e., $A_{i,j} = A_{k,j}$. for all *j*, $1 \leq j \leq n$.

*Property 2. Inconsistency.* For $t_i$ and $t_k$ $i \geq 1, k \leq m, i \neq k$; two transitions represent the rules *i* and *k* of the rule base, respectively. If $t_i$ and $t_k$ satisfy $CC(i) = CC(k)$ and $CA(i) \neq CA(k)$ then rules *i* and *k* are conflictive rules. If $t_i$ and $t_k$ execute the same action and either $p_b \in CC(i)$ or $p_r \in CC(i)$ are not equivalent in *CC(k)*, then that condition is necessary in $t_i$ and $t_k$.

Proof:

1) If $A_{i,j} = A_{k,j}$. For all $j$ $1 \leq j \leq n$ in $C_{-1,k}$ and $C_{-1,i,}$ and the places $p_b \in CC(i)$ and $p_r \in CC\ (i)$ are equivalent in *CC(k)*, but $C_{1,k} \neq C_{1,i}$.

2) If $A_{i,j} = A_{k,j}$. For all $j$ $1 \leq j \leq n$ in $C_{-1,k}$ and $C_{-1,i}$ and the places $p_b \in CC(i)$ and $p_r \in CC(i)$ are equivalent in *CC(k)* and $C_{1,k} = C_{1,i}$. However, $p_b \in CA(i)$ and $p_r \in CA(i)$ are different in *CA(k)*.

3) If $A_{i,j} = A_{k,j}$. For all $j$ $1 \leq j \leq n$ in $C_{-1,k}$ *y* $C_{-1,i,}$ and someone places $p_b \in CC(i)$ and $p_r \in CC(i)$ are not equivalent in *CC(k)*.

*Property 3. Circularity.* For $t_i, t_k, \ldots, t_m$, $i \geq 1, k \leq m, i \neq k, \ldots, \neq m$; *MT*:* memory work. All the transitions add their respective places to *MT** (if they were not in). In such a way if $p_b \in MT\ *$ or $p_r \in MT\ *$ and if it is deduced again in some transitions, then the place *p* causes circularity.

Proof:

1) $t_i$ and $t_k$ are two transitions represent the rules *i,k* of the rule base. $MT^* = \{ \{p_b \mid p_r\} \in p \mid p$ is the *j-th* column of *A* such that $A_{ij} = 1\}$ and $MT^* = \{ \{p_b \mid p_r\} \in p \mid p$ is the *j-th* column of *A* such that $A_{ij} = -1\}$. It checks $t_k$, $\{A_{k,j} = 1\} \in MT\ *$. Then there is contradiction in the rules *k* and *i*, and we consider particular circularity.

2) $t_i$,$t_k$,…,$t_m$ are transitions representing the rules *i,k,…,m* of the RB. *MT**= $\{ \{p_b \mid p_r\} \in p \mid p$ is the *j-th* column of *A* such that $A_{ij} = 1\}$ and *MT**= $\{ \{p_b \mid p_r\} \in p \mid p$ is the *j-th* column of *A* such that $A_{ij} = -1\}$. It checks in $t_{m,,}$ $\{A_{k,j} = 1\} \in MT*$. Then there is global circularity.

*Property 4. Incompleteness.* $t_i$ y $t_k$, $i \geq 1, k \leq m, i \neq k$ are two transitions representing the rules *i* and *k* of the rule base, respectively; $P_j$ is in wherever place of the transitions matrix.

Proof:

1) $\exists P_j \in C_{-1,i}$ and $\{p_b \mid p_r\} \in p_j$ for some *i*, such that $p_j \notin C_{-1,k}$ for all *k*. $P_i$ is the condition of some rule, but it is not the conclusion of any rule. Then the corresponding rule has a dangling condition $P_j$.

2) $\exists P_j \in C_{1,i}$ and $\{p_b \mid p_r\} \in p_j$ for some *i*, such that $p_j \notin C_{-1,k}$ for all *k*. $P_j$ is the condition of some rule, but it is not matched as condition of some other rule. Besides, $p_j$ is not a goal. The corresponding rule of $p_j$ is a rule with dead-end.

## 7. Knowledge Validation

In this section, we propose a mechanism to analyze the resultant rules from the verification process of a NSHS. This mechanism presents a multiple color scheme in the enhanced Petri net (see subsection Enhanced Petri net model). The reachability of the production system is probed on the basis of the dynamic logic inference made on the enhanced Petri net. The method consists of three phases: *rule modeling to EPN, dynamic modeling of the EPN, evaluation of results.*

*Rule modeling to EPN*

This step is done in order to generate the enhanced Petri net as shown in section Enhanced Petri net model. From which the verified knowledge base is received.

First, The KB is mapped to the EPN model and the mathematical representation of the same one is obtained. Then, the *transposed incidence matrix* is generated, and it is represented as $A^T$.

The places of the enhanced Petri net model are classified into two categories: $P = \{P_C, P_R\}$. For analysis conveniences, $P_C$ is divided into three sub-sets, $P_C = \{P_{CE}, P_{CI}, P_{CG}\}$. $P_{CE}$ is the places collection which can obtain information through inputs by the user. $P_{CI}$ is the places collection produced in the inference process and $P_{CG}$ is the places collection being the conclusion of the system. On the other hand, $P_R$ is the collection of transition places used by the model to avoid firing the same transition once the system has already fired it. It is due to the initial and deduced facts (in the inference process) are kept when the transition fire and can be used by the system has already fired it multiple rules. The reason is that the input places of each transition were held as output places of the same transition (see figure 11).

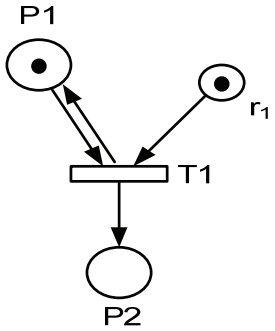



Fig. 11. Representation of the places: PCE= {P1} and PR={r1}.

*Closed world assumption.* This Enhanced Petri net model works under closed world assumption, which says that if a fact is unknown, any query about it is falsified. In fact, it assumes that all positive information has been specified. Any other fact not specified is assumed as false. The negation acts as if some additional rules are added to insert all the negative information when the NSHS is consulted (Wu & Lee, 1997).

*Unknown and know facts.* The known facts act as input and output of the rules. The Petri net model identifies them as tokens. When the transition fires, the model deletes a token that represents an input fact from some positive input place of the transition. On the other hand, when the rule fires and it has negative propositions in the right side of a rule, it falsifies the existence of the token in that place. In order to preserve the known facts, the model must preserve the negative propositions on the left side of the rule if these exist (Wu & Lee, 1997; Wu et al., 2005).

*Refraction.* The known facts reside in the work memory after rules fire. In order to do that, the Petri net model was modified to Enhanced Petri Net, attaching input places of a transition as well as output places of the same one (Nazareth, 1993; Wu & Lee, 1997; Wu et al., 2005)

*Dynamic modeling of the EPN*

This step is done in order to model dynamically the EPN. The developed validation method uses negation in reasoning of Close World Assumption and an initial marking from facts known by the user, which will be incorporated to the system. It uses the transposed matrix, composed by the input and output places of the transitions. The method incorporates matrixes and a reachability problem studied by means of an equation matrix set.

This modeling is made from goals and facts known by the user.

The facts known by the user act as inputs and outputs from rules. The Petri net model identifies them as tokens in the $P_{CE}$ places. The color set: $D = \{b, r_d, r_f, f\}$ is used to the marking of the EPN and the possible tokens in places are: $b, r_d, r_f$ and $f$. Color $b$ means that the clause or conclusion represented by the place is true. Colors $r_d$ and $r_f$ mean that the clause or conclusion represented by the place is *deduced false* and *defaulted false*, respectively. Tokens in $P_R$ have the color $f$ which means the rule has been already fired or not. Formally, the marking of the EPN is an indexed vector with respect to the places, which gives to each place $p$ a defined *Multi-Set* (MS). The *initial marking* of the EPN is defined by using a formal notation based on sums as follows: $M_0 = p_1(1b+1f) + p_2(1b+1f),$.

The marking $M_0(P_R) = [1f]$, means that exist no rules fired initially.

*Goals of the system* are also provided by the user and they represent the $P_{CG}$ places. They act as conclusions of the system and the goals expected by the user from the dynamic modeling. Rules forward chaining is used to the dynamic modeling of rules. Dynamic equations control dynamic behavior of the system. A *t* transition could fire if it is enabled with a color *(X(t))* under a marking $\mu$. If *t* is fired then the $\mu$ marking is changed to the reachable marking $\mu' = \mu'_b \cup \mu'_r$ by means of:

$$\begin{aligned}
&\mu'_b(p) = \mu_b(p) - I_{b-}(p,t)(X(t)) + I_{b+}(p,t)(X(t)) \\
&\mu'_{r_d}(p) = \mu_{r_d}(p) - I_{r-}(p,t)(X(t)) + I_{r+}(p,t)(X(t)) \\
&\mu'_{r_f}(p) = \mu_{r_f}(p) - I_{b+}(p,t)(X(t)) - I_{r+}(p,t)(X(t)) \\
&- I_{b-}(p,t)(X(t)) - I_{r-}(p,t)(X(t))
\end{aligned} \tag{3}$$

All $p \in P$, where the operators "+" and "-" are operations of "addition" and "subtraction" on multi-set, respectively. Where *(X(t))* is considered a fire element and $\mu'$ the following marking of $\mu$. The transition firing model in the validation phase is a dynamic logic inference and an explanation logic reasoning.

Figure 12(a), presents the modeled Petri net that simulates the behavior of the rule: $R1: A \rightarrow B$. In the model, an additional node named rule place is provided.

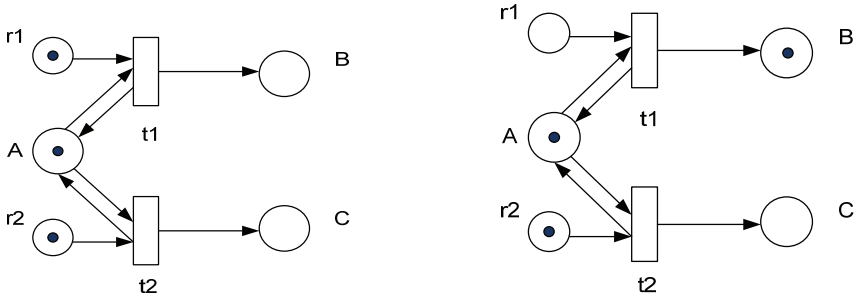



Fig. 12. Modeling of the transition t1: (a) Initial marking, (b) Firing of *t1*

Figure 12(b) expresses that the condition A is true, and then the transition of the corresponding rule *r* is enabled because "the rule place" has a token. It expresses that the conclusion B is true after the transition firing and that the condition *A* keeps the token even after the transition firing.

Finally, the inference mechanism is stopped when it cannot fire any other transition (there are no enabled transitions) or when some deduced fact is a goal proposition. The obtained marking is known as the final marking of the chaining and it represents the reachability of the net from initial marking to final marking. In each test, the system obtains the following data: initial facts used in the net, visited rules, fired rules, deduced facts and finally, the

obtained results notification (valid or invalid results). If the final marking contains some place considered as a goal then this test is labeled as valid result. Otherwise, the test is labeled as an invalid result.

*Evaluation of results*

This step is done in order to evaluate all the performed tests with different initial marking. It allows the evaluation of the reliability of the obtained results with such initial markings.

The system is considered reliable when valid results are obtained from applied test input. If no valid results are obtained in any performed test, the quality of the input test is analyzed. In case of having applied properly test inputs, we deduce the system contains inconsistent, partially erroneous, or incomplete knowledge. In reason to, it is convenient to perform the verification process again.

## 8. Experimental results

Based on our approach addressed in previous sections, the validation and verification example 1 of a KB used by a NSHS is showed.

*Example 1.* We propose a KB extracted from an example base which contains information about 24 patients who were examined to diagnose if they should use contact lenses according to some of their symptoms. The following data are the principal features of the example base.

**Attribute information:**

**Age of the patient:** Young, Pre-presbyopic, Presbyopic
**Tear production rate:** Reduced, Normal
**Astigmatic:** No, Yes
**Spectacle prescription:** Myope, Hypermetrope

**Classes**: Hard Contact, Soft Contact, No Contact

**Distribution of the classes:**

**Hard Contact:** 4 instances
**Soft Contact:** 5 instances
**No Contact:** 15 instances

The following KB was extracted from the example base mentioned above. In this case, the set of rules is normalized. It can be easily mapped into an EPN.

*R1:* ***If*** *-(Tear(Reduced))* ***Then*** *Hard_Contact*
*R2:* ***If*** *Astigmatic* ***Then*** *Hard_Contact*
*R3:* ***If*** *-(Spectacle(Hypermetrope)) And -(Age(Are- presbyotopic)) And -(Age(Presyopic))* ***Then*** *Hard_Contact*
*R4:* ***If*** *-(Tear(Tormal))* ***Then*** *Hard_Contact*
*R5:* ***If*** *-(Astigmatic) And -(Tear(Reduced))* ***Then*** *Soft_Contact*
*R6:* ***If*** *Spectacle(Hypermetrope) And Age(Young) And Age(Are-presbyotopic)* ***Then*** *Soft_Contact*
*R7:* ***If*** *-(Age(Presyopic)) And -(Spectacle(Myope))* ***Then*** *Soft_Contact*
*R8:* ***If*** *Tear(Tormal) And NewUnit1* ***Then*** *Soft_Contact*
*R9:* ***If*** *Tear(Reduced)* ***Then*** *No_Contact*
*R10:* ***If*** *-(Age(Young))* ***Then*** *No_Contact*
*R11:* ***If*** *-(NewUnit1)* ***Then*** *No_Contact*
*R12:* ***If*** *Astigmatic* ***Then*** *No_Contact*
*R13:* ***If*** *Spectacle(Hypermetrope)* ***Then*** *No_Contact*

*R14:* ***If*** *Age(Presyopic) And Age(Are-presbyotopic) And -(Spectacle(Myope)) And Tear(Tormal)* ***Then*** *No_Contact*
*R15:* ***If*** *Astigmatic And Age(Are-presbyotopic) And Tear(Reduced) And Spectacle(Hypermetrope)* ***Then*** *NewUnit1*
*R16:* ***If*** *-(Spectacle(Myope))* ***Then*** *NewUnit1*
*R17:* ***If*** *-(Tear(Tormal))* ***Then*** *NewUnit1*

### 8.1 Results of the verification process

For this analysis, the propositions: *Hard_Contact, Soft_Contact and No_Contac* were used as goals. First of all, the normalization process was made to the KB (For this case, the KB was already normalized).
Next, the mapping of KB for EPN modeling and its incidence matrix was made. Finally, the verification process was done. The table 4 shows the obtained results from verification process applied to the previous KB.

| **Rule** | **Evaluation** |
|---|---|
| 2 | Conflict with R12, dangling conditions |
| 4 | Conflict with R17, dangling conditions |
| 12 | Conflict with R2 |
| 17 | Conflict with R4, dangling conditions |
| 1,3,5,6,7,8,9,10, 11,13,14,15,16 | Dangling conditions |

Table 4. Results from the verification process.

In this process we can see that incomplete knowledge due to dangling conditions was detected in all rules (conditions will not be matched with any conclusion).
The users make elimination of rules according to their requirements, except for duplicated rules, which are eliminated automatically. The new rule base contains less rules than the original one, it is free of errors and is better structured.

### 8.2 Results of the validation process

For this analysis, each of the test used different initial facts. *Hard_Contact, Soft_Contact* and *No_Contact* were used as goal propositions. Table 5 shows the obtained results from validation process applied to the verified KB.

| *Tests* |
|---|
| Test:1<br>*Initial facts:* Spectacle (Hypermetrope), Age(Young), Age(Are-presbyotopic),<br>*Visited rules:*1;<br>*Fired rules:*1,<br>*Deduced facts:*Hard_Contact,<br>*Validate:*YES->goal: Hard_Contact |
| Test:2 |

| |
|---|
| *Initial facts:* Spectacle(Hypermetrope), Age(Young), Age(Are- presbyotopic), Tear(Reduced),<br>*Visited rules:*1,2,3,4;<br>*Fired rules:*4,<br>*Deduced facts:*Hard_Contact,<br>*Validate:*YES->goal: Hard_Contact |
| Test:3<br>*Initial facts:* Spectacle(Hypermetrope), Age(Young), Age(Are-presbyotopic), Tear(Reduced), Tear(Tormal),<br>*Visited rules:*1,2,3,4,5,6;<br>*Fired rules*:6,<br>*Deduced facts*:Soft_Contact,<br>*Validate:*YES->goal: Soft_Contact |

Table 5. Results from the validation process.

## 9. Summary

In this chapter an Enhanced Petri Net model is presented for the verification and validation of Neural Symbolic Hybrid Systems. KBs of the NSHS are expressed in production rules based on propositional logic. Such KBs can involve negative information and contain disjunctions in their production rules. These aspects can be expressed by using our EPN method, but not in a traditional Petri net. It is due to in a traditional Petri net some authors create a new place to represent a negative proposition, and use a new place to represent a set of disjunctions. Our method reduces the processing time in validation and verification processes.

The verification module allowed us to formalize the checking concepts of the KB using a formal and conceptual frame to the specification of such checking. This formalization improves the understanding of verification in reason to the group of anomalies that might arise and the algorithms that can be used to detect them.

The considered anomalies have to be with four fundamental properties of verification, which are: redundancy, circularity, inconsistency, incompleteness. Besides, this method shows to the knowledge engineer the rule group potentially in conflict to its previous analysis.

Our verification method is based on incidence matrix of an EPN. This method has the advantage that it is independent from the initial marking of the net.

According to the developed tests, the extracted knowledge in form of production rules and any other KB of a Rule Based System, can contain the errors presented in the anomaly definitions.

The validation method is based on a reachability analysis of the enhanced Petri net. This analysis is executed from test cases and expected goals with such selected inputs. Important aspects of RBS such as facts conservation, refraction and closed world assumption, can be easily modelled from the color scheme here presented.

Cases of test are considered according to: complexity of the KB, relations between evaluated and deduced propositions following an inference process. Besides, cases of test are considered with respect to the expected goals.

As future work we consider to include verification and validation of production rules with uncertainty factors. In order to do this, it would be necessary to redefine other verification

definitions. This adaptation leads us to consider extending the redundancy and inconsistency definitions in order to detect such anomalies in a set of rules. Another promising line of work we are tackling is an extension of the proposed method that permits detecting incompleteness using submarking reachability and simulation.

## 10. References


Chavarria, L., & Li X. (2006). Structural error verification in active rule based systems using Petri nets. *Proceedings of the Fifth Mexican International Conference on Artificial Intelligence (MICAI'06)*, pp. 12 - 21, ISBN: 0-7695-2722-1, Apizaco, Mexico, Nov 2006, IEEE Computer Society.

Cloete, I. & Zurada, J. (2000). *Knowledge-Based Neurocomputing (Eds.)*. MIT Press, ISBN: 0-262-03274-0, Cambridge, MA.

Cruz, V. (2004). Neural-Symbolic Hybrid System to refine the knowledge of an Artificial Vision System. Master Thesis, Cenidet, Cuernavaca, Morelos, Mexico.

Cruz, V.; Reyes, G.; Vergara, O.; Perez, J. & Montes, A. (2005). Compilation of symbolic knowledge and integration with numeric knowledge using hybrid systems. *Proc. 4th Mexican International Conference on Artificial Intelligence (MICAI'05)*, pp. 11 – 20, ISBN: 3-540-29896-7, Monterrey, Nuevo León, México, Nov 2005, Springer Lecture Notes in Computer Science.

Cruz, V.; Reyes, G.; Vergara, O. & Pinto, R. (2006). A Combined Representation to Refine the Knowledge Using a Neuro-Symbolic Hybrid System applied in a Problem of Apple Classification. *Proc. 16th IEEE Int. Conf. on Electronics, Communications and Computers (CONIELECOMP-06)*, pp. 30-36, ISBN: 0-7695-2505-9, Puebla, México, Feb 2006, IEEE Computer Society.

David, R. & Alla, H. (2005). *Discrete, continuous and Hybrid Petri nets*, Springer- Verlang, ISBN: 978-3-540-22480-8, Berlin Heidelberg.

He, X.; Chu, W. & Yang, H. (2003). A new Approach to Verify Rule-Based System Using Petri Nets. *Information and Software Technology*, Vol.45, No. 10, 663-669, ISSN 0950-5849.

Knauf, R.; Gonzalez, J. & Abel, T. (2002). A Framework for Validation of Rule-Based Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, part b: cybernetics, vol. 32, no. 3, (Jun 2002), pp. 281-295, ISSN: 1083-4419.

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, pp. 541-580, ISSN: 0018-9219, Illinois, Chicago, USA, April 1989.

Nazareth, D. (1993). Investigating the Applicability of Petri Nets for Rule-Based Systems Verification. *IEEE Transactions on Knowledge and Data Engineering*, vol. 4, no. 3, (June 1993), pp. 402-415, ISSN: 1041-4347.

Nazareth, D. & Kennedy M. (1991). Verification of Rule-Based Knowledge Using Directed Graphs. *Knowledge Acquisition*, vol. 3, 339–360.

Negnevitsky, M. (2005). *Artificial Intelligence. A guide to Intelligent Systems*, Second Edition. ADDISON WESLEY, ISBN: 0-321-20466-2, University of Tasmania.

Nikolopoulos, C. (1997). *Expert Systems*. MARCEL DEKKER INC, ISBN: 0-8247-9927-5, Bradley University, Peoria Illinois.

Ramaswamy, M.; Sarkar, S. & Chen, Y. (1997). Using Directed Hypergraphs to Verify Rule-Based Expert Systems. *IEEE Transactions on Knowledge and Data Engineering,* vol. 9, no. 2, 221-237.

Ramirez, J. & De Antonio, A. (2001). Checking Integrity Constraints in Reasoning Systems based on Propositions and Relationships. *Proceedings of the 13th International Conference on Software Engineering and Knowledge Engineering, SEKE.01*, pp. 188-196, Buenos Aires, Argentina, June 2001.

Ramirez, J. & De Antonio, A. (2007). Checking the Consistency of a Hybrid Knowledge Base System. *Elsevier Science. Knowledge-Based Systems*, Volume 20, Issue 3, April 2007, 225-237, ISSN: 0950-7051.

Santos, F. (1998). INSS - Un Système Hybride Neuro-Symbolique pour l'Apprentissage Automatique Constructif, PhD Thesis, LEIBNIZ-IMAG, Grenoble – France.

Tsai, W.; Vishnuvajjala, R. & Zhang, D. (1999). Verification and Validation of Knowledge-Based Systems. *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, January/February 1999, 202-212, ISSN: 1041-4347.

Vermesan, A. (1998). Foundation and Application of Expert System Verification and Validation, In: *The Handbook of Applied Expert Systems*, Jay Liebowitz, Ed. CRC Press LLC, pp. 5.1–5.32, CRC Press, Inc. Boca Raton, ISBN:0849331064, FL, USA.

Villanueva, J.; Cruz, V.; Reyes, G. & Benítez, A. (2006). Extracting Refined Rules from Hybrid Neuro-Symbolic Systems. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'06)*, pp. 3021-3025, ISBN: 0-7803-9490-9, Vancouver, Canada, July 2006.

Wu, C. & Lee S. (1997). Enhanced High Level Petri Nets with Multiple Colors for Knowledge Verification/Validation of Rule-Based Expert Systems. *IEEE Trans. on Systems, Man, and Cybernetics*, Part B. vol 27, no. 5, October 1997, 760-773, ISSN: 1083-4419/97.

Wu, C., & Lee, S. (2000). A Token-Flow Paradigm for Verification of Rule-Based Expert Systems. *IEEE Trans. on Systems, Man, and Cybernetics*, Part B. vol. 30, no. 4, August 2000, 616-624, ISSN: 1083-4419/00.

Wu, Q.; Zhou, C.; Wu, J. & Wang, C. (2005). Study on Knowledge Base Verification Based on Petri Nets. *International Conference on Control and Automatization (ICCA2005)*, ISBN: 0-7803-9137-3/05, Budapest, Hungry.

Yang, S.; Lee, A.; Chu, W. & Yang, H. (1998). Rule Base Verification Using Petri Nets. in *Proccedings 22-nd Annual Int. Computer Software and Applications Conf. (COMPSAC-98)*, ISBN: 0-8186-8585-9, pp. 476-481, Vienna, Austria, august.

Yang, S.; & Tsai, J. & Chen, C. (2003). Fuzzy Rule Base System Verification Using High-Level Petri Nets. *IEEE transactions on knowledge and data engineering*, vol. 15, no. 2, March/April 2003, 457-473, ISSN: 1041-4347.

9

# A new Control Synthesis Approach of P-Time Petri Nets

Bonhomme Patrice
*University François Rabelais of Tours, Laboratoire d'Informatique,*
*64 avenue Jean Portalis, 37200 TOURS*
*France*

## 1. Introduction

Petri nets (PN) (David & Alla, 1994 ; Murata, 1989) are recognized as an appropriate tool for the modeling and analysis of asynchronous, discrete event systems with concurrency. Originally proposed as a causal model explicitly neglecting time, they have been extended and adapted in several ways in order to fulfill the requirements of specific application areas (Vand der Aalst, 1993 ; Diaz & Senac, 1994 ; Khansa*, et al.* 1996 ; Merlin & Faber, 1976 ; Roux & Déplanche, 2002).

In particular for systems whose functionalities are defined with respect to time (Berthomieu & Diaz, 1991 ; Bonhomme, 2006 ; Bonhomme *et al*., 2001 ; Bucci & Vivario, E., 1995 ; Calvez *et al*., 2004 ; Jiroveanu *et al*., 2006 ; Wang *et al*., 2000) and whose correctness can only be proved by taking time into consideration, PNs are extended with two time parameters representing the minimum and the maximum delay. For Merlin's model (Merlin & Faber, 1976), also called T-time Petri nets or time Petri nets (TPN on short), this interval of firability is associated with each transition of the model representing its static earliest and latest firing time. For P-time Petri nets (P-TPN) (Khansa*, et al.* 1996), each place is associated with a time interval representing the operation duration of a token in this place. To ensure the safe behavior of such systems there is a need for formal approach to model and analyze their correctness.

The idea behind the control concept is to ensure that the system satisfies a set of imposed specifications, its behavior is then restricted to an acceptable one. From a supervisory control point of view, it means that the system never enters a forbidden state or, being given a set of legal words (a set of firing sequence), no illegal word is generated. Inspired by the well-known event-feedback control of Ramadge and Wonham (Ramadge & Wonham, 1989), recently, (Guia *et al*., 2004) proposed a state-feedback control with event observer and initial macromarking (the distribution of the tokens is partially known). The safety specifications under consideration is the limitation of the weighted sum of markings in subsets of places of the studied PN, called generalized mutual exclusion constraints (GMEC). In their approach, they focus on T-timed Petri nets and developped a control scheme with a deadlock recovery procedure by means of observers.

Concerning time models many control techniques are based on linear algebra and thus, are restricted to subclasses of PNs (such as marked graphs, state machines, ...). Similarly, (Freedman, 1991) proposed the structural analysis of a subclasss of T-time Petri nets called Ω which are either deterministic or exhibiting a particular type of conflict called choice among alternatives. (Sathaye & Krogh, 1998) proposed an extension of T-time Petri nets called controlled time Petri nets (CtlTPNs) to model the dynamics of real-time discrete event systems. To fully represent the logical behavior of CtlTPNs control class graphs (CCGs) are also defined. Thanks to this graph (which is an extension of the state classes graph of (Berthomieu & Diaz, 1991) dealing with the control effects) a real-time supervisor, based on a nondeterministic logical supervisor for the CCG, is designed ensuring that the desired specifications are satisfied. However, the method is restricted to bounded CtlTPNs for which a finite CCG exists. More recently, an original approach (Wang *et al.*, 2007) based on PN unfoldings was proposed to enforce transitions deadline in TPN but only safe models are considered.
This paper introduces a new control approach based on the analysis of the P-time Petri Net model structure and more specifically on the set of feasible firing sequences of the underlying untimed Petri net. Furthermore, the proposed approach is not restricted to sub-classes or safe time Petri nets. So, thanks to the determination of an inequalities system generated for a possible evolution of the autonomous model considered, the performances evaluation and the determination of an associated control for a definite functioning mode for the time model are made possible. Thanks to the introduction of partial order on the execution of particular events the developed approach of control is more reactive. Indeed, it is proposed to not impose precedence constraints among operations which can occur concurrently, leading to a flexibility gain (on the operations) in the determination of a feasible control as the scheduling of particular events can be modified. It can be noticed that the idea of not considering the firing order into the timing constraints, for concurrent events, was proposed by (Lilius, 1999). Thanks to a new semantics for time PN, the application of partial-order theory, originally developed for untimed Petri nets, is made possible for TPN. However, its method is restricted to contact-free TPN.
Moreover, the inequalities system is written just once, (it can be done for symbolic values – the inequalities are then written in terms of all the parameters, symbols other than numerics) so, changing the timing constraints do not modify its form and its non-emptiness can also be used to answer questions about reachability of particular markings.
The paper is organized as follows: an informal discussion of the introduction of time in Petri nets is realized in the next section. A formal definition of P-time Petri nets is given in the third one. Section four recalled the material required for the approach, originally devoted to an exhaustive simulation purpose. The proposed approach is then presented in section five and an illustrative example consisting in the supervision of a control distributed in several Programmable Logic Controllers is proposed in the sixth one.
Finally, in the last section some conclusions and future work are presented

## 2. Petri nets and time

It is assumed, in the following, that the reader is familiar with Petri nets. If it is not the case, please refer to (Murata, 1989) for the basic definitions and terms. There exists several

extensions of Petri nets dealing with time in the literature, each one being dependent on the application considered and are aimed at expressing different types of constraints.
The introduction of timing issues in Petri nets drives to several extended models. The first extension of PN with time was introduced by Ramchandani (Ramchandani, 1973). In his model, called T-timed PN, a delay is associated with each transition representing the duration of an operation. They have been extensively used in a performances evaluation context (Zuberek, 2000). Another timed model proposed by Sifakis (Sifakis, 1977), called p-timed PN associates to each place of the net a delay representing the sojourn time of a token in this place. It was demonstrated that both models are expressively equivalent.
In the timed models a token can wait indefinitely in a place. Indeed, a transition is not forced to fire unless the decision to fire it is explicitely made. Consequently, they are not suitable for the modelling of time critical systems, for which a minimum and a maximum sojourn time are imposed.
Among the time models, the Merlin's model is the most commonly used but others time models, dealing with operation durations specified as intervals, can be found in the literature.
Van der Aalst (Van der Aalst, 1993) proposed the interval timed coloured Petri net (ITCPN), its main feature is the time mechanism associated with each token of the net. Indeed, in this model a timestamp is attached to every token. This timestamp indicates the instant at which a token becomes available. The enabling time of a transition is the maximum timestamp of the tokens to be consumed. Transitions fire as soon as possible and the transition with the smallest enabling time will fire first. Firing is an atomic action which produced tokens with a timestamp of at least the firing time. The difference between the firing time and the timestamp of such a produced token is called the firing delay and it is specified by an interval.

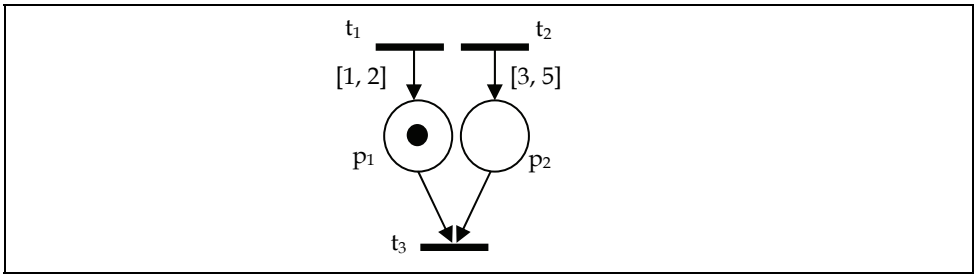



Fig. 1. Portion of ITCPN

For instance, consider the portion of ITCPN above, if transition $t_1$ is fired at time 0, its firing creates the token in place $p_1$ with a timestamp within the interval [1, 2]. So, the token contained in place $p_1$, which input the synchronization transition $t_3$, can wait indefinitely because this transition will be forced to fire only when the token created by the firing of transition $t_2$ will become available (at the time: (firing instant of $t_2$) + $x$, with $x \in [3, 5]$). To avoid the previous situation, a transformation can be realized and it drives to the representation shown in Fig. 2.

Furthermore, it must be specified that when a conflicting situation occurs the priority is given to the firing of transition $t_3$, but this particular policy contradicts the fact that the transition with the smallest enabling time should fire first.

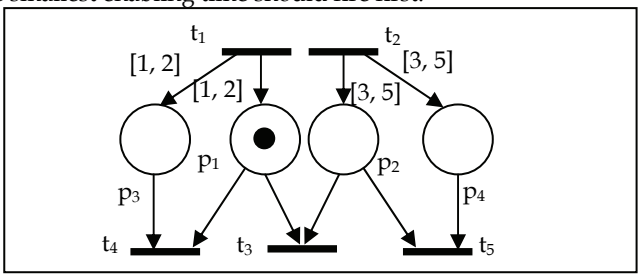



Fig. 2. Transformation of ITCPN.

Time stream Petri nets (TSPN) (Diaz & Senac, 1994) was presented as an extension of the Merlin's model aiming at specifying synchronization constraints in distributed asynchronous multimedia systems and applications. Informally speaking, in this model a time validity interval $[\alpha_i, \beta_i]$ is associated to each arc linking a place $p$ to a transition $t$. This interval means that if a token arrives in place $p$ at time $\tau$, transition $t$ is force to fire in the interval $[\tau + \alpha_i, \tau + \beta_i]$.

The main feature of this model is the representation of several synchronization mechanisms. Indeed, a complete set of firing rules is proposed to accurately enforce actual synchronization policies between different and related multimedia streams. The "pure-and" rule allows to represent satisfactorily the synchronization mechanism but the different semantics associated with the multiple enabledness of transitions drive to a firing rule which can be sometimes hard to handle.

Consider the following portion of TSPN where the tokens in place $p_1$ and $p_2$ are arrived at time 0. The enabledness intervals of transition $t_1$ associated with the three synchronization modes, strong-or, weak-and and pure-and, are also represented.

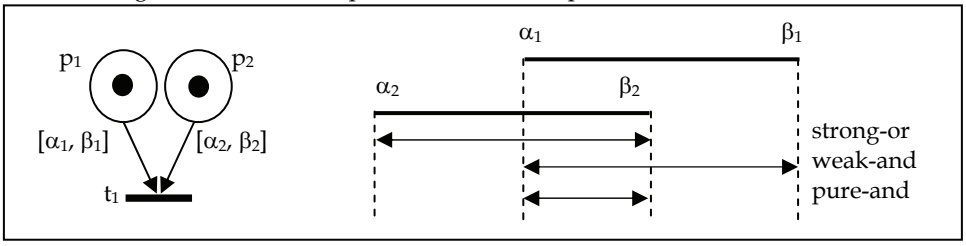



Fig. 3. Time stream Petri net and synchronization.

(Ghezzi *et al*., 1991) proposed a high-level model called Entity-Relation net (ERN), as the one developed by Van der Aalst, each token carries a set of informations and each transition is associated with relations. These relations allow to select the different tokens involved in the enabledness of particular transitions and they also indicate the type of the tokens which can be created (depending on the consumed ones).

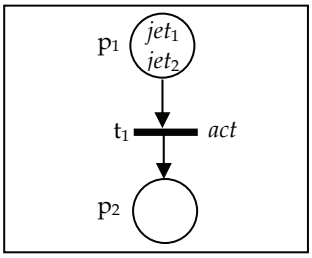



Fig. 4. Portion of ER net.

For instance, in Fig. 4, place $p_1$ contains two tokens $jet_1$ = {<$x$, 1>, <$y$, 1>} and $jet_2$ = {<$x$, 0>, <$y$, 1>}, the relation attached to transition $t_1$ is $act$ = {<$p_1$, $p_2$> | $p_1.x = p_1.y$ and $p_2.x = p_1.x$ and $p_2.y \in \{z \mid p_1.y \le z \le p_1.y + p_1.x\}$}. In the previous relation $p_1$ means any token in this place (in this case, either $jet_1$ or $jet_2$) and <$x$, 1> means $p_1.x = 1$. The firing of transition $t_1$ is represented by the expression <$p_1$, $t_1$, $p_2$> or equivalently when it is clear from the context <$p_1$, $p_2$>. The token $jet_1$ is satisfying $act$ while $jet_2$ is not, so, the token created by the firing of $t_1$ denoted as $p_2$ will verify $p_2.x = 1$ and $p_2.y \in [1, 2]$ and the token $jet_2$ will be considered as dead (thus, it can be removed from the net).

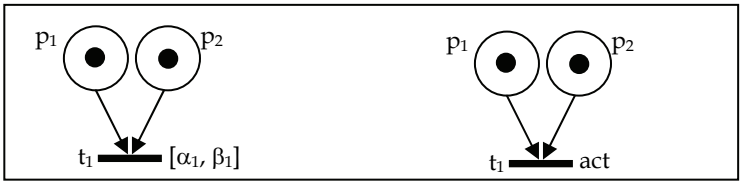



Fig. 5. Merlin's model / Time ER model.

In this model, the time factor can be introduced thanks to a timestamp (denoted as *chronos*) associated to each token of the net and representing its creation instant – the time ER model (TER) is then obtained. Originally, a weak time semantic is associated with this model (i.e. transitions are not force to fire) but the authors proposed a non trivial transformation rule allowing to consider a strong time semantic.
For instance, consider the two portions of PN (Merlin's model / time ER model) depicted on Fig. 5, both synchronization transitions will have the same behaviour if the transition $t_1$ of the TER model is associated with the following relation act:

$$act = \{x \mid \max\{p_1.chronos, p_2.chronos\} + \alpha_1 \le x \le \max\{p_1.chronos, p_2.chronos\} + \beta_1\}.$$

In the next section, the chosen modeling tool, the P-time Petri nets is presented

## 3. P-Time Petri nets

The formal definition of a P-TPN (Khansa *et al.*, 1996) is given by a pair < $Nr$; $I$ > where:

$Nr$ is a marked Petri Net (David & Alla, 1994)

$I: P \rightarrow (Q^+ \cup \{0\}) \times (Q^+ \cup \{\infty\})$

$p_i \rightarrow I_i = [a_i, b_i]$ with $0 \le a_i \le b_i$

Where:

***P***: the set of places of net *Nr*,
$\boldsymbol{Q}^+$: the set of positive rational numbers.
$I_i$ defines the static interval of the operation duration of a token in place $p_i$.

A token in place $p_i$ will be considered in the enabledness of the output transitions of this place if it has stayed for $a_i$ time units at least and $b_i$ at the most. Consequently, the token must leave $p_i$, at the latest, when its operation duration becomes $b_i$. After this duration $b_i$, the token will be "dead" and will no longer be considered in the enabledness of the transitions. Notice that: a dead token is not removed from the place, this token state indicates that a potential time violation has occurred. As the death of tokens generally occurs in places which are input places of a synchronization transition an algebraic approach using (min, max, +) algebra was proposed to check their correct behaviour for P-time marked graphs (Declerck & Alaoui, 2004).
The particularity of this model requires analysis techniques, allowing taking account efficiently of the various functionalities associated with the modeled system, as well as its time features. It leads ineluctably to the need for having formal methods ensuring the system control. Indeed, the policy consisting in firing a transition as soon as it becomes enabled is not always feasible and usually leads to a potential constraint violation.
It can be noticed that for P-TPN the contribution of each token present in the net must be taken into account because it must be prevented from dying whether it participates to the enabledness of a transition or not. In the Merlin's model the situation is not the same. Indeed, consider the following synchronization mechanism for the Merlin's model depicted on Fig. 6.

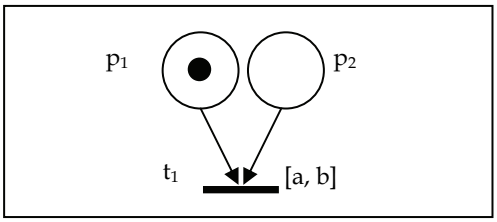



Fig. 6. Synchronization and T-time PN

The token in place $p_1$ can wait indefinitely and the local clock associated with transition $t_1$ is triggered when a token arrives in place $p_2$.
For a net *Nr* the conventional dot notation which can naturally be extended to set of nodes, will be used:

- $^\circ t$ ($t^\circ$) = the set of input (output) places of transition $t$.
- $^\circ p$ ($p^\circ$) = the set of input (output) transitions of place $p$.

The following section is devoted to the control approach, based on the evaluation of the firing condition.

## 4. Firing instant notion

***Definition 1*** (Boucheneb & Berthelot, 1993)*:* A fired transition denoted by $t_j$ will be associated with the $j^{th}$ firing instant (i.e. the firing sequence considered is $t_1t_2t_3..t_q$). A variable $x_i$ will represent the elapsed time between the $(i - 1)^{th}$ and the $i^{th}$ firing instant.

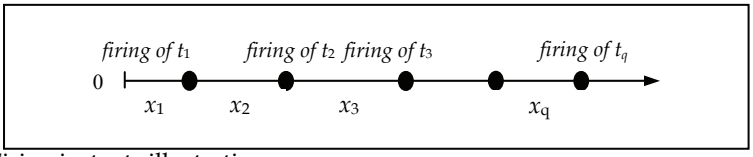



Fig. 7. Firing instants illustration

For instance on Fig. 7, $(x_2 + x_3)$ is the time elapsed between the first firing instant and the third one.

In a P-TPN, the sojourn time (i.e. the amount of time that a token has been waiting in a place) is counted up as soon as the token has been dropped in the place as seen previously. Thus, quantitative (i.e. performance) considerations take precedence over qualitative (i.e. logical) ones, in opposition to Merlin's time PN model. To compute the firing instants, this approach requires that a token is identified by three parameters: the place that contains it, the information of its creation instant and the information of its consumption one.
Function *TOK* is defined with this purpose. When the weight of the P-TPN arcs is element of $\boldsymbol{N}$, $TOK(j, n)$ is a multi-set.
For the sake of simplicity, only P-TPN with arcs weight element of {0, 1} are considered here. Thus:

$$TOK: \boldsymbol{N} \times \boldsymbol{N}^* \rightarrow \wp(P) \quad \text{(with } \boldsymbol{N}^* \text{ the set of strictly positive natural numbers),}$$

$TOK(j, n) = \{p \in P \,/\, p$ contains a token created by the $j^{th}$ firing instant and consumed by the $n^{th}$ one in $\sigma\}$.
With:

$\wp(P)$ the set of subsets of $\boldsymbol{P}$,
and $\sigma$ a considered firing sequence.

Several tokens contained in the same place will be differentiated by the values $j$ and $n$ associated with them. So, it is possible to impose any token management, but in the sequel a FIFO mode will be considered for the sake of simplicity. Moreover, the determination of these sets is closely linked to the firing sequence considered.

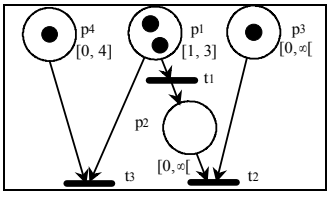



Fig. 8. Illustration of set *TOK*

For instance, the firing sequence considered is $t_1t_2t_3$. The following sets are obtained:

$TOK(0, 1) = \{p_1\}$, $TOK(0, 2) = \{p_3\}$,
$TOK(1, 2) = \{p_2\}$, $TOK(0, 3) = \{p_1, p_4\}$.

Using these sets, the minimal and maximal effective sojourn times of each token in its place are evaluated by:

$$Dsmin(j, n) = \begin{cases} \max(a_i),\ p_i \in TOK(j,n), \\ \text{else } 0 \text{ if } TOK(j,n) = \varnothing \end{cases}$$

$$Dsmax(j, n) = \begin{cases} \min(b_i),\ p_i \in TOK(j,n) \\ \text{else } +\infty \text{ if } TOK(j,n) = \varnothing \end{cases}.$$

Indeed, tokens, with the same creation instant, located in different places and involved in the same transition firing may mutually constrained their (static) sojourn time.
For instance, consider the portion of P-TPN depicted in Fig. 8, $Dsmin(0,3) = \max(0,1) = 1$ and $Dsmax(0,3) = \min(3,4) = 3$ and the initial token in place $p_4$ is affected by the token contained in place $p_1$ as they are involved in the firing of transition $t_3$.
As we are intrested in cyclic behavior, the next paragraph deals with the periodic functioning mode.

### 4.1 Time periodic control

The behavior of this mode is fully determined by:

$$\forall k \geq 1,\ s_i(k) = s_i(1) + (k - 1).\pi \qquad (1)$$

Where $s_i(k)$ is the $k^{th}$ firing date of the transition $t_i$ and $\pi$ the cycle time (or the functioning period). That means that the times of the first firing of the transitions and the functioning period are sufficient to entirely describe the periodic functioning mode. So, a periodic time schedule can be built.
The behavior of a P-TPN can be discribed in terms of a firing schedule.

***Definition* 2**: A P-TPN *Nr* firing schedule, will be a sequence of ordered pairs $(t_i, \sum_{k=0}^{i} x_k)$; transition $t_i$ firable at time $(\sum_{k=0}^{i} x_k)$, obtained from the state reached by starting from *Nr* initial state and firing the transitions $t_j$, $1 \leq j < i$, in the schedule at the given times. It can be noticed that in order to facilitate the understanding, it is considered that the studied firing sequence is the following: $\sigma = t_1 t_2 t_3 \ldots t_q$, thus the $r^{th}$ event is related to the firing of transition $t_r$ at the absolute time $\sum_{k=0}^{r} x_k$.

The next section deals with the control approach.

## 5. Control approach

### 5.1 Interval arithmetic

Let $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$, with $0 \leq a_1 \leq b_1 \leq +\infty$ and $0 \leq a_2 \leq b_2 \leq +\infty$. Then $I_1 + I_2$ can be defined as the interval $[a_1 + a_2, b_1 + b_2]$ and $I_1 - I_2$ as the interval $[a_1 - a_2, b_1 - b_2]$ if $a_2 \leq a_1$ and $b_2 = a_2$. Let IV be the interval $[\alpha, \beta]$, $IV^{min}$ will represent $\alpha$ and $IV^{max}$ will represent $\beta$.

In this subsection, for the sake of simplicity, it will be considered that, starting from the initial state the firing of an enabled transition $t_i$ will lead to the state labelled by $i$ (the $i^{th}$ state).

The principle of the proposed control approach will be first illustrated on the following simple example.

At the initial state, the interval of availability of each token in place $p_i$, $DS_0(p_i)$ can be evaluated as follows:

$$\forall p_i \in TOK(0, .),\ DS_0(p_i) = [a_i, b_i],$$
$$\text{and } MIN_0 = Dsmax(0, 1),$$

On the P-TPN model of Fig. 9:

$$DS_0(p_1) = [4, 6],\ DS_0(p_3) = [3, 5] \text{ and } MIN_0 = 5.$$

$DS_0$ contains the static intervals of all the tokens initially present in the net and $MIN_0$ represents the time after which a time constraint is ineluctably violated leading to the death of a token.

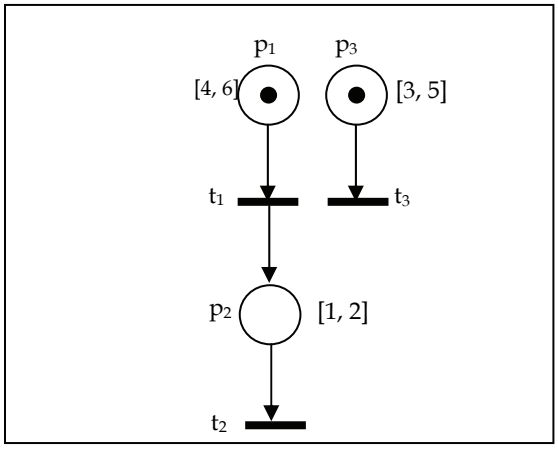



Fig. 9. Illustration.

An enabled transition $t_1$ is said to be fireable if:

$t_1$ is enabled in the autonomous PN sense (i.e. $M_0 \geq \text{Pre}(., t_1)$, with $M_0$ the initial marking)

and

$$Dsmin(0, 1) \leq MIN_0.$$

On the P-TPN model of Fig. 9, transition $t_1$ is fireable because there is one token in place $p_1$ and $DS_0^{\min}(p_1) \leq 5$.

After the firing of transition $t_1$, two situations should be considered:

1. for the newly created token(s) (created by the first firing instant, the firing of transition $t_1$):

$$\forall p_i \in TOK(1, .), DS_1(p_i) = x_1 + [a_i, b_i],$$

2. for the token(s) created initially and not involved in the firing of $t_1$:

$$\forall p_i \in TOK(0, j), j \neq 1, DS_1(p_i) = [\max(x_1, a_i), b_i],$$
$$MIN_1 = \min(DS_1^{\max}(p_i)), \forall p_i \in TOK(0, j) \cup TOK(1, k), j \neq 1 \text{ and } k > 1.$$

On the P-TPN model of Fig. 9:
The token created in place $p_2$ by the firing of $t_1$ (at time $x_1$) will be considered in the enabledness of transition $t_2$ within the interval $DS_1(p_2) = x_1 + [1, 2] = [x_1 + 1, x_1 + 2]$.
For the token in place $p_3$, created initially and not involved in the firing of $t_1$, the minimal time at which it could participate to the firing of transition $t_3$ must be updated.
For instance, if transition $t_1$ is fired at time $x_1 = 4$, as the firing of $t_3$ will occur after the firing of $t_1$, the token in place $p_3$ will be considered at the earliest (in the firing of transition $t_3$) at time $max(x_1, 3) = 4$, so $DS_1(p_3) = [4, 5]$.
A generalization of the previous principle is realized in the following:

For a given firing sequence $\sigma = t_1 t_2 t_3 \ldots t_q$ (its length corresponding to $|\sigma| = q$) ($M_i$ the marking reached by the firing of $t_i$) the lower (resp. upper) bound denoted $\mu_\sigma^{\min}$ (resp. $\mu_\sigma^{\max}$) of its average cycle time can be computed by the intermediary of the following linear programs, stated as follows:

$$\mu_\sigma^{\min} = \text{minimize}(\pi) \text{ and } \mu_\sigma^{\max} = \text{maximize}(\pi),$$
$$\text{with } \pi = \sum_{i=0}^{|\sigma|} x_i ,$$

subject to the set of constraints:

$$\sum_{i=0}^{j} x_i \in INTV_j, \forall j \in \{0, \ldots, |\sigma|\}.$$

With $INTV_j, \forall j$ obtained as follows:

$$INTV_1 = [Dsmin(0, 1), MIN_0],$$

Evaluation of the set $DS_1$:

- contribution of token(s) created by the firing of $t_1$ (the first fired transition):

$$\forall p_i \in TOK(1, .), DS_1(p_i) = x_1 + [a_i, b_i],$$

- contribution of token(s) created initially and not involved in the firing of $t_1$:

$$\forall p_i \in TOK(0, j),\ j > 1,\ DS_1(p_i) = [\max(x_1, a_i),\ b_i],$$
$$\text{and } MIN_1 = \min(DS_1^{\max}(p_i)),\ \forall p_i \in M_1.$$

The case $i > 1$:

From the marking $M_{i-1}$ the firing of transition $t_i$ leads to the following sets:

$$INTV_i = [\max(\boldsymbol{DS}_{\boldsymbol{i-1}}^{\min}(\boldsymbol{p_k})),\ MIN_{i-1}],$$
$$\forall p_k \in TOK(., i),$$

Evaluation of the set $DS_i$:

1. the newly created token(s):

$$\forall p_j \in TOK(i, .),$$
$$DS_i(p_j) = \sum_{u=0}^{i} x_u + [a_j,\ b_j],$$

the token(s) created by the $s^{th}$ firing instant, with $s < i$, which are not involved in the firing of $t_i$:

$$\forall p_k \in TOK(s, j),\ j > i,$$
$$DS_i(p_k) = [\max(\sum_{u=0}^{i} x_u\ ,\ \sum_{u=0}^{s} x_u + a_k),\ \sum_{u=0}^{s} x_u + b_k],$$

and $MIN_i = \min(DS_i^{\max}(p_k)),\ \forall p_k \in TOK(., j),\ j > i.$

### 5.2 Concurrency considerations

It can be noticed that a strict ordering of the events is imposed: if the obtained inequalities system is written for a firing sequence $\sigma = t_1 t_2 t_3 \ldots t_q$, its emptiness means that the considered sequence is not feasible.

Consider the P-TPN of fig. 10 and the firing sequence $\sigma = t_1 t_2$, the system $S_\sigma$ is:

$$\boldsymbol{S_\sigma} \begin{cases} 4 \le \boldsymbol{x}_1 \le \min(6, 3) \\ \max(\boldsymbol{x}_1, 1) \le \boldsymbol{x}_1 + \boldsymbol{x}_2 \le 3 \end{cases}$$

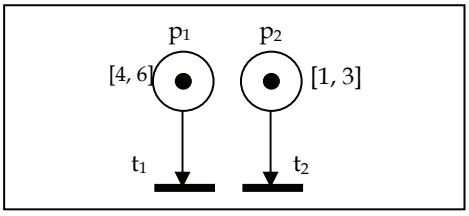



Fig.10. P-TPN with concurrency.

On this simple illustration, due to the timing constraints it is obvious that this system admits no solution traducing the fact that transition $t_1$ cannot be fired before transition $t_2$.

In the system obtained previously a relation of precedence was imposed on the firing instants of the two concurrent transitions $t_1$and $t_2$. This relation was explicitly exhibited by the inequality $\max(x_1, 1) \leq x_1 + x_2$ i.e. the firing of $t_1$ must occur before the firing of $t_2$.
It is proposed to not imposed precedence constraints on the firing of parallels events and no longer consider the constraints induced by token(s) not involved in the firing instant considered which initiate another firing.
For the example of Fig. 10, it yields to the new following system:

$$\boldsymbol{S_\sigma} \begin{cases} 4 \leq \boldsymbol{x}_1 \leq 6 \\ 1 \leq \boldsymbol{x}_1 + \boldsymbol{x}_2 \leq 3 \end{cases}$$

for instance if the firing instant of transition $t_1$ is $x_1 = 6$, it follows: $-5 \leq x_2 \leq -3$, this inequality means that the firing of transition $t_2$ must occur before the firing of $t_1$ and its firing instant corresponds to $x_1+x_2$. Its earliest one is 6 + (-5) = 1 and its latest one is 6 + (-3) = 3.
So, an example of schedule is:

$$\varpi = ((t_2, x_1 + x_2 = 6\text{-}5 = 1), (t_1, x_1 = 6)).$$

It can be noticed that the ordering of the events in the schedule considered will be made on the basis of the value of the sum $\sum_{k=0}^{i} x_k$ .
More formally, the set *TS*, defined as follows need to be introduced:

*TS*: $\boldsymbol{N} \rightarrow$ subset of (*T*),
$i \rightarrow \{t \in T \mid t$ is enabled by the $i^{th}$ firing instant and $t$ is a persistent transition$\}$.
With *T* the set of transitions of the net and ***N*** the set of naturals numbers. An enabled transition is called a persistent transition if it can be disabled only by its own firing.
For the evaluation of each firing instant the following procedure must be applied.
Consider the current fired transition of the considered sequence to be $t_i$. So, the net is in the state reached after the firing of the $i^{th}$ fired transition.

Let $TS(i) = \{t_{k1}, t_{k2}, t_{k3}, ...., t_{km}\}$ ($|\, TS(i)\,| = n$, i.e. the cardinal of the set).

The firing of transition $t_{kj}$ with $j = 1, ..., n$ will be associated with the $(i + j)^{th}$ firing instant.

$$\forall u = i + 1, ..., i + n, \forall\, p_v \in TOK(., u),$$

$$INTV_u = [\max(\boldsymbol{DS}^{\min}_{\boldsymbol{u}-1}(\boldsymbol{p_k})), \boldsymbol{MIN}_{\boldsymbol{u}-1}\big|\boldsymbol{p_r} \in \boldsymbol{TOK}(., \boldsymbol{u}) \cup \boldsymbol{TOK}(., \boldsymbol{s})],$$

$$\text{with } s > i + n.$$

$\boldsymbol{MIN_s}\big|\boldsymbol{p_r} \in \boldsymbol{P'}$ with $\boldsymbol{P'} \subset \boldsymbol{P}$ is the restriction of *MINs* to the set of places *P'* (i.e. *MINs* is evaluated by considering only the contribution of the places of the set *P'*).
The preceding expression allows to consider, for a particular firing instant, the tokens which participate to this firing and those present at the evaluation instant but which are not involved in the enabledness of a transition (they must be prevented from dying). Thus, the tokens involved in the firing of a parallel transition are not considered.
The benefits of this approach are patent because in presence of concurrency, it is not necessary to take into consideration the set of all possible interleavings. An "untimed firing sequence" (feasible on the untimed underlying Petri net model) will be considered and via

linear programming techniques the solution obtained by solving the obtained system (associated with the firing sequence considered) will determine the "real" order of the events and the exact duration of the resulting firing sequence. For instance, consider the case of $n$ transitions which can concurrently fire, from a state. The application of the presented method will result in an unique inequations system instead of $n!$ systems, each one being associated with a possible combination of the order of the events.
Furthermore, this approach can be used to bring answers to solve reachability problems. Indeed, by considering the possible evolutions from a source marking and leading to a target marking, in terms of firing sequence, a set of inequalities systems can be obtained. So, if this set of systems admits no solution it means that there is no feasible firing sequence on the P-TPN model allowing to, starting from the source marking, reach the target one. It is well known that enumerative based methods usually face the so-called state space explosion problem. Although the investigation of the feasible sequences is realized on the untimed model, the approach may be hampered by the complexity inherent to this kind of procedure.

## 6. Illustrative example

In this section, an example is used to illustrate the developed approach. The background is the supervision of a control distributed in several Programmable Logic Controllers (PLC on short).
The supervision consists of accessing via a network to control system variables (temperature, pressure, level in a tank,...) by means of different requests to the PLCs. The duration of requests depends on many factors as medium and protocol used, PLC activity, number of variables to be read or write, ... Generally, these durations are known not exactly but by a time interval.
However, ensuring the consistency of the controlled variables to be spied and/or modified requires that the duration between different requests does not exceed a defined duration imposed by the dynamic of the observed process.
Then, the consistency constraint is not a time-out one, but an operation time one (its latency can be stated by means of a time interval in a general case).
Consequently, to model naturally the repetitive functioning of the supervisor, a P-time PN model can be built. Fig. 11 represents a scheduling problem of a supervisor.
This supervision consists of making 6 different requests ($p_{12}$, $p_{14}$, $p_{22}$, $p_{25}$, $p_{32}$, $p_{34}$) to the three PLCs ($p_{10}$, $p_{20}$, $p_{30}$).
This model takes into account:

- Precedence constraints (e.g. by means of ($p_{12}$, $t_{12}$, $p_{13}$, $t_{13}$, $p_{14}$) the request associated with $p_{14}$ must be processed after the one associated with $p_{12}$ ).
- Synchronization constraints, e.g. by means of $t_{40}$, all the requests must be processed to ensure the consistency constraint (no request can be processed twice without processed all the others).
- Shared resources constraints: the PLC associated with $p_{20}$ cannot process the requests $p_{22}$ and $p_{25}$ at the same time.

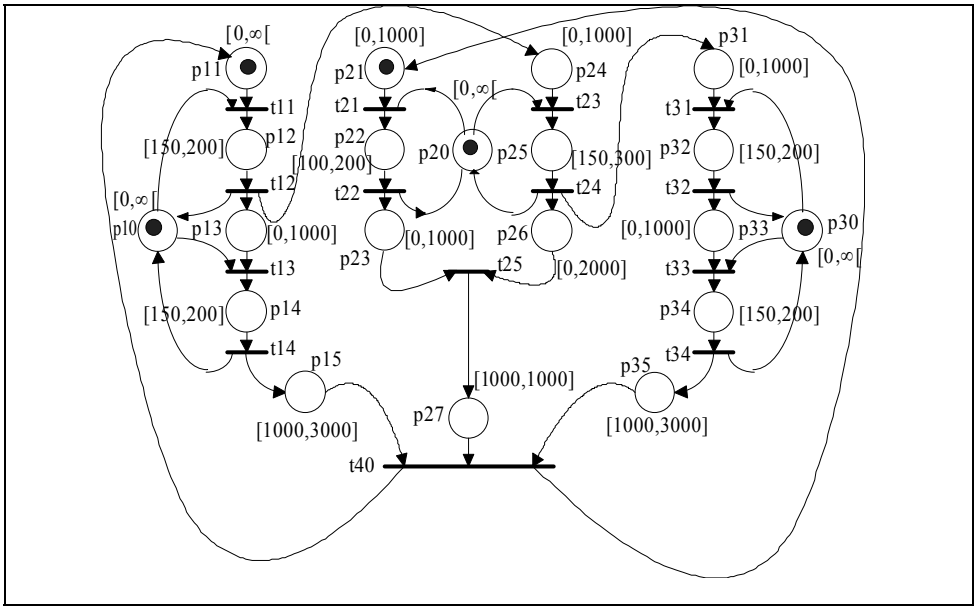



Fig. 11. P-TPN model

| | |
|---|---|
| $t_{11}$ | start processing request 1 |
| $t_{12}$ | completion of request 1 |
| $t_{13}$ | start processing request 2 |
| $t_{14}$ | completion of request 2 |
| $t_{21}$ | start processing request 3 |
| $t_{22}$ | completion of request 3 |
| $t_{23}$ | start processing request 4 |
| $t_{24}$ | completion of request 4 |
| $t_{25}$ | consistency of request 3 with request 4 |
| $t_{31}$ | start processing request 5 |
| $t_{32}$ | completion of request 5 |
| $t_{33}$ | start processing request 6 |
| $t_{34}$ | completion of request 6 |
| $t_{40}$ | global consistency |

Table 1. Meaning of transitions of Fig. 11

Consider the firing sequence $\sigma = t_{11}t_{12}\, t_{13}t_{14}\, t_{21}t_{22}\, t_{23}t_{24}\, t_{25}t_{31}\, t_{32}t_{33}\, t_{34}t_{40}$, the simplified following inequalities system is then obtained:

$$0 \le x_1 \le \infty$$
$$150 \le x_2 \le 200$$
$$0 \le x_3 \le 1000$$
$$150 \le x_4 \le 200$$
$$0 \le \sum_{i=1}^{5} x_i \le 1000$$
$$\sum_{i=1}^{5} x_i + 100 \le \sum_{i=1}^{6} x_i \le \sum_{i=1}^{5} x_i + 200$$
$$\max(\sum_{i=1}^{6} x_i, \sum_{i=1}^{2} x_i) \le \sum_{i=1}^{7} x_i \le \min(\sum_{i=1}^{5} x_i + 200, \sum_{i=1}^{4} x_i + 3000)$$
$$\sum_{i=1}^{7} x_i + 150 \le \sum_{i=1}^{8} x_i \le \min(\sum_{i=1}^{4} x_i + 3000, \sum_{i=1}^{7} x_i + 300)$$
$$\max(\sum_{i=1}^{6} x_i, \sum_{i=1}^{8} x_i) \le \sum_{i=1}^{9} x_i \le \min(\sum_{i=1}^{4} x_i + 3000, \sum_{i=1}^{6} x_i + 1000, \sum_{i=1}^{8} x_i + 2000)$$
$$\sum_{i=1}^{8} x_i \le \sum_{i=1}^{10} x_i \le \min(\sum_{i=1}^{4} x_i + 3000, \sum_{i=1}^{8} x_i + 1000, \sum_{i=1}^{9} x_i + 1000)$$
$$\sum_{i=1}^{10} x_i + 150 \le \sum_{i=1}^{11} x_i \le \min(\sum_{i=1}^{4} x_i + 3000, \sum_{i=1}^{9} x_i + 1000, \sum_{i=1}^{10} x_i + 200)$$
$$\sum_{i=1}^{11} x_i \le \sum_{i=1}^{12} x_i \le \min(\sum_{i=1}^{4} x_i + 3000, \sum_{i=1}^{9} x_i + 1000, \sum_{i=1}^{11} x_i + 1000)$$
$$\sum_{i=1}^{12} x_i + 150 \le \sum_{i=1}^{13} x_i \le \min(\sum_{i=1}^{4} x_i + 3000, \sum_{i=1}^{9} x_i + 1000, \sum_{i=1}^{12} x_i + 200)$$
$$\max(\sum_{i=1}^{4} x_i + 1000, \sum_{i=1}^{9} x_i + 1000, \sum_{i=1}^{13} x_i + 1000) \le \sum_{i=1}^{14} x_i$$
$$\sum_{i=1}^{14} x_i \le \min(\sum_{i=1}^{4} x_i + 3000, \sum_{i=1}^{9} x_i + 1000, \sum_{i=1}^{13} x_i + 3000)$$

Let $\pi = \sum_{i=0}^{14} x_i$ and $\mu_\sigma^{\min}$ = minimize($\pi$) and $\mu_\sigma^{\max}$ = maximize($\pi$),

subject to the constraints of the previous system lead to $\mu_\sigma^{\min}$ = 1600 and $\mu_\sigma^{\max}$ = 3200 units of time.

For instance, the firing instants (the $x_i$'s) corresponding to $\mu_\sigma^{\min}$ are:

[0, 150, 300, 150, -550, 100, 0, 150, 300, -300, 150, 0, 150, 1000] and the associated schedule is the following: $\varpi$ = (( $t_{11}$, 0), ($t_{21}$, 50), ( $t_{12}$, 150), ($t_{22}$, 150), ( $t_{23}$, 150), ($t_{24}$, 300), ( $t_{31}$, 300), ($t_{13}$, 450), ( $t_{32}$, 450), ($t_{33}$, 450), ( $t_{14}$, 600), ($t_{25}$, 600), ( $t_{34}$, 600), ($t_{40}$, 1600)).
It can be noticed that, although the inequalities system was obtained on the basis of firing sequence $\sigma = t_{11}t_{12}\ t_{13}t_{14}\ t_{21}t_{22}\ t_{23}t_{24}\ t_{25}t_{31}\ t_{32}t_{33}\ t_{34}t_{40}$, the real feasible sequence is $t_{11}t_{21}\ t_{12}t_{22}\ t_{23}t_{24}\ t_{31}t_{13}\ t_{32}t_{33}\ t_{14}t_{25}\ t_{34}t_{40}$.
In the case of $\mu_\sigma^{\max}$, the corresponding firing instants (the $x_i$'s) are:

[1600, 150, 300, 150, -1200, 200, 550, 150, 300, -300, 150, 0, 150, 1000] and the associated schedule is the following:
$\varpi'$ = (( $t_{21}$, 1000), ($t_{22}$, 1200), ( $t_{11}$, 1600), ($t_{12}$, 1750), ( $t_{23}$, 1750), ($t_{24}$, 1900), ( $t_{31}$, 1900), ($t_{13}$, 2050), ( $t_{32}$, 2050), ($t_{33}$, 2050), ( $t_{14}$, 2200), ($t_{25}$, 2200), ( $t_{34}$, 2200), ($t_{40}$, 3200)).

The inequalities system was obtained for a firing sequence which can be viewed as a "pseudo firing sequence".
Indeed, it is used to determine the logical constraints (precedence, mutual exclusion and/or concurrency) but the real order of the events (i.e. the real firing sequence) is obtained thanks to the value and the sign of the firing instants resulting from the resolution of the considered system.

## 6. Conclusion and future work

Time has become a major issue in the analysis of production systems. Indeed, for time-critical systems where all taks are time-constrained, this parameter does not affect only the system performances but also its correctness.
Thus, an acceptable behavior of such systems depends not only on the order of the events but particularly on the time at which the results are produced. Consequently, correctness and performance issues are closely linked. Due to time intervals specifications, time critical systems require a time control.
In this paper, a new approach to design a time control has been proposed. It uses P-time Petri net as a modeling tool and it uses the firing instants notion that does not require strong structural properties (it is not restricted to subclasses of PN).
The proposed method is based on a token player algorithm and it investigates the set of feasible firing sequences of the underlying untimed Petri net of the considered P-TPN.
The presented technique yields to the obtaining of an inequalities system written once (at the beginning of the analysis) for symbolic values of the timing constraints associated with each place of the considered time model. The ability of not considering only numerical quantities can also be used in order to test real-time systems specifications.
Indeed, as seen in the illustrative example, the (logical) ordering of particular operations can be imposed by the timing constraints, so if an obtained ordering of the events is inappropriate the timing constraints can be modified without altering the system considered and the new scheduling will result in the resolution of the original system.
An issue currently being investigated is the integration of the presented approach in the evaluation of the robustness of the modelled system.

## 7. References

Van der Aalst, W. M. P. (1993). Interval Timed Coloured Petri Nets and Their Analysis, In: *Application and Theory of Petri Nets,* M. Ajmone Marsan, (Ed.), 453-472, volume 691 of Lecture Notes in Computer Science, Springer-Verlag

Berthomieu, B. & Diaz, M. (1991). Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. Soft. Eng,* Vol. 16, No. 3, 259-273

Bonhomme, P. (2006). Control and Performances Evaluation of Time Dependent Systems Using an Enumerative Approach, *Proceedings of Conf. on Control Applications, CCA/ISIC*, Germany, Munich

Bonhomme, P., Aygalinc P. & Calvez, S. (2001). Towards the Control of Time-Critical Systems, *Proceedings of Conf. on Control Applications, CCA/ISIC*, Mexico

Boucheneb, H. & Berthelot, G. (1993). Towards a simplified building of Time Petri Nets reachability graph, *Proceedings of PNPM'93,* pp. 46-55

Bucci, G. & Vivario, E. (1995). Compositional validation of time-critical systems using communicating time Petri nets. *IEEE Trans. Soft. Eng,* Vol. 21, No. 12, 969-992

Calvez, S., Aygalinc P. & Bonhomme, P. (2004). Proactive/Reactive Approach for Transient Functioning Mode of Time Critical Systems, *Proceedings of the Third Conference. on Management and Control of Production and Logistics(MCPL'2004),* pp. 65-70, Santiago, Chili

David, R. & Alla, H. (1994). Petri Nets for Modeling of Dynamic Systems - A survey. *Automatica,* Vol. 30, No.2, 175-202

Declerck, P. & Alaoui, M. (2004). Modelling and analysis of p-time event graphs in the (min, max, +) algebra, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC'04),* pp. 1807-1812, The Hague, The Netherlands

Diaz, M. & Senac, P. (1994). Time stream Petri nets: a model for timed multimedia information, *Proceedings of the 15th International Conference on Application and Theory of Petri Nets,* 219-238, volume 815 of Lecture Notes in Computer Science, Springer-Verlag

Freedman, P. (1991). Time, Petri nets, and robotics Robotics and Automation. *IEEE Transactions on Robotics and Automation,* Vol. 7, No. 4, 417 – 433

Ghezzi, C. Mandrioli, D., Morasca, S. & Pezzè, M. (1991). A Unified High-level Petri Net Model for Time Critical Systems, *IEEE Trans. Soft. Eng,* Vol. 17, No. 2, 160 – 172

Giua, A., Seatzu, C. & Basile, F. (2004). Observer-based state-feedback control of timed Petri nets with deadlock recovery. *IEEE Trans. On Automatic Control,* Vol. 49, No. 1, 17-29

Jiroveanu, G., Boel, R. K. & De Schutter, B. (2006). Fault diagnosis for time Petri nets, *Proceedings of the 8th International Workshop on Discrete Event Systems (WODES'06),* pp. 313–318, Ann Arbor, Michigan

Lilius, J. (1999). Efficient state space search for time Petri nets, *Electronic Notes in Theoretical Computer Science,* Vol. 18

Khansa, W., Denat, J. P. & Collart, S. (1996). Fault P-Time Petri Nets for Manufacturing Systems, *Proceedings of Wodes'96,* pp. 19-21, Edinburgh, UK

Merlin, P. & Faber, D. (1976). Recoverability of communication protocols - implications of a theoretical study. *IEEE Trans. Communications,* Vol. 24, No. 9, 381-404

Murata, T. (1989). Petri Nets : Properties, Analysis and Applications. *Proc. of the IEEE,* Vol. 77, No. 4, 541-580

Ramadge, P. J. & Wonham, W. M. (1989). The control of discrete event systems. *Proc. of the IEEE,* Vol. 77, No. 1, 81-98

Ramchandani, C. (1973). Analysis of asynchronous concurrent systems by Petri nets. Ph.D. Thesis, MIT

Roux, O. H.. & Déplanche, A. M. (2002). A t-time Petri net extension for real time-task scheduling modeling. *European Journal of Automation (JESA),* Vol. 36, No. 7, 973-986

Sathaye, A. S. & Krogh. G. H. (1998). Supervisor Synthesis for Real-Time Discrete Event Systems, *Discrete Event Dynamic Systems,* Vol. 8, No. 1, 5-35

Sifakis, J. (1977). Use of Petri nets for performance evaluation, In: *Measuring, Modelling and Evaluating Computer Systems,* H. Beilner and E. Gelenbe, (Ed.), 75-93

Wang, J., Deng, Y. & Xu, G. (2000). Reachability Analysis of real-Time Systems Using Time Petri Nets. *IEEE Trans.* on Systems, Man and Cybernetics-Part B: Cybernetics, Vol. 30, No 5, 725-736

Wang, H., Grigore, L., Buy, U. & Darabi, H. (2007). Enforcing Transition Deadlines in Time Petri Nets, *Proceedings of the 12th IEEE Conf. on Emerging Technologies and Factory Automation, ETFA 2007,* Greece

Zuberek, W. M. (2000) Timed Petri net models of cluster tools, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC'2000)*, pp. 3063-3068, Vol. 4, Nashville, TN

10

# CL-MAC: Cross-layer MAC Protocol for Delay Sensitive Wireless Sensor Network Applications

Kechar Bouabdellah and Sekhri Larbi
*University of Oran*
*Algeria*

## 1. Introduction

Recent advances in micro-electromechanical systems (MEMSs) technology, wireless communications field and nanotechnology have enabled the design of low-power, low cost smart sensor nodes equipped with multiple onboard functions such as sensing, computing, and communications. Such intelligent devices networked through wireless links have been referred to as Wireless Sensor Networks (WSN). The basic function of the network is to observe some phenomenon by using the sensors and communicate the sensed data to a common destination called the base station or the Sink. In most application scenarios, sensor nodes are powered by small batteries, which are practically non-rechargeable, either due to cost limitations or because they are deployed in hostile environments.

Many WSN applications, that are delay sensitive in case when an abnormal event occurs, exist in practice: environmental monitoring (for example forest fire detection, intruder detection), assistance for old or disabled people and structural health monitoring. In these applications, the detected event is considered as an urgent data which must be transmitted quickly towards the Sink for fast intervention. To achieve this requirement, it is necessary to decrease latency at MAC layer when transmitting urgent data from the source node to the Sink.

These considerations motivate well energy saving and low latency WSN designs.

Many research works have been developed for energy efficiency at each layer of the protocol stack by proposing new algorithms and protocols. In particular, the MAC layer was of great interest for many researchers because it was considered as an important source of energy waste. It is summarized in (Zhi-Wen et al., 2005; Injong et al., 2005; Muneeb et al., 2006; Sohraby et al., 2007):

- *Overhearing*: a sensor node receives packets that are transmitted for other nodes. This is mainly due to the radio transmission nature (omni-directional) forcing every node of the neighborhood to waste energy when receiving and decoding these packets. These packets are eventually dropped after the node realizes that the destination address is different from its own address.
- *Collision*: since the radio channel is shared by many nodes, a collision takes place every time when two nodes try to send their packets at the same time. Collisions increase energy consumption and latency in case of packets deliverance mechanism due to retransmissions.

- *Control packets (overhead)*: packet headers and control packets (*RTS/CTS/ACK*) used by a MAC protocol do not contain application data, thus they are considered as supplementary data (*overhead*). Control packets can be of importance since most applications use data packets with reduced size.
- *Idle listening*: it is a dominant factor for energy waste in WSN. Indeed, when a node is not in the transmission mode, it must continuously listen to the channel in order to receive possible traffic that is not sent. In this case, the amount of energy waste is almost equal to the energy dissipated by a normal reception according to (Wei et al., 2004) (the ratios of $E_{idle}$:$E_{receiving}$:$E_{transmitting}$ are 1:1.05:1.4).
- *Over emitting*: this case occurs when a sensor node receives a packet while it is not ready. This situation forces the sender to perform new retransmissions that are strongly linked to synchronisation problem and therefore wastes energy.

In order to decrease or at least eliminate these various sources of energy waste, several protocols have been proposed these last years. They can be divided into two main classes: TDMA-based MAC protocols and Contention-based MAC protocols.

### 1.1 TDMA-based MAC protocols

These protocols (known as deterministic) are employed to avoid collisions by exclusively allocating time slots to sensor nodes. However, these protocols require the presence of a management authority (for example a dedicated access point) to regulate the access to the medium by broadcasting a schedule that specifies when, and for how long, each controlled sensor node may transmit over the shared channel. In these protocols, the channel is divided into time slots, which are grouped into logical frames (see Fig. 1 in which a set of N contiguous slots form a logical frame). In each logical frame each sensor node is assigned a set of specific time slots. This set constitutes the schedule according to which the sensor node operates in each logical frame.

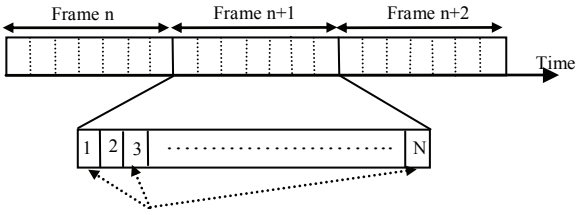



Fig. 1. Logical frame in TDMA-based protocols for WSN

The schedule can be either fixed, constructed on demand on a per-frame basis by the base station or hybrid (Sohraby et al., 2007). Outside these assigned slots, a sensor node goes to sleep mode in which the radio transceiver is completely turned off to conserve energy. However, in WSN we need distributed protocols to allocate time slots to sensor nodes (Willig, 2006), but such distributed schemes tend to be somewhat complex (see for example SMACS (Sohrabi & Pottie, 1999; Sohrabi et al., 2000), TRAMA (Rajendran et al., 2003) or LEACH (Heinzelman et al., 2002)). Network topology changes (due for example to sensor nodes running out of energy, the deployment of new nodes or node mobility) require the slot allocation protocol been executed periodically. In addition, TDMA-based MAC schemes

require tight time synchronization between nodes to avoid overlap of time slots. This in turn requires continuous execution of a time synchronization protocol.
This makes the use of these protocols more complex in WSN where each node has, in general, no priority assigned and very limited resources.

### 1.2 Contention-based MAC protocols

These protocols known as CSMA-based are usually used in multi-hop wireless networking due to their simplicity and their adequacy to be implemented in a decentralized environment like WSN. When these protocols are used, collisions can be occur in case of a receiver is located in the radio range of at least two sensor nodes transmitting simultaneously data packets to it. Collisions waste the energy of both the transmitter and the receiver and as a result packet retransmissions can occur which create additional load for a congested channel. In CSMA-based protocols, collisions are often the result of hidden terminal problem. Consider the situation in Fig. 2 where A and B can hear each other, B and C can hear each other but A and C cannot. Nodes A and C both want to transmit a packet to their common neighbour B. Both nodes sense an idle channel and start to transmit their packets. The signals of nodes A and C overlap at B and are destroyed (collision problem).

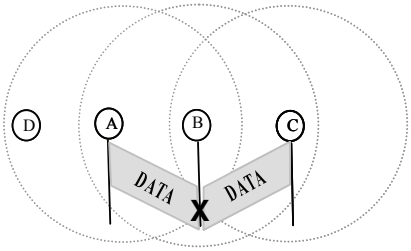



Fig. 2. Hidden terminal scenario

To reduce these collisions in ad-hoc (sensor) networks, the 802.11 standard defines a virtual carrier mechanism based on the Request-To-Send (RTS)/(Clear-To-Send) CTS scheme defined in MACA protocol (MultiAccess Collision Advoidance) (Karn, 1990). By using this scheme, collisions between hidden nodes at common neighbors can be avoided. A sensor node (node A in Fig. 3) wanting to transmit a unicast packet initiates a handshake by transmitting an RTS control packet after a specified time called the Distributed Inter Frame Space (DIFS). The receiver (node B in Fig. 3)) waits a Short Inter Frame Space (SIFS) before responding by sending a CTS control packet, which informs all its neighbors of the upcoming transfer. Since the SIFS interval is set shorter than the DIFS interval, the receiver takes precedence over any other sensor node attempting to send a packet (Koen & Gertjan, 2004). The effective DATA transfer (from A to B) is now guaranteed to be collision free. So, after a SIFS period, DATA packet is transmitted by sender (A) and receiver (B) waits a SIFS period before acknowledging the reception of the data by sending an ACKnowledgement control packet (ACK). If sender (A) does not receive the ACK packet, it assumes that the data was lost due to a collision at receiver (B) and enters a binary exponential backoff procedure. This same procedure can be used when two RTS packets collide, which is technically still possible. The RTS/CTS control packets specify in their header the duration

of the upcoming DATA/ACK sequence which allow neighboring nodes to overhear the control packets, set their Network Allocation Vector (NAV) and defer transmissions until it expires (NAV(RTS) at neighbor node D and NAV(CTS) at neighbor node C in Fig. 3). The radio can be switched off for the duration NAV to save energy waste. Note that, all broadcast and multicast packets are transmitted without RTS/CTS scheme.
In the contention-based MAC protocols, the RTS/CTS scheme is sufficient to greatly reduce collisions and increase bandwidth utilization, but unfortunately does not completely solve the hidden terminal problem, according to (Sohraby et al., 2007).

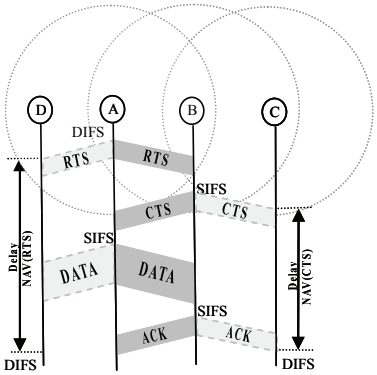



Fig. 3. Collision avoidance using RTS/CTS handshake (Sohraby et al., 2007)

A number of MAC protocols have been developed to considerably reduce other sources of energy waste. They focus on reducing idle listening, but collisions, control packets (overhead) and overhearing are also addressed.
The first approach to reduce idle listening is based on preamble sampling and operates at the physical layer (see for example (Halkes et al., 2005; Bachir et al., 2006; EL-Hoiydi, 2002; Polastre et al., 2004; Buettner et al., 2006). In this asynchronous approach, as illustrated in Fig. 4, a receiver turns on periodically the radio to sample for incoming data and detects if a preamble is present or not (Halkes at al., 2005 ). If it detects a preamble, it will continue listening until the start-symbol arrives and the message can be properly received. If no preamble is detected the radio is turned-off again until the next sample.

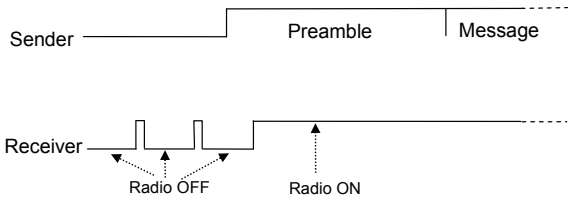



Fig. 4. A sender uses a longer preamble to allow the receiver to only turn its radio on periodically (Halkes et al., 2005)

The second approach trying to mitigate the idle listening uses wake-up/sleep mechanisms and/or *RTS/CTS/DATA/ACK* signalling scheme from 802.11x standard to reduce collision, overhearing and control packet (overhead). A well know MAC protocols in the literature using this approach are S-MAC, T-MAC (with the automatic adaptation of the duty cycle to the network traffic), D-MAC and Z-MAC (see section 2.1 for more detail).
Recently, a new generation of MAC protocols (*Cross-layer MAC protocols)* using several layers in order to optimize energy consumption has emerged. These layers can be exploited into two modes: *interaction* or *unification* as depicted in Fig. 5. In the interaction mode, the MAC protocol is built by exploiting the data generated by other adjacent layers. MAC-CROSS Protocol (Suh et al., 2006) is an example of Cross-layer approach which allow the routing information of the network layer to be exploited by the MAC layer (interaction between MAC and network layers) by leaving only the communicating nodes in activity and by putting into sleep mode the other neighbor nodes (not concerned by this communication). In order to avoid collisions, MAC-CROSS uses the control messages *RTS/CTS/ACK*. On the other hand, a Cross-layer design mode by unification requires the development of only one layer including at the same time the functionalities of considered layers.

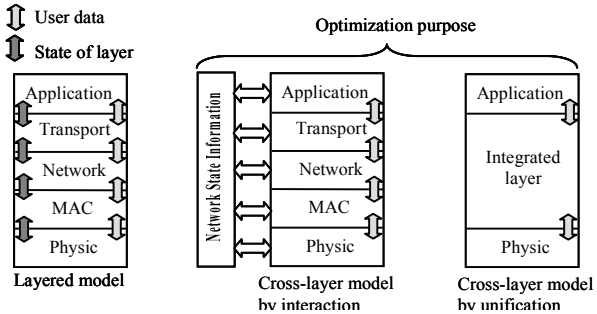



Fig. 5. Cross-layer model illustration

In this chapter, we propose a Cross-layer protocol named CL-MAC, based on the same ideas used by MAC-CROSS. The fundamental difference between our proposal and MAC-CROSS lies on the level of the number of consecutive nodes that are implied in MAC functioning at each frame. Indeed, MAC-CROSS acts on three consecutive communicating nodes while CL-MAC uses all the nodes included in a given routing path from the source node to the Sink in one frame. Two main operations take place simultaneously in this routing path after an RTS/CTS exchange at the beginning of each data transmission: on one hand successive transmissions of CTS packets which advance quickly towards the Sink in order to reserve a path. All nodes included in this path remain in activity and all other nodes in their vicinity enter sleep mode for a given time interval. On the other hand DATA/ACK packet exchanges between communicating nodes in the routing path (a relatively slow process) which advance progressively.
Temporal Petri nets are introduced in order to model underlying operation of the proposed protocol and the TiNA tool is carried out for analytical validation of some related properties. A comparative study between CL-MAC, MAC-CROSS and S-MAC in term of energy saving and low latency has been performed for evaluation purpose by using a home simulator.

The rest of the chapter is organized as follow: in the next section, we introduce main works in literature related to energy saving at the MAC layer level. Some OSI-based protocols and others based on Cross-layering approach are given in this section. In section 3, we give more details about the proposed protocol CL-MAC. In Section 4, we present a formal representation of the CL-MAC protocol using time Petri nets modeling approach. The analytical validation of some properties of CL-MAC using the TiNA software tool is given in section 5. The performance evaluation of CL-MAC protocol by comparison with a similar MAC protocols like S-MAC and MAC-CROSS is presented in section 6. Finally, we conclude our work and discuss some future perspectives.

## 2. Related work

In this section, we present some MAC layer protocols developed some years ago that enable energy conservation in WSN. First, compatible OSI protocols are presented and followed by two important cross-layer protocols: MAC-CROSS and XLM. Especially, MAC-CROSS protocol is considered as a basis of the development of our proposal.

### 2.1 Compatible OSI protocols

Many studies in WSN have showed that energy consumption during a communication is four times greater than the energy consumed in both the processing and sensing operations. This fact lead communication protocols designers to take a particular interest into the WSN-MAC layer and to propose some original ideas to efficiently manage that layer. The medium access must take into account all sources of energy waste considerations.

Sensor-MAC (S-MAC) protocol is a very popular protocol developed at California University (Wei et al., 2002; Koen & Gertjan, 2004; Zhi-Wen et al., 2005). Its main objective is to conserve energy in WSN and it takes into consideration that fairness and latency are less critical issues compared to energy conservation. The basic idea behind S-MAC is the management of local synchronizations and the schedule of sleep/listen periods based on these synchronizations. Neighboring nodes form virtual clusters in which they periodically broadcast special SYNC messages to keep synchronized. The period for each node to send a SYNC packet is called the synchronization period. If two neighboring nodes reside in two virtual clusters, they wake up at the listen periods of both clusters (Demirkol, 2006). Every frame in S-MAC as shown in Fig. 6 is divided into an active period and a sleep period. The active period is divided into three parts for SYNC, RTS and CTS packets. In this figure, nodes 1 and 3 are synchronized to the schedule of node 2 by receiving its SYNC packet. This means that nodes 1, 2 ad 3 share a same virtual cluster. Node 3 initiates an RTS/CTS exchange with node 1 to transmit data. When CTS packet is received, data transmission will immediately follow. Nodes 1 and 3 stay active until the completion of data transfer, whereas node 2 follows its normal sleep schedule. In the sleep period, when data transmission ends, communicating nodes enter sleep mode by switching off their radio transceivers. Collision avoidance is achieved by a carrier sense (CS in Fig. 6).

S-MAC also includes message passing support in which long messages are divided into frames and sent in a burst-mode. In this case, only one RTS and one CTS are used to reserve the medium for the time needed to transmit all fragments.

Several other energy efficient protocols in the literature are based on wake-up/sleep mechanism: T-MAC (Koen & Gertjan, 2004), D-MAC (Lu et al., 2004), Z-MAC (Injong et al.,

2005). T-MAC and D-MAC are considered similar to the S-MAC protocol with an adaptive duty cycle.
The T-MAC (Timeout-MAC) protocol adapts the duty cycle to the network traffic in order to improve S-MAC, but instead of using a fixed length active period, T-MAC uses a time-out

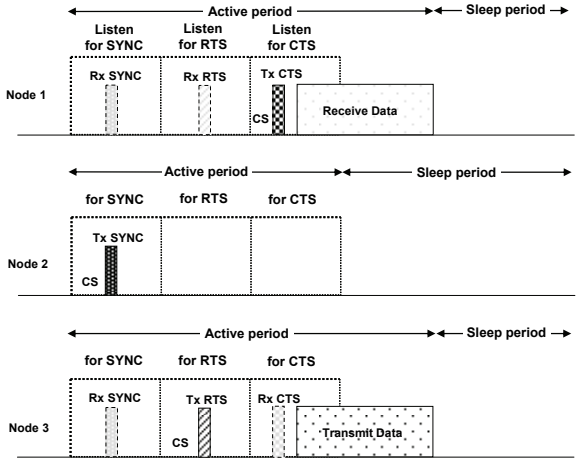



Fig. 6. Timing relationship between different sensor nodes in SMAC (This figure was redrawn from (Dewasurenda & Mishra, 2005))

mechanism to dynamically determine the end of the active period (Halkes, 2005). In Fig. 7, when a node does not detect any activity within the activity time-out period (TA), it can safely assume that no neighbor wants to communicate with it and then enters sleep period. If the node engages or overhears a communication, it simply starts a new TA period after that communication finishes.
The D-MAC (Data-gathering MAC) protocol includes an adaptive duty cycle like T-MAC for energy efficiency and ease of use. In addition, it provides low node-to-Sink latency, which is achieved by supporting convergecast communication paradigm that is the mostly observed communication pattern within sensor networks. DMAC achieves very good latency compared to other sleep/listen period assignment methods, but unfortunately collision avoidance methods are not utilized.

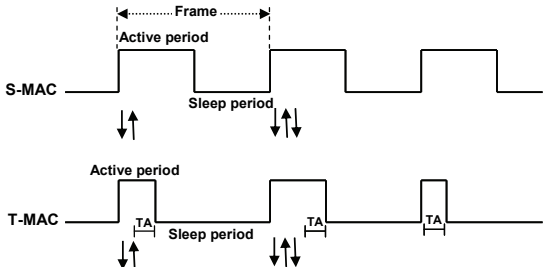



Fig. 7. The S-MAC and T-MAC duty cycles; the arrows indicate transmitted and received messages; note that messages come closer together (TA: *Activity Time-out period*) (Halkes et al., 2005)

Z-MAC (Zebra MAC) is a hybrid MAC protocol for wireless sensor networks that combines the strengths of TDMA and CSMA while offsetting their weaknesses. The main feature of Z-MAC is its adaptability to the level of contention in the network so that under low contention, it behaves like CSMA, and under high contention, like TDMA. A distinctive feature of Z-MAC is that its performance is robust to synchronization errors, slot assignment failures and time-varying channel conditions. Z-MAC is used as the default MAC for Mica2 mote.

### 2.2 Cross-layer protocols

Other protocols based on OSI layer models try to reduce problems encountered in WSN. Network layer protocols tend to optimize paths between network's nodes and the Sink while application layer protocols try to obtain correct, accurate and compressed and/or aggregated information (Holger et al., 2003; Cheng et al., 2006) so as to reduce the amount of packets in the network. OSI-based protocols are not flexible, not optimal and consequently reduce network performances. To mitigate these drawbacks, a new MAC approach based on interaction or unification of two or more adjacent layers, called Cross-layer MAC optimization, has emerged (Akyildiz & Ismail, 2004; Suh et al., 2006; Akyildiz et al., 2006). Some protocols using a Cross-layer technique in medium access control layer can be found in literature such as MAC-CROSS (Suh et al., 2006) and XLM (Akyildiz et al., 2006).

- *MAC-CROSS protocol:* in this protocol, only a few nodes concerned with the actual data transmission are asked to wake-up, while other nodes that are not included on a routing path and hence are not involved in the actual transmission at all. In exchanging *RTS* and *CTS* packets, a field corresponding to a final destination address is added. The neighborhood nodes belonging to the path extend their wake-up time while other nodes prolong their sleep time.
- *XLM* (*Cross-Layer Module for Wireless Sensor Networks*) *protocol:* this protocol proceeds differently comparing to others traditional architecture based protocols for WSN. The communication in XLM is based on the *initiative* concept considered as the core of XLM and implicitly incorporates the intrinsic functionalities required for successful communication in WSN. A node starts a transmission by transmitting to his neighborhood an *RTS* packet to indicate that it has a packet to send. Upon receiving an *RTS* packet, each neighborhood node decides to participate to the communication by determining an *initiative I* defined as follows:

$$I = \begin{cases} 1, \ if \begin{cases} \xi_{RTS} \geq \xi_{Th} \\ \lambda_{relay} \leq \lambda_{relay}^{Th} \\ \beta \leq \beta^{\max} \\ E_{rem} \geq E_{rem}^{\min} \end{cases} \\ 0, \quad otherwise \end{cases} \quad (1)$$

$\xi_{RTS}$: Signal Noise Ratio (*SNR*) received from *RTS* packet,
$\lambda_{relay}$: packets rate transmitted by relay and by node,
$\beta$: node's buffer occupation,
$E_{rem}$: node's residual energy.
Values in the right side of inequalities give us respective thresholds and initiative *I* is initialised to 1 if all conditions illustrated in (1) are satisfied.
1. The first condition ensures that reliable links are built for the communication.
2. The second and third conditions are employed by local control congestion. Second condition prevents congestion by limiting the transmitted traffic by a relay node, while the third condition ensures that no buffer overflow exists for this node.
3. The last condition ensures that the residual energy $E_{rem}$ of a node do not exceed a minimal threshold $E_{rem}^{\min}$.
Cross-layering functionalities of the XLM protocol are represented by the constraints defined in the initiative I of a node enabling it to carry out a local congestion control, hop by hop reliability and distributed operation.
The component of local congestion control of XLM ensures energy efficiency and a reliable communication. Results of performance evaluation revealed that XLM is better than one-layer protocols in terms of communication processing and implementation complexity considerations.

## 3. CL-MAC protocol presentation

CL-MAC (*Cross-Layer MAC*) can be added to MAC protocols class by exploiting interactions between adjacent layers in order to minimize all energy waste sources and decrease latency during a multi-hop routing of a delay sensitive traffic from a particular source node towards the Sink in a WSN. The MAC layer enables access to the medium with wake-up/sleep schedule. Before presenting CL-MAC, we consider the following suppositions:
- Typical utilization scenario: to make clear our contribution, Fig. 8 shows a typical example of WSN architecture in which CL-MAC must be used. When a delay sensitive event has been detected (for example the detection event of forest fire around the area covered by sensor node S1 in Fig. 8), the data related to this event can be considered as an urgent traffic and must be delivered quickly to the Sink. CL-MAC acts at the MAC layer and aims to reserve a low latency and energy efficient path to deliver this urgent traffic from the source node generating this traffic towards the Sink. We can distinguish two types of nodes included in this path, in addition to the Sink: 1) a sensor node that collects sensed data which must be sensitive to the end-to-end delay (node S1 in Fig. 8), 2) relaying nodes (like

A or B in Fig. 8) that are responsible only for relaying a delay sensitive traffic. In the scenario showed by Fig. 9, the path S1-A-B-C-D-E-F-Sink is an example of an urgent

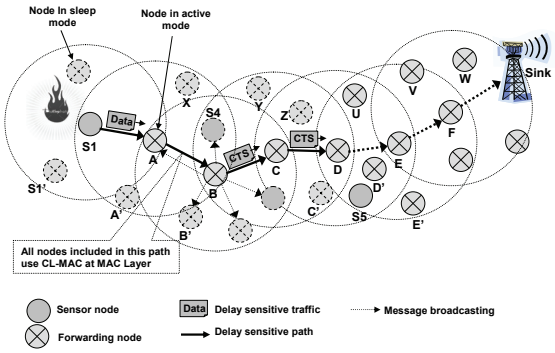



Fig. 8. Typical utilization scenario of CL-MAC

reserved path using CL-MAC. Each node included in this path like node B and its neighbors like nodes B' and S4 use CL-MAC at their MAC layer.

- The network is randomly deployed in a coverage area followed by a synchronisation phase.
- Nodes are locally and periodically synchronised like in Z-MAC protocol.

In the rest of the chapter, we refer to a delay sensitive traffic as '*urgent traffic*' or simply '*Data packet*' and to a delay sensitive path as '*urgent path*'.

In the following sections, we successively present the CL-MAC layer in the context of a cross layering approach, the new data structures of RTS and CTS control packets and their interpretation, the detailed algorithm for CL-MAC and its principal advantages and, finally, some other related details.

### 3.1 CL-MAC Layer

The neighborhood list established by each node contains information about neighboring nodes (identifier, position, and schedule table) and the routing table maintained by a routing agent in the network layer. The routing protocol used is based on the greedy approach, referred to as position-based routing, in which packet forwarding decision is achieved by utilizing location information about candidate nodes in the vicinity and the location of the final destination only. The distance-based Greedy forwarding scheme, proposed by Finn (Finn, 1987), has been adopted in our case. In this forwarding approach, a next hop node is the nearest neighbor node to the final destination (Sink). Only the nodes closer to destination than the current node are considered.

As illustrated in Fig. 9, CL-MAC operates at the MAC layer and the node implementing it can transmit two kind of unicast traffic: delay sensitive traffic generated locally (node implementing CL-MAC acts as a source node) or delay sensitive traffic received externally from a neighborhood (node implementing CL-MAC acts as a relaying node). When a source node acts as a relaying node and in the same time wants to transmit its local delay sensitive traffic, both kind of traffic can be transmitted towards the Sink.

CL-MAC exploits the routing information through a cross-layer design approach based on interaction strategy. In this strategy, CL-MAC interacts with the adjacent routing layer by using a simple Get/Store mechanism on a common memory storage dedicated to store the

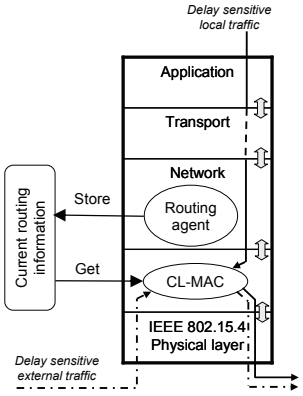



Fig. 9. CL-MAC : interaction between MAC and network layers

current routing information (see Fig. 9). This mechanism allows the routing agent to write in common memory information about the next hop forwarding node to which the current sender must forward a packet. This routing information can be obtained and exploited thereafter by CL-MAC to carefully perform a cross layering MAC operation. By knowing the next hop each time it advances in the routing path, CL-MAC can reserve a shared wireless medium during a time involved in each data communication.
In Fig. 9, we have proposed the IEEE 802.15.4 (IEEE, 2003) standard at the physical layer, which is dedicated at the origin for Low Rate-WPAN networks, but it is also very adapted for the WSN because it is designed for low-data-rate, low-power-consumption and low-cost applications.

### 3.2 New Data structure for RTS and CTS

Before describing further details about CL-MAC operations, one should be aware that weak modifications are made in *RTS* and *CTS* message structures without violating the IEEE 802.11 standard. The modified message structures of *RTS* and *CTS* illustrated by Fig. 10-(a) and (b) are proposed. The newly added field in RTS is the 'Next_Node_Adress', obtained by the sender's routing agent, and designates the address of the next node in the routing path to which packets must be transmitted. The new fields of CTS are 'Next_Node_Adress' and 'Sender_Adress'. The 'Next_Node_Adress' field has the same significance than 'Next_Node_Adress' of an RTS packet, but it is obtained by the routing agent of the receiver of the CTS packet. By specifying its address in a 'Sender_Adress' field of a sending CTS message, a node ignores receiving a CTS in which its address is specified as a previous node address. This change from the sender address to previous address has made in the receiver of the CTS message.
To make CL-MAC do that in a correct way, the Sink node address is supposed to be aware at the level of each node of the network in case of a mono-Sink WSN.

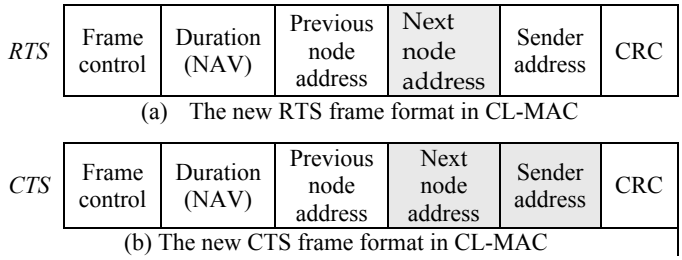



Fig. 10. Structure of *RTS* and *CTS* messages in CL-MAC ( added fields)

However, it is important to note that *CTS* and *RTS* messages have identical structure but the difference is in their interpretation by the receiver node. Thus, if the receiver is:
- *next node*: the message acts as an *RTS* sent by the sender.
- *previous node*: the message acts as a *CTS* coming from a transmitter as a response to an *RTS*.
- *any other node*: the message controls the behaviour of the node and forces it to switch to sleep mode.

In a wake-up period, a node continues to listen to the medium for a short period. Two cases can be considered if the medium is not allocated:
*Case 1*. The node has urgent data to transmit towards the Sink, then takes immediately possession of the medium and informs its neighboring nodes of that decision.
*Case 2*. Node has no data to transmit. In this situation, it turns off its radio in order to avoid energy dissipation in idle listening and overhearing situations. In this way, the node takes more time in sleep mode than in S-MAC protocol (½ frame + communication time) and stays in this mode until the next frame corresponding to the current schedule.
If the medium is allocated in this period, this means that a communication is occurring or that another node is trying to get medium control. In this case two situations can occur again:
- *Situation 1*. Another node tries to transmit packets. Then, the Backoff algorithm (Ignatius, 2006) is used to resolve this contention problem and makes possible for only one node (elected node) to obtain the control access rights. All other nodes enter sleep mode except the receiver of the packet which remains awake in order to communicate with the elected node.
- *Situation 2*. The node has no data to send. Then, if it is concerned with the current communication it remains in wake-up mode, otherwise it enters into sleep mode until the next frame.

A receiver node is identified by the sender which refers to its routing table that contains all the information related to the path between the sender and the Sink (distance, hops number of each alternative path, previous node identifier).

### 3.3 Algorithm for CL-MAC

Fig. 11 provides a detailed algorithm in order to implement the CL-MAC protocol at the level of each node. In case of urgent traffic our protocol forces all neighboring nodes, which are not selected for routing paths, to switch to sleep mode.

Fig. 12 illustrates the temporal behaviour of CL-MAC corresponding to the routing path (S1-A-B-C-D-E-F-Sink) depicted by Fig. 8. In Fig. 12, the data transferring follows a unidirectional path from a source node (S1) to the Sink. This node (S1) begins by transmitting an RTS control packet to its next hop (A). When this control packet has been successfully received, node (A) replies with a CTS control packet back to sender (S1). All neighbors of the sender and the receiver turn off their radio to save energy (nodes S1' and A in Fig. 8) except the next hop of the receiver that stays active (node B in Fig. 8). When receiver (A) is ready to receive the DATA packet (i.e. after it receives a CTS control packet from its next hope (B) which is interpreted as an confirmation of the transmission success of its CTS), sender (S1) exchanges a DATA/ACK packets with it. From node A, the forwarding process of CTS packets continues between the nodes included in the routing path (nodes B, C, D in Fig. 8) until reaching the Sink. This process rapidly advances towards de Sink and aims to reserve nodes belonging to the path by remaining them active and by putting into sleep mode all corresponding neighbors. The process of exchanging DATA/ACK packets between active nodes advances and follows the reservation process. To each DATA/ACK communication corresponds a transmission time during which each neighbor can be switched into sleep mode for energy saving. For example *t1* is a necessary duration during which node S1 can transmit RTS/CTS/DATA/ACK packets to its next hop A.

The algorithm for CL-MAC, given by Fig. 11, implements at each node these two processes in addition to a wake-up/sleep mechanism required for energy saving purposes. The idea behind a sleeping mechanism on which our protocol CL-MAC is based enables to eliminate all sources of energy waste described in section 1. This makes our protocol suitable to a

**ALGORITHM FOR CL-MAC**

```
1:  Input :  Table; //Routing table
2:  Begin
3:  Build-a-neighbor-list (Liste) ;
4:  Synchronize-the-schedule;
5:  Get-routing-table(Table) ;  //By recovering routing information stored
                                //by the routing agent
6:  Label1: Nav = 0;
7:  State = WakeUp;
8:  Repeat
9:  If There-is-data-to-send and (State = WakeUp) Then
10:   If Channel-is-free Then
11:    If Other-nodes-want-to-access-to-medium Then
12:        Backoff-Procedure-to-resolve-contention-problem;
13:    EndIf
14:     Destination = Get-destination-from-Table(Table);
15:     Send RTS to Destination;
16:     State = Wait-for-CTS;
17:   Else
18:     Put-into-Sleep-mode-and-WakeUp-next-frame ;  Goto
         Label1 ;
19:   EndIf
20: Else
21:   If NOT(Channel-is-free) Then
22:     Receive (message);
23:     If (message→destination = ID) Then // ID of the receiver
                                           //node of the packet
24:       Case Type-of(message) Of
25:         'RTS' : Build-and-send (CTS); State = Wait-for-
                    DATA;
26:        'DATA' : Build-and-send-to-source (ACK);
27:                 Rebuild-and-send-DATA-to-next-hop (DATA);
28:                 State = Wait-for-ACK;
29:         'ACK' : Put-into-Sleep-mode ;
30:                 Goto Label1;
31:         'CTS' : If (State = Wait-for-CTS) Then
32:                   Send (Data); State = Wait-for-ACK;
33:                 Else If State ≠ Wait-for-DATA Then
34:                        Rebuild-and-send (CTS)
35:                        State = Wait-for-DATA ;
36:                      EndIf
37:                 EndIf
38:       EndCaseOf
39:     Else
40:       Nav = message→duration;
41:       Put-into-Sleep-mode ; Goto Label1;
42:     EndIf
43:   Else
44:     Put-into-Sleep-mode-and-WakeUp-next-frame; Goto Label1;
45:   EndIf
46: EndIf
47: Until (Battery-level < Threshold) ;
48: Fin CL-MAC
```

Fig. 11. Detailed algorithm for CL-MAC at each sensor node

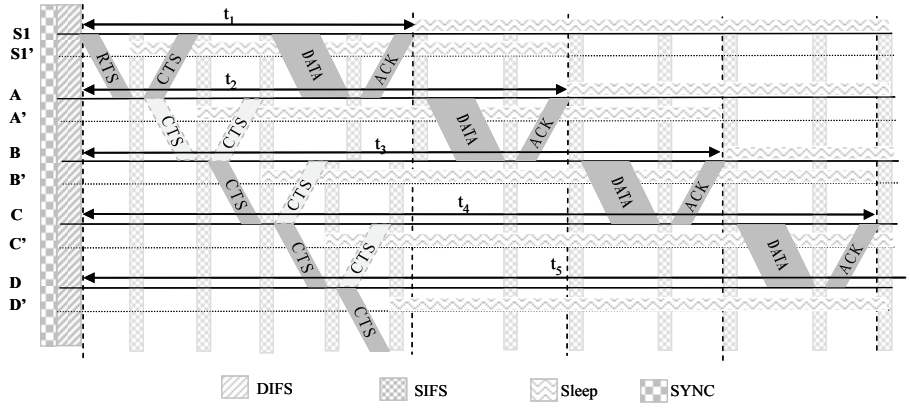



Fig. 12. Temporal behaviour of CL-MAC

very large spectrum of applications where the guaranty of the end-to-end delay transmissions and, at the same time minimum energy expenditure, is very challenging.
Lines 11 and 12 address a case of contention problem when, for example, two RTS control packets collide in case two source nodes want to transmit a DATA packet at the same time. If this problem happens, CL-MAC uses a backoff procedure to resolve this problem.
There are different ways to deal with a sleep schedule in the algorithm:
- Sleep schedule after the reception of an ACK control packet (line 29) resulting from a success of the DATA transmission. After a sleeping period, a node wake-up according to a current sleep schedule.
- Sleep schedule indicated by NAV of a neighbor node not included in the urgent routing path (line 41).
- Finally, sleep schedule of each node not concerned by the urgent communication and that has nothing to send (line 44). In this case, the waking up is planned to be happen according to the current sleep schedule.

Each sensor node executes the CL-MAC algorithm as long as the battery level does not reach a threshold.

### 3.4 Advantages of CL-MAC

Fig. 13 and 14 illustrate the main advantage of the CL-MAC protocol comparing to others MAC-protocols like S-MAC and MAC-CROSS.
As explained above, in the S-MAC protocol, a frame is divided into two fixed periods: one for a listening which causes a useless loss of energy consumption and another for sleeping. In the listening period, the control messages *CTS*/*RTS* are exchanged. The CSMA/CA mechanism is used for packet transmission in order to announce to a source node the next data transmission: for example, source node *A* in Fig. 13. It is clear that a node like *B'*, not concerned by the communication (between nodes *A* and *B*), must enter into sleep mode during all the communication time or after, according to the listening period fixed by the protocol. Just after the end of the communication, node *B* can starts another communication

without waiting until the expiration of sleep period, e.g. waiting for the next frame. This was adopted for designing an adaptive S-MAC protocol. In each frame, the MAC-CROSS

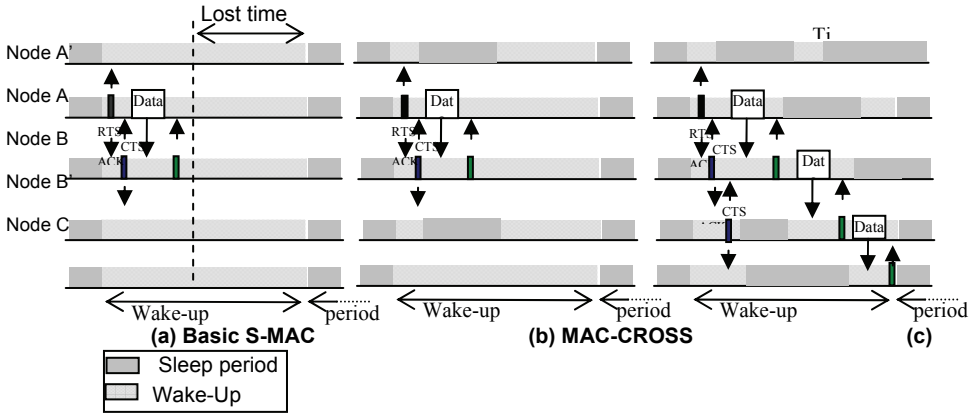



Fig. 13. Main functionalities of S-MAC (a) MAC-CROSS (b) and CL-MAC (c)

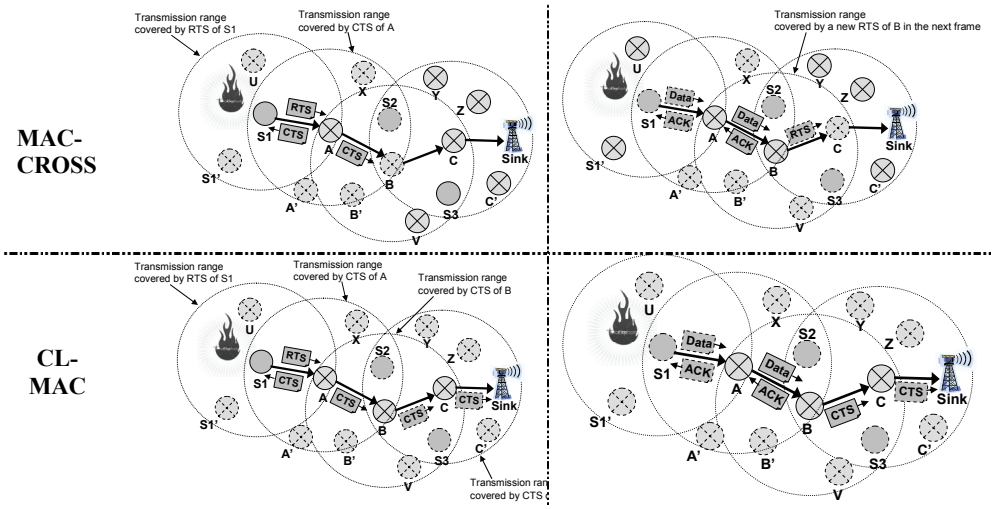



Fig. 14. Main advantages of CL-MAC protocol

protocol enables the exchange of *RTS/CTS/DATA/ACK* messages between three consecutive nodes using routing information obtained from the routing agent in network layer, while the CL-MAC does the exchange of the same messages but only at the beginning of each frame. Then, the third node transmits however directly the received *CTS* to the next node in the path. This *CTS* message is interpreted in case of CL-MAC by a receiver node as a *RTS* of normally the next frame in MAC-CROSS protocol, for each routing node to another in the path until arriving to the Sink during the same frame. This allows all nodes belonging to the routing path to remain active (or reserved) until each node transmits successfully data packets to the next node. Just after transmitting these data packets, the node either enters

into sleep mode or prepares to begin a new transmission frame as a new source if it has data to send.

### 3.5 Other details about CL-MAC

Several aspects of CL-MAC must be studied for showing all of its advantages. Among these aspects, lies the problem of the fragility of the path being reserved for a long time (long frame) and the coexistence of several urgent paths sharing a routing path. We address these two aspects in the following, with illustrations through some scenarios.

*Reservation and release of the routing path:* Fig. 15 illustrates that the nodes of the routing path reserved for the transmission of data packet (for example nodes S1, A, B) are released when the urgent traffic advances by three hops. They can therefore participate in other communications without waiting until the communication completely finishes between S1 and the Sink. For example, node A' can accept a traffic transmitted by one of its neighbors to relay it to another node, whereas the data packet transmitted by S1 did not reach a Sink yet (it is received by node E in Fig. 15). This shows that the routing path cannot be reserved for long time (long frame) when CL-MAC is applied and therefore the problem of the fragility of the path is consequently excluded.

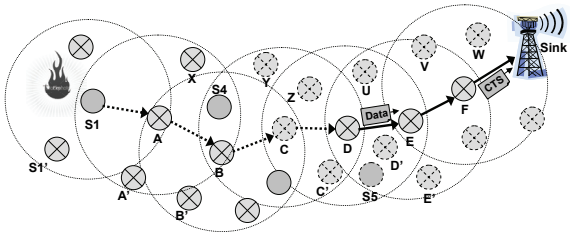



Fig. 15. Reservation and release of the routing path in CL-MAC

*Coexistence of several urgent paths: a*s depicted in Fig. 16, a new urgent path which starts from S5 towards the Sink may include nodes E and F that can be reserved to be used in an another urgent path starting from S1. The reservation of the path S1-A-B-C-D-E-F-Sink (S1-Sink for short) is in progress and before the CTS packet reaches the node E, S5 transmits RTS packet to node E that means it initiates a new communication towards the Sink. In this case, the path S1-Sink may be blocked at the level of node C, because its neighbor D (next hop) can be in sleep mode when node C wishes to communicate with it. This can be accepted in the context of typical applications in which a nearest urgent event can be detected and received before another urgent event far away which was started in the first place. The nearest event (detection of forest fire in our example such as sudden temperature rise) will immediately produce an alarm at the Sink for a fast intervention in the suspected nearest location, but also for dispatching, for example, a team to check the situation in all the sensed area. By doing this, the event far away will be detected even if the urgent message related to it can not be reached by the Sink.

## 4. Modeling CL-MAC protocol

In order to formally prove the correct operation of our protocol, it is important to model it in

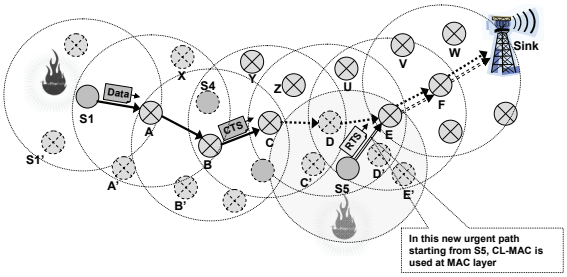



Fig. 16. Case of coexistence of two delay sensitive paths: S1-A-B-C-D-E-F-Sink and S5-E-F-Sink

a convenient mathematical model according to its specifications. The introduction of time Petri nets (Merlin, 1974; Berthomieu & Menasche, 1983) is motivated by their ability to model easily temporal constraints and the existence of a TiNA analyser tool for properties verification. Time Petri nets or TPN for short are a convenient model for real time systems and communication protocols (Berthomieu & Diaz, 1991).
TPN extend Petri nets by associating two values (min, max) of time (temporal interval) to each transition. The value min (min ≥ 0), is the minimal time that must elapse, starting from the time at which transition *t* is enabled until this transition can fire and max ($0 \leq \max \leq \infty$), denotes the maximum time during which transition *t* can be enabled without being fired. Times min and max, for transition *t*, are relative to the moment at which *t* is enabled. If transition *t* has been enabled at time α, then *t* cannot fire before α + min and must fire before or at time α + max, unless it is disabled before its firing by the firing of another transition.
Fig. 17 depicts the time Petri net model of CL-MAC protocol. The values in the intervals associated to transitions refers to relative time of transmitting packets (*RTS, CTS, DATA, ACK*) according to IEEE 802.11 standard (IEEE Std, 1999) and respected by our proposal.

### 4.1 Model hypotheses

The hypotheses on the behaviour of CL-MAC are as follows: we assume that *DIFS* duration = *SIFS* duration = 1 time unit, Control messages *RTS*, *CTS* and *ACK* consume 3 time units, DATA requires 10 time units for its transmission. Initially, only the places *p1*, *p8*, *p14* and *p17* are marked by one token.

### 4.2 Model explanation

The meanings of the transitions are given in Table 1. In Fig. 17, the sender part, modelling the node wishing to forward a packet to the Sink, is described by four transitions specifying the node behaviour: *t1* and *t2* to send *RTS* and *DATA* packets respectively at precise moments. These moments are indicated by the intervals associated with the transitions (an *RTS* packet is sent after a *DIFS* and *DATA* packet is sent after having received a *CTS* within 3 to 4 units of time, this is represented by the temporal interval associated to transition *t2*). Transition *t3* allows node to enter into sleep mode after having received the *ACK*, whereas

*t4* allows the token to remain in the place *p19* (sleep state), time needed before beginning the next communication.

The Receiver part reacts to packets transmitted by the network part. The transition *t5*, generating the *CTS* control packet, is activated once the token, coming from the firing of transition *t1*, takes place in *p2*. It is allowed to fire at the moment $3 \leq \tau \leq 4$ (time for the

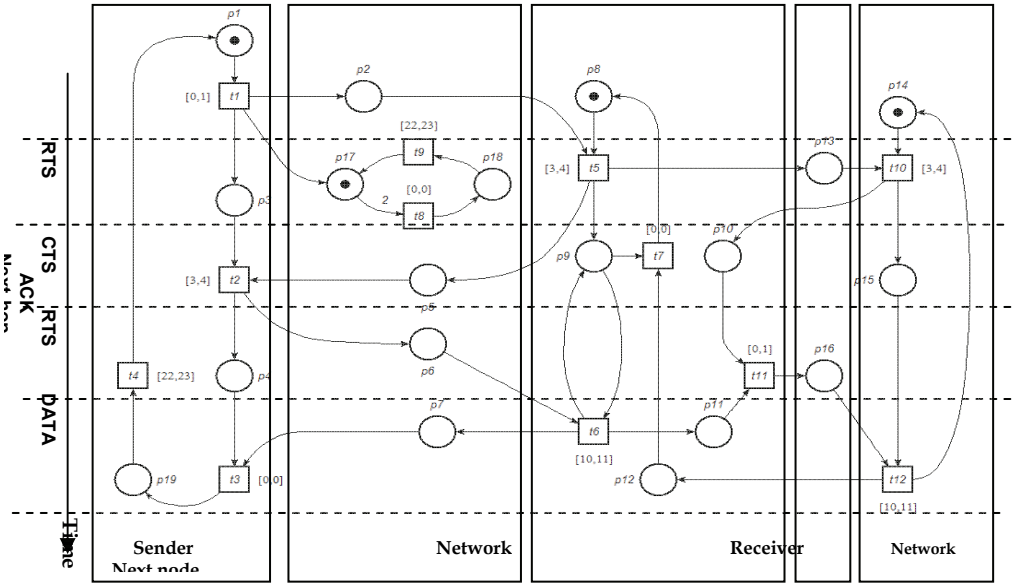



Fig. 17. Time Petri net model for CL-MAC protocol: columns represent temporal behaviour and lines represent spatial behaviour

| Transition | Meanings |
|---|---|
| t1 | Transmission of *RTS* packet in unicast mode but the nature of the medium acts as broadcast |
| t2 | Transmission of *DATA* packet in unicast mode |
| t3 | Sender node enters sleep mode |
| t4 | Awakening of the sender to start a new frame |
| t5 | Transmission of *CTS* packet in multicast mode by the receiver |
| t6 | Transmission of *ACK* packet in unicast mode after receiving *DATA* packet |
| t7 | Switching to sleep mode of the receiver after achieving his communication (frame) |
| t8 | Entering sleep mode of neighbor node not concerned by the current communication |
| t9 | Awakening of neighbor node not concerned by the current communication after the last one ends |
| t10 | Transmission of *CTS* packet by the next node |
| t11 | Transmission of *DATA* packet by the receiver to the next node |
| t12 | Entering sleep mode of the next node |

Table 1. Model transitions and their corresponding explanation

reception of *RTS* + *SIFS* time: 3, 3+1). In the same way, the transition *t6*, models the transmission of an *ACK* control packet after receiving *DATA* packet.
The Network part is modelled by two transitions *t8* and *t9* and a set of places {*p2, p5, p6, p7, p17, p18*}. The places *p2*, *p5*, *p6* and *p7* respectively represent the propagation of *RTS, CTS, DATA and ACK* packets in the wireless medium.
The part formed by the two transitions *t8, t9* and the places *p17*, *p18* models the vicinity of the sender and that of the receiver. A node which wakes up (a token in the place *p17*) and receives a packet which is not intended to it, immediately returns into sleep mode. This explains the labelling of the transition *t8* by the interval [0, 0]. The transition *t9* plays the role of *t4* for the neighbor.

## 5. Validation of CL-MAC protocol

The TPN proposed for CL-MAC has been validated by the TINA tool software (Berthomieu & Vernadat, 2006; Berthomieu et al., 2004; *http://www.laas.fr/tina*). In this section, a brief presentation of the TINA tool is introduced. The obtained results analysis are also given.

### 5.1 Presentation of the TINA tool

Tina (TIme Petri Net Analyzer) is a software environment to edit and analyze Petri Nets and Time Petri Nets. It is developed and maintained by a group of researchers of the LAAS/CNRS laboratory at Toulouse University – France (*http://www.laas.fr*). TINA is a powerful tool which allows the checking of many aspects of Petri Nets (bounded, deadlock, re-initialization). It is based on the intrinsic properties of Petri Nets and proposes in particular a functionality of formal validation which brings the mathematical proof that the studied property is checked with a confidence degree of 100%. The functionalities of TINA allow a temporal study of a TPN model based on the reachability analysis method for usual Petri nets. Before using TINA, other software tools in connection with Petri Nets were investigated, in particular HPSim (*http://www.winpesim.de*) and ARP (*http://www.ppgia.pucpr.br/~maziero/doku.php/software:arp_tool*). However, TINA was chosen for its capacities for TPN formal analysis.
Tina accepts input in graphical or textual formats, including PNML (an XML based exchange format for Petri nets).Transition system outputs can be produced for external checkers in a number of textual or binary formats.

### 5.2 Results analysis

TINA allowed us to validate the following properties of our CL-MAC TPN model:
*Boundedness property*: this property relates to the finite number of tokens in each place of a TPN for any marking reachable from an initial marking. A TPN is said to be safe if it is 1-bounded. This aspect must be checked in the first place. Otherwise the formal checking of others properties does not have any significance. The not bounded property is characterized by an infinite number of tokens in at least one place among other places of the TPN. In this case, we notice that the model diverges or the implemented system will require an abnormally high quantity of resources (memory, CPU time, etc).
*Liveness property*: this property allows the detection of portions of died code, i.e. the absence of liveness of some places and/or the blocking of some transitions from TPN, for any initial

and accessible marking from the network. The absence of liveness makes it possible to highlight portions of code which are never performed (thus to detect the modelling errors) and situations where the system modelled is likely to be blocked.

*Reversibility property*: the system re-initialisation supposes that the system finds its initial state (initial marking) on the basis of any other state during its operation. This property is fundamental to validate automata based systems which present a cyclic operation.

These three properties of our model have been successfully validated by TINA. In the following, we present the classes generated by the TINA tool and the reachability graph formed by these classes. From these results, the validation of these properties has been proved.

*Classes: t*he following classes (*C0* to *C18*) are generated by TiNA tool during the analysis phase:

*class 0 p1 p14 p17 p8, 0 <= t1 <= 1*
*class 1 p14 p17*2 p2 p3 p8, 3 <= t5 <= 4, 0 <= t8 <= 0*
*class 2 p14 p18 p2 p3 p8, 3 <= t5 <= 4, 22 <= t9 <= 23*
*class 3 p13 p14 p18 p3 p5 p9, 3 <= t2 <= 4, 18 <= t9 <= 20, 3 <= t10 <= 4*
*class 4 p10 p15 p18 p3 p5 p9, 0 <= t2 <= 1, 14 <= t9 <= 17, t2 - t9 <= ~14*
*class 5 p10 p15 p18 p4 p6 p9, 10 <= t6 <= 11, 14 <= t9 <= 17*
*class 6 p10 p11 p15 p18 p4 p7 p9, 0 <= t11 <= 1, 0 <= t3 <= 0, 3 <= t9 <= 7*
*class 7 p15 p16 p18 p4 p7 p9, 10 <= t12 <= 11, 0 <= t3 <= 0, 3 <= t9 <= 7*
*class 8 p15 p16 p18 p19 p9, 10 <= t12 <= 11, 22 <= t4 <= 23, 3 <= t9 <= 7*
*class 9 p15 p16 p17 p19 p9, 3 <= t12 <= 8, 15 <= t4 <= 20, t12 - t4 <= ~11, t4 - t12 <= 13*
*class 10 p12 p14 p17 p19 p9, 11 <= t4 <= 13, 0 <= t7 <= 0*
*class 11 p14 p17 p19 p8, 11 <= t4 <= 13*
*class 12 p10 p11 p15 p18 p19 p9, 0 <= t11 <= 1, 22 <= t4 <= 23, 3 <= t9 <= 7*
*class 13 p15 p16 p18 p19 p9, 10 <= t12 <= 11, 21 <= t4 <= 23, 2 <= t9 <= 7, t4 - t9 <= 20, t9 - t4 <= ~15*
*class 14 p15 p16 p17 p19 p9, 3 <= t12 <= 9, 15 <= t4 <= 20, t12 - t4 <= ~10, t4 - t12 <= 13*
*class 15 p12 p14 p17 p19 p9, 10 <= t4 <= 13, 0 <= t7 <= 0*
*class 16 p14 p17 p19 p8, 10 <= t4 <= 13*
class 17 p13 p14 p18 p4 p6 p9, 0 <= t10 <= 1, 10 <= t6 <= 11, 14 <= t9 <= 17, t10 - t9 <= ~14
class 18 p10 p15 p18 p4 p6 p9, 9 <= t6 <= 11, 14 <= t9 <= 17, t9 - t6 <= 7

These classes represent the chronological temporal behavior of CL-MAC by using the firing mechanism of the corresponding time Petri net.

- *Reachability graph :* Fig. 18 depicts the reachability class graph of time Petri net representing the previous classes. This analysis reveals that the CL-MAC protocol has effectively the good properties previously mentioned:

- *Liveness*: the net is deadlock freeness and each transition is always able to be fire infinitely.
- *Boundedness*: the number of tokens in every place is limited to one token except the place *p17* that is 2-bounded.
- *Reversibility*: the return of the TPN to its initial state shows that the CL-MAC TPN model is reversible (able to be reinitialized).

Fig. 19 summarizes some results given by *Net Draw Control* of the TiNA tool.

## 6. Evaluation of the CL-MAC protocol

Performance evaluation of our protocol includes some metrics such as energy and latency which must be provided in some typical applications as metrics of performance. Because of the lack in practice of simulation tools dedicated to Cross-layering design approach based protocols, we have developed our home simulator software using the C++ Builder programming language. Using this software, we have implemented and compared the CL- MAC protocol described by the algorithm given by Fig. 11 with the MAC-CROSS and S-

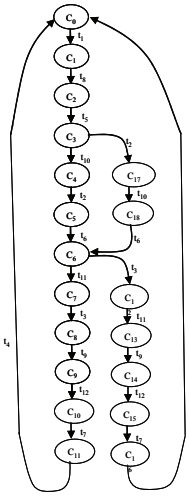


Fig. 18. Reachability class graph

```
LIVENESS                                                    ANALYSIS
.....
Possibly live
0 dead classe(s), 19 live classe(s)
0 dead transition(s), 12 live transition(s)
STRONG CONNECTED COMPONENTS:
0: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
SCC GRAPH:
0 -> t1/0, t8/0, t5/0, t10/0, t2/0, t6/0, t11/0, t3/0, t9/0, t12/0, t7/0,
t4/0
0.000s
```

Fig. 19. Analysis results given by TiNA tool

MAC protocols presented above. This comparison is justified by the fact that : in one hand MAC-CROSS is considered as a basis of our work and in the other hand S-MAC is regarded as a reference chosen by the community of researchers for studying energy efficient MAC layer issues for WSN. All experimentations have been performed according to the scenario given by Fig. 9 in which one or several sources transmit delay sensitive traffic towards the Sink.

### 6.1 Simulation Environment

An example of deployed WSN network generated by our simulator and used for CL-MAC evaluation is illustrated by Fig. 20 and simulation parameters are summarized in Table 2. The latency (μ second) used in our experimentations is defined as an elapsed time between the time of message sending by a source node and the time of arrival of this message to the final destination (Sink). In order to compute the energy consumed during each data transmission from the source node to the Sink, we have used a first order energy model introduced by Heinzelman et al. (2000).

### 6.2 Performances analysis

The aim of simulation is to analyze the effect of the variation of some parameters such as the number of data sources, density and hop number on CL-MAC, MAC-CROSS and S-MAC behaviour in terms of dissipated total energy and latency in the network. Fig. 21 shows that if the number of data sources increases in the network, the total energy consumed by MAC-CROSS and S-MAC increases more quickly than that consumed by CL-MAC. But, more than 43 active data sources simultaneously relative to MAC-CROSS and 46 relative to S-MAC, make our protocol consuming more energy. This can be explained by the fact that each time a data source is added to the network, more nodes will be mobilized to remain in

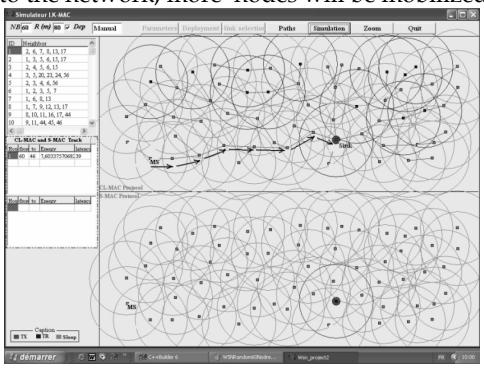



Fig. 20. Example of WSN with 60 sensor nodes. This simulation interface snapshot is divided into two windows: the top window for CL-MAC and the bottom window for S-MAC, the left part of the screen is reserved for simulation traces.

| Parameter Type | Test Value |
|---|---|
| Number of sensor nodes (in the same deployment space) | Changes according to the evaluation example |
| Number of Sink | 01 |
| *RTS*/*CTS* Message size | 118 bits |
| *ACK* message size | 112 bits |
| *Data* message size | 800 bits |
| Throughput | 8 bits/µs |

Table 2. Simulation parameters

active state (thus their duty cycle increases) in order to participate in routing paths as intermediate nodes. Thus, there will be fewer nodes in sleep state (but the effectiveness of our protocol is based on its ability to put into sleep mode any node in the vicinity not concerned by the routing operation). On the other hand, S-MAC has the capability to put into sleep mode each sensor node for a half cycle independently of the number of source nodes. Therefore, no changes will be made in the node's duty cycle. MAC-CROSS behaves like S-MAC when traffic loads are heavy, except that MAC-CROSS begins a new frame after each two successive hops.

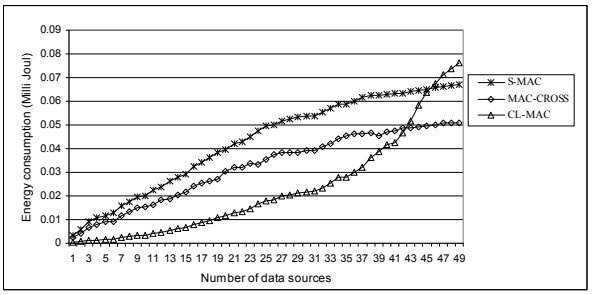



Fig. 21. Energy consumption vs the number of data sources with 60 deployed nodes

In the experimentation producing the results of Fig. 22, we have varied the density of the network in a fixed deployment space and we have chosen a delay sensitive data source located at 10 hops far from the Sink. The data source sends data packets to the Sink on the routing path. Because the number of neighbors of each node belonging to the path is variable, we have chosen the average number of these neighbours in numerical evaluation of total energy consumed by the network. In Fig. 22, CL-MAC always maintains its best performance as a protocol with a minimum energy consumption compared to MAC-CROSS and S-MAC. The latter consume more energy if the network size increases.

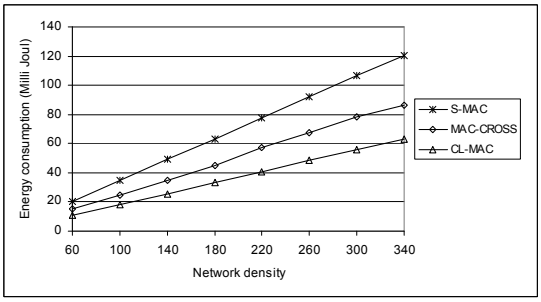



Fig. 22. Energy consumption in case of varying the network density

Fig. 23 shows well that latency increases quickly in the case of CL-MAC when a new source node of urgent traffic is added to the number of nodes already in communication. On the other hand we note, for a given number of sources lower than 3 relative to MAC-CROSS and 4 relative to S-MAC, that CL-MAC records latency was better than MAC-ROSS's and S-MAC's. This can be explained by the nature of the progressively generated urgent paths. Indeed, in the presence of not disjoint paths (paths having at least one intermediate node in common), the CL-MAC protocol leaves in active mode only one node included in a routing path and puts the remainder of neighbor nodes into sleep mode. However, if this active node is included in several routing paths, then the only path, which will be operational, is the one that will have possession of this active node. The other paths will be delayed until the current routing step at this active node is finished. More precisely, the neighboring nodes of the active node, that have been included in other routing paths, will be found in sleep mode at the moment when the nodes, which precede them in their

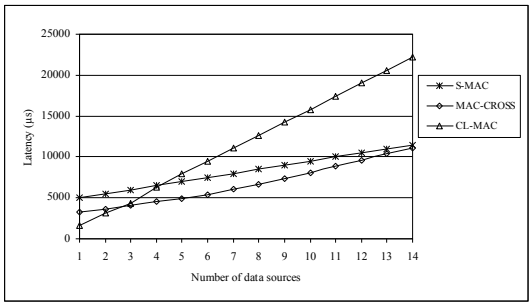



Fig. 23. Latency vs number of data sources with 150 deployed nodes and hop number=10

corresponding paths, want to transmit a message to them. It is consequently clear that in the case of disjoint paths that have no common neighbors, CL-MAC maintains its performances. However, the guaranty of the presence of these disjoint paths in the network from many sources is in practice a very challenging issue. We can see again that MAC-CROSS converge to S-MAC when this number of data sources is greater than 14. This is due, as we said previously, to the fact that when traffic loads are heavy, MAC-CROSS behaves like S-MAC.
The results of Fig. 24 are obtained by carrying out 5 transmissions of data packets for each hop where the resulting average value is considered. We note in this figure that each time a sensor node moves away from the Sink, the CL-MAC protocol records a better latency compared to MAC-CROSS and S-MAC. Indeed, while moving away from the Sink, the routing path will contain more sensor nodes participating in the routing operation. Therefore, according to the strategy on which is founded CL-MAC, there will be more potential sensor nodes able to switch into sleep mode. This explains the gain in term of latency in presence obviously of only one routing path at a given time.

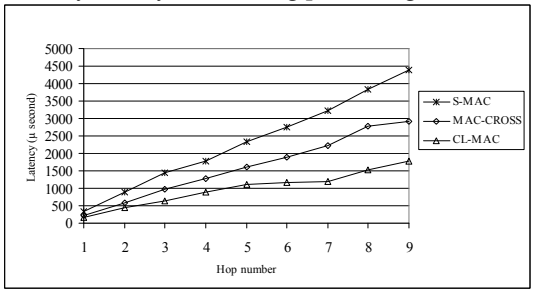



Fig. 24. Latency vs hop number with 60 deployed nodes

## 7. Conclusion and future work

In this chapter, we have proposed a novel energy efficient and low latency MAC protocol for delay sensitive traffic in WSN using a Cross layer optimization approach, named CL-MAC. In order to prove the correctness of its operation, we have modelled it using time Petri nets and some related properties like liveness, boundedness and reversibility have been validated analytically using the TiNA tool. Various experiments that we have performed show that CL-MAC outperforms MAC-CROSS and S-MAC in term of energy

saving and low latency in the following cases: 1) when the data source is far from the Sink, and 2) when, at a given time, the number of data sources is low. However, CL-MAC described in this chapter acts to enable energy saving and low latency in each urgent path only. But in default operation mode (absence of delay sensitive traffic) nodes must use an adequate MAC protocol to deliver a normal traffic (for example the ambient temperature measured periodically) from sources to the Sink using best effort paths. CL-MAC must be extended to take into account this second kind of traffic. We propose as future work an Adaptive Cross-Layer MAC protocol (ACL-MAC) that could be able to differentiate and manage these two types of traffic.

## 8. References

Akyildiz, I. F. & Ismail, H. K. (2004) 'Wireless sensor and actor networks: research challenges', Georgia Institute of Technology, *www.sciencedirect.com*.

Akyildiz, I. F.; Vuran, M. C. and Akan, O. B. (2006) 'A Cross layer protocol for wireless sensor networks', *Proc. Conference on Information Sciences and Systems (CISS'06)*, Princeton, NJ.

Bachir, A.; Barthel, D., Heuss, M., Duda, A. (2006) 'Micro-Frame Preamble MAC for Multihop Wireless Sensor Networks', IEEE ICC2006 Proceedings.

Berthomieu, B.; Vernadat, F. (2006) 'Time Petri Nets Analysis with TINA', tool paper, *In Proceedings of 3rd Int. Conf. on The Quantitative Evaluation of Systems (QEST 2006), IEEE Computer Society.*

Berthomieu, B.; Ribet, P.O., Vernadat, F. (2004) 'The tool TINA - Construction of Abstract State Spaces for Petri Nets and Time Petri Nets', *International Journal of Production Research,* Vol. 42, no 14.

Berthomieu, B. & Diaz, M. (1991) 'Modeling and Verification of Time Dependent Systems Using Time Petri Nets', *IEEE Trans. On Soft. Eng.*,Vol. 17, n° 3, pp. 259-273.

Berthomieu, B. & Menasche, M. (1983) 'An Enumerative Approach for Analyzing Time Petri Nets', *IFIP Congress Series*,Vol. 9, pp. 41-46 North Holland.

Buettner, M.; Gary Y., Eric, A. and Richard, H. (2006) 'X-MAC: A Short Preamble MAC Protocol For Duty-Cycled Wireless Sensor Networks', *Proceedings of the 4th international conference on Embedded networked sensor systems*, pp.307-320.

Cheng, H.; Qin, L. and Xiaohua, J. (2006) 'Heuristic Algorithms for Real-time Data Aggregation in Wireless Sensor Networks', School of Computing, Wuhan University, China. *IWCMC'06*, July 3–6.

Demirkol, I.; Ersoy, C., Alagoz, F. (2006) 'MAC protocols for Wireless Sensor Networks: a survey', *IEEE Communications Magazine*, pp.115-121.

Dewasurenda, D.; Mishra, A. (2005) 'Design Challenges in Energy-Efficient Medium Access Control for Wireless Sensor Networks', *Handbook of Sensors Networks: Compact Wireless and Wired Sensing Systems*, Edited by Mohammad Ilyas and Imad Mahgoub, CRC Press.

EL-Hoiydi, A. (2002) 'Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks', *In IEEE International Conference on Communications (ICC)*, New York.

Finn, G. G. (1987) 'Routing and addressing in large metropolitan-scale internetworks', *ISI Research Report*, ISU/RR-87–180.

Halkes, G.P.; Van Dam, T., Langendoen, K.G. (2005) 'Comparing Energy Saving MAC Protocols for Wireless Sensor Networks', *Mobile Networks and Applications 10*, pp.783-791, Springer Science.

Heinzelman, W. R.; Chandrakasan, A. P., Balakrishnan, H. (2002) 'An application-specific protocol architecture for wireless microsensor networks'. *IEEE Transactions on Wireless Comm.*, 1 (4): 660–670.

Heinzelman, W. R.; Chandrakasan, A., Balakrishnan, H. (2000) 'Energy-Efficient Communication Protocol for Wireless Microsensor Networks', *The Hawaii International Conference On System Sciences*, January 4-7, MAUI, HAWAII.

Holger, K.; Marc, L. and Tim, N. (2003) 'A Data Aggregation Framework for Wireless Sensor Networks', *European research project EYES*, Berlin.

IEEE 802.15.4, (2003) Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), Standard, IEEE.

IEEE Std. 802.11- (1999), *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, ISO/IEC 8802-11:1999(E), IEEE Std. 802.11.

Ignatius, M. (2006) 'Energy-efficient Wireless Sensor Network MAC Protocol', *PhD Thesis in Electrical Engineering*, March 31, Faculty of Virginia Polytechnic Institute and State University.

Injong, R.; Ajit, W., Mahesh, A. and Jeongki, M. (2005) 'Z-MAC: an Hybrid MAC for Wireless Sensor Networks', *Proceedings of the 3rd international conference on Embedded networked sensor systems*, San Diego, California, USA, pp.90-101.

Karn, P. (1990) 'MACA - A new channel access method for packet radio'. In ARRL/CRRL Amateur Radio 9th Computer Networking, pp. 134–140.

Koen, L. & Gertjan, H. (2004) 'Energy-Efficient Medium Access Control', The embedded Systems Handbook, Delft University of Technology, Faculty of Electrical Engineering, *Mathematics and Computer Science Mekelweg 4*, 2628CD Delft, The Netherlands.

Lu, G.; Krishnamachari, B. and Raghavendra, C.. (2004) 'An adaptive energy-efficient and low-latency MAC for data gathering in sensor networks', *In Int. Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, Santa Fe, NM.

Merlin, P. M. (1974) 'A Study of the Recoverability Systems of Computing', Irvine Univ. California, *PhD thesis*.

Muneeb, A.; Umar, S., Adam, D., Thiemo, V., Kay, R., Koen, L., Joseph, P., Zartash, A. U. (2006) 'Medium Access Control Issues in Sensor Networks', *ACM SIGCOMM Computer Communication Review*, Vol.36.No.2.

Polastre, J.; Hill, J. and Culler, D. (2004) 'Versatile low power media access for wireless sensor networks', in *SenSys'04*, pp.12-24.

Rajendran, V.; Obraczka, K., Garcia-Luna-Aceves, J. J. (2003) 'Energy-efficient, collision-free medium access control for wireless sensor networks', *In: Proc. ACM SenSys 03*, Los Angeles, California.

Sohraby, K.; Minoli, D., Znati, T. (2007) 'Medium Access Control Protocols for Wireless Sensor Networks', *Book chapter in Wireless Sensor Networks : Technology, Protocols and Applications*, Wiley-Interscience, pp.142-173.

Sohrabi, K.; Gao, J., Ailawadhi, V., Pottie G. J. (2000) 'Protocols for Self-Organization of a Wireless Sensor Network', *IEEE Personal Communications*, Vol. 7, No. 5, pp. 16–27.

Sohrabi, K.; Pottie, G. J. (1999) 'Performance of a Novel Self- organization Protocol for Wireless Ad Hoc Sensor Networks', *Proceedings of the IEEE 50th Vehicular Technology Conference (VTC'99)*, pp. 1222–1226.

Suh., C.; Young-Bae, K. and Dong-Min, S. (2006) 'An Energy Efficient Cross-Layer MAC Protocol for Wireless Sensor Networks', Graduate School of Information and Communication', Ajou University, Republic of Korea, *APWeb 2006*, *LNCS 3842*, pp. 410-419.

Wei, Y.; Heidemann, J., Estrin, D. (2002) 'An energy-efficient MAC protocol for wireless sensor networks', Proceedings *of INFOCOM 2002, Twenty-First Annual Conference of the IEEE Computer and Communications Societies*,. IEEE.

Wei, Y.; Heidemann, J., Estrin, D. (2004) 'Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks', *IEEE/ACM Transactions on Networking*, Vol. 12, No. 3.

Willig, A. (2006) 'Wireless Sensor Networks: concept, challenges and approaches', *Elektrotechnik & Informationstechnik*, pp.224-231.

Zhi-Wen, O.; Shruthi, B. K. and Sang S. K. (2005) 'Medium Access Control for Wireless Sensor Networks', *CS258 - Advanced Communication Networks*, San Jose State University.

11

# Distributed Implementation of Petri nets in Control Applications

Ramon Piedrafita, Danilo Tardioli and Jose Luis Villarroel
*Department of Computer Science and Systems Engineering, University of Zaragoza*
*Spain*

## 1. Introduction

In this chapter, we propose a platform for the implementation of distributed discrete event control systems. We further propose a new architecture for the distributed implementation of Petri nets (PN) in control applications. The architecture covers the local execution of PN and the communication among controllers. Both synchronous and asynchronous communication paradigms are supported. A framework of classes for supporting the real-time communication and real-time execution or remote calls has been developed. The Real-time Wireless Multi-hop Protocol has been adopted. With this protocol and the framework of classes developed, control deadlines of the system can be guaranteed. In order to ensure the access to common resources, a new class of communication places, the Auction Communication Places, is proposed. To support the real-time and concurrent characteristics, we have used the Java real-time specification. In order to implement PN, we have extended the concurrent coordinators technique, developed in previous works, to distributed systems introducing the distributed coordinators. To demonstrate the practical utility and feasibility of the architecture, we have applied it to the control of a flexible manufacturing cell.
A software implementation of a PN is a program that triggers the firing of the net transitions observing the marking evolution rules, that is, it plays the token game. Depending on the criteria, a Petri net implementation can be classified as compiled or interpreted, sequential or concurrent, and centralized or decentralized. An implementation is interpreted if the net structure and the marking are codified as data structures. The data structures are used by one or more tasks called interpreters to establish the net evolution. The interpreters do not depend on the implemented net. A compiled implementation is based on the generation of one or more tasks whose control flow corresponds to net evolutions.
In previous works (Piedrafita and Villarroel 2006 a) a centralized control architecture and a development environment based on Petri nets and Java language was developed. In the present work, this control architecture is extended to be applied to distributed control systems. In this work we develop the distributed implementation of PN. To achieve this, we have developed a new technique called distributed coordinators, and to solve conflicts between distributed processes we propose a new technique: the Auction Communication Place.
In this technique, the execution of the coordinators takes place in several hosts, connected by a real-time communication network. In each host one or more coordinators executes its Petri net. The coordinators read the inputs, execute its Petri net (the token player) and write the

outputs, i.e. achieving local machine control. For the necessary information interchange between hosts, the implementation is completed using the necessary communication mechanisms.
The partition into two or more subnets leads to the appearance of shared places among subnets. These places are known as communication places. The communication places can be shared between Petri nets of the same host, or between Petri nets of different hosts. A communication place is implemented in one of the hosts and belongs to the PN executed in the machine in question.
A Middleware that allows the implementation of distributed PN has been developed. For this purpose, a set of Java Classes to implement synchronous and asynchronous communication paradigms has been designed. These classes implement the local and remote communications places, the auction communication place and the auction solver.
To implement the communication between the different coordinators, a protocol that supports real time traffic over Ethernet is needed. In our case the protocol RT-WMP (Tardioli andVillarroel 2007) developed at the University of Zaragoza, has been chosen.
The RT-WMP is a novel protocol that supports hard real-time traffic. In this sense in RT-WMP, end-to-end messages delay has a bounded and known duration and it manages global static priorities of messages. The protocol has been designed to connect a relatively small group of nodes. It is based on a token-passing scheme. RT-WMP has a built-in error recovery mechanism that can recover from certain types of errors without jeopardizing the real-time behaviour. The RT-WMP can run over commercial hardware without modifications.
The practical application of the technologies presented in this paper were tested using a flexible manufacturing cell installed at the Department of Computer Science and Systems Engineering at the University of Zaragoza, for research and teaching purposes.
We have installed a new distributed control system based on several computers, connected by communications networks, and a fieldbus. One of the computers is equipped with a fieldbus master card, where the input/output modules of the machines are connected. The rest of computers control the input/output hardware by Industrial Ethernet.
This chapter is organized as follows: in Section 2, we review the different techniques used to implement PN. Section 3 describes distributed implementation of PN and in Section 4 the characteristics of the RT-WMP based implementation are indicated. The application over the flexible manufacturing cell is described in Section 5. In Section 6, we present conclusions and suggest future lines of research.

## 2. Software Implementation of Petri Nets

A Petri net is a graphical, mathematical tool that can be used to model discrete event systems (Murata 1989) . PNs are bipartite graphs that have two types of nodes: places and transitions, which are represented by circles and bars respectively. Directional arcs connect places and transitions. Input arcs connect places with transitions, and output arcs connect transitions to places. Places contain a positive or null number of tokens. The token distribution over PN places is called marking and represents the system state. Transitions represent events. A transition firing models the occurrence of an event that changes the system state (the marking of the PN).

In the last 25 years, researchers have devoted considerable attention to the software implementation of PN; see for example (Taubner 1988), (Bruno and Marchetto 1986), (Brams 1982), (Briz and Colom 1994), and (Colom, Silva et al. 1986). A software implementation is a program that triggers the firing of the net transitions observing the marking evolution rules, that is, it plays the *token game*. Depending on the criteria, a Petri net implementation can be classified as compiled or interpreted, sequential or concurrent, and centralized or decentralized. An implementation is *interpreted* if the net structure and the marking are codified as data structures. The data structures are used by one or more tasks called interpreters to establish the net evolution. The interpreters do not depend on the implemented net. A *compiled* implementation is based on the generation of one or more tasks whose control flow corresponds to net evolutions.
In a *centralized* implementation, the token player is executed by a single task, which is called the coordinator. To characterize a centralized implementation, the algorithm to determine which transitions are enabled and can fire is important. Apart from the simple exhaustive test of all of the transitions, there are various solutions for reducing the costs of the enabling test, including static or dynamic representing places (Colom, Silva et al. 1986) and transition-driven techniques (Briz and Colom 1994).

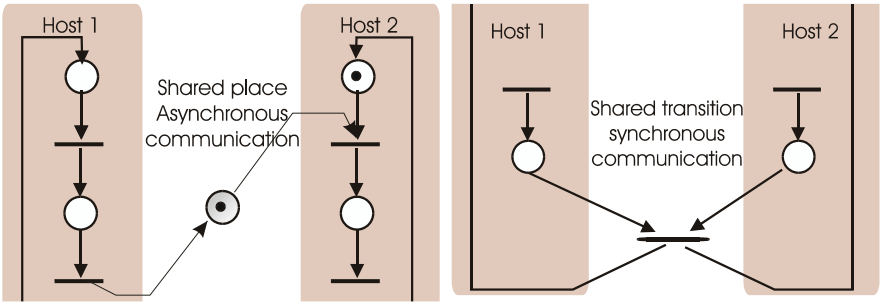



Fig. 1. Asynchronous and synchronous communication.

In a *decentralized* implementation, the net is decomposed into a set of sequential structures. Each of the structures is implemented by a task and the implementation is completed using the necessary communication and synchronization mechanisms.
In (Piedrafita and Villarroel 2006 a) we introduced centralized techniques into decentralized implementations, thereby creating a new technique called *concurrent coordinators*. The application can run several coordinators simultaneously by executing a sub-net for each subsystem. In this technique the net can be split following control or functional criteria, different from the behavioral ones needed for other decentralized techniques.
In this work we develop the distributed implementation of PN, extending the previous concurrent coordinators technique. We call this technique the *distributed coordinators*. In it, the execution of the coordinators takes place in several hosts, connected by a communication network. In each host one or more coordinators executes its Petri net. The coordinators read the inputs, executes its Petri net (the token player) and writes the outputs, i.e. achieving local machine control. For the necessary information interchange between hosts, the implementation is completed using the necessary communication mechanisms.

A Middleware that allows the implementation of distributed PN has been developed. For this purpose, a set of Java Classes to implement synchronous and asynchronous communication paradigms has been designed. These classes implement the *local and remote communications places*, the *auction communication place* and the *auction solver*. To implement the communication between the different coordinators, a protocol is needed that supports real-time traffic over Ethernet. In our case the protocol RT-WMP has been chosen, but the same implementation could have been made with other real-time protocols such as RT-RMI (Borg and Wellings 2003) or RT-EP (Martínez, Harbour et al. 2003).
The classes for managing communication with the Real-time RT-WMP protocol have been developed. This set of classes also implements the necessary real-time execution of the remote procedure calls. Also, the classes for communicating with input/output modules in industrial field buses CANopen, Interbus and Industrial Ethernet have been developed.
We have implemented the distributed implementation techniques of PN in the Java language using the Java Real-time extension (Bollella and Gosling 2000) and following some ideas presented in (Piedrafita and Villarroel 2006 a) and (Piedrafita and Villarroel 2006 b). .
The real-time extension provides Java with the necessary aspects for the programming of real-time systems, e.g., preemptive planning based on static priorities, asynchronous transfer of control, real-time high resolution clocks, and the possibility of execution over the Java garbage collector. In our implementations, we used the Real-time Java Virtual Machine JamaicaVM v2.7 (Aicas 2007). The target hardware was a personal computer with Pentium IV processor at 1.7 GHz, running Red Hat Linux 2.4.

## 3. Distributed implementation of Petri nets

In order to make a distributed implementation of PN, a net partition has to be made. In previous works (Colom, Silva et al. 1986), (García and Villarroel 1998), (Villarroel 1990), the authors show how to make a Petri net partition in sequential processes, applying structural and behavioral analysis. In this work the partition is done following a functional criterion. The PNs are subdivided into several subnets with local access to the input/output modules criteria and the machine's local control criteria.
This partition sets out a set of subnets, a set of shared places and a set of shared transitions. The subnets can be implemented in several computers. The implementation of the subnets is made in a centralized form. In each computer one or more coordinators are in charge of executing the corresponding subnets. The coordinators are threads with real-time requirements. From the perspective of the Java Real-time Specification, they are periodic RealtimeThreads with high priority that are scheduled without round-robin.
The shared places model the asynchronous communications; the shared transitions model the synchronous communication. The communication between distributed PNs can be made in asynchronous form by means of shared places, and in synchronous form by means of shared transitions (see Fig. 1).

### 3.1 Communication Places

The partition into two or more subnets leads to the appearance of shared places among subnets. These places are known as communication places (see Fig. 2). A communication place is implemented as a protected object with synchronized methods to mark and unmark the place. Program 1 shows the *communication place* code.

The communications places can be shared between Petri nets of the same host, or between Petri nets of different hosts. A *communication place* is implemented in one of the hosts and belongs to the PN executed in the machine in question.

```
public class Commmunicationplace extends state implements Serializable {
public Commmunicationplace (state e) {
   super(e);setCompartido(true);
}
synchronized void mark (int intokens) {
   tokens= tokens + intokens;
}
synchronized boolean unmark (int outtokens) {
   if (getTokens() >= outtokens){
   tokens= tokens - outtokens; return true;}
   Else return false;
}
```

Program 1. Communication place

The interface of *communication places* is also implemented in the rest of the nodes that share it, as *communication places remote interface*. The calls to the methods of *communication places remote interface* are sent to *communication places* through the real-time communication network.

It is possible to access this place as:

- *communication place*, when the access is local in the host machine. The local coordinator and its communications places are implemented in the same Java Virtual machine. The Local coordinator access to its *communication places* callings is methods directly.
- *communication place remote interface* (CPRI), when the access is from another machine of the network. The CPRI is the remote implementation of the corresponding *communication place*. The CPRI and the remote coordinator are implemented in a remote host in its Java Virtual Machine. The Remote coordinator access to its CPRI callings is methods directly. These callings are directed to the *communication places* through the real-time communication network over the protocol RT-WMP.

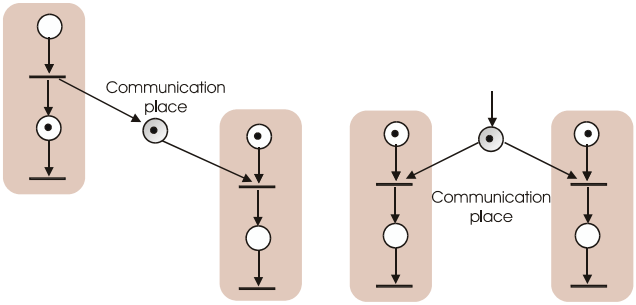



Fig. 2. Communication place: a) single destination b) multiple destination

### 3.2 Auction Communication Places

If, in the execution of its Petri net, a coordinator needs to unmark a *communication place,* it is necessary to consult the number of available tokens and then unmark the communication place. This fact can cause problems since operation is not atomic. In our decentralized implementation (Piedrafita and Villarroel 2006 a), several coordinators, all of them implemented as Real-time Threads with the same priority, run simultaneously in a single computer. The execution is made in a single processor and threads are scheduled following a static priorities policy without round-robin. A coordinator can consult the number of tokens in a place and next fire the transition and unmark the place without pre-emption by another coordinator.

However, if implementation is distributed, the simultaneous execution of several coordinators takes place in several computers, and it can happen that several coordinators want to unmark a *communication place* simultaneously. These external conflicts can occur when a *communication place* has several output transitions distributed in different subnets. In this case, the conflict has been solved by means of the introduction of an auction where the transitions are fired in priority order. These shared places are implemented by means of a protected object called *auction communication place*.

```
public auctionCommmunicationplace (state e..) {
   super(e);setCompartido(true);
}
synchronized boolean tobidup(tokens, priority) {
if (auctionopen == false) {
   auctionopen = true; requestlist.clear();
   counter = 0;
   requestfiring c =  Totallist.get(counter);
   c.setTokens(tokens);c.setPriority(priority);
   counter++;requestlist.add(c);
   try { wait(4);} catch (Exception e) {}
   auctionopen = false;
   selection();notifyAll();
   return requestlist.contains(c);
   }
   else {
   requestfiring c = Totallist.get(counter);
   c.setTokens(tokens);c.setPriority(priority);
   counter++;requestlist.add(c);
   try {wait();} catch (Exception e) {}
   return requestlist.contains(c)
    }  }
}
```

Program 2. Auction Communication Place

When a coordinator wants to unmark an *auction communication place,* it carries out a call to the method *tobidup* () of this one. The call is stored with its priority and the number of tokens requested. Next, if it is the first request, the caller (i.e. the first coordinator) is suspended for 4 ms. Coordinators are implemented as Periodic Real-time Threads with 25 ms of period to obtain good control of the machines. The conflict must be solved within the 25 ms of period of the coordinators. During this 4 ms, other local or remote coordinators can enter the auction to unmark that place. Their requests are stored and they are suspended by

calling a *wait*() method. After 4 ms, the first coordinator wakes up and closes the auction. Then it calls to the method *selection()* to assign tokens, and next with *notifyall*() wakes up the rest of the coordinators. If the assigned request list contains the request of the coordinator, the method *tobidup*() returns *true*. The execution of a *tobidup()* call consists of the execution of the test and the unmarking of the *auction communication place*. These two operations are made in atomic form.

In this way, the conflict is correctly solved, since all the transitions that want to unmark a place enter in the auction and the transitions are fired in priority order. With this method, remote coordinators can access the tokens. The maximum blocking time is 4 ms to avoid control problems, but also to allow the calls to *tobidup()* that travel through the real-time communication network to be taken into account.

In case of distributed simple conflicts where the communication places have several output transitions in the same PN, the implementation of the *communication place* must be made in an active object, in a thread. If several transitions that belong to the same PN wish to unmark the same *communication place*, the first call of *tobidup*() blocks the execution of the coordinator, thus the rest of transitions would not enter the auction. Therefore, to solve this problem, an active object that manages these situations is needed. We call it *auction solver*.

The *auction solver* receives requests from the output transitions, reserves tokens according to the priority and informs in a list which transitions can fire. When the suspend time is expired and the place is not unmarked by the transition, the *auction solver* cancels the reservation of the marks. The implementation of the *auction solver* is made up in a Real-time Thread of higher priority than the coordinator Thread.

### 3.3 Synchronous Communication

For the implementation of shared transitions in Java, it would be necessary to have some primitive of synchronous communication. Java does not have instructions that allow synchronous communication such as the ADA *rendez-vous*. Moreover, even if it were available, it could not be used since the rendez-vousconstruction blocks the caller thread. Therefore, the coordinator is blocked until the acceptance of the appointment and this involves the loss of control during that time. In order to implement the shared transitions in Java it is necessary to transform the PN according to Fig. 3. In this way, a synchronous communication using a shared transition is transformed into two asynchronous communications using communication places.

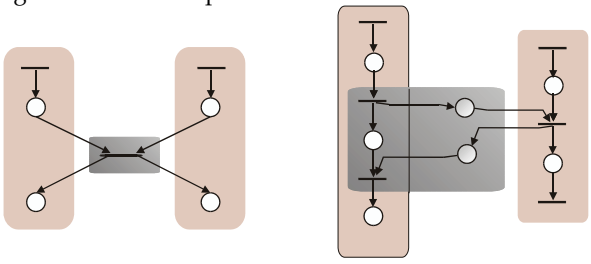
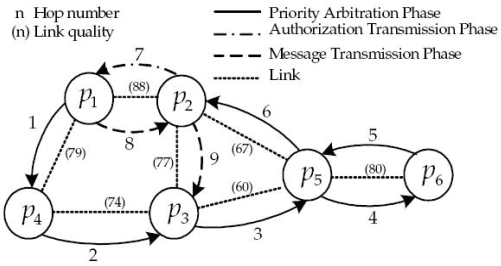


Fig. 3. Conversion of a synchronous communication into two asynchronous communications.

## 4. Real-Time Communication

In order to guarantee a bounded time response, real-time communication is necessary among the nodes of the PN, and the execution of the methods should be made with the appropriate real-time priority.
The remote access to a shared place is made by means of messages in protocol RT-WMP as explained later.
The Real-time Wireless Multi-hop Protocol (RT-WMP) is a protocol for MANETs. It was designed to work over the 802.11 protocol and supports real-time traffic. In fact, end-to-end message delay in RT-WMP has a bounded and known duration and the protocol also manages global static message priorities. Besides, RT-WMP supports multi-hop communications. The protocol has been designed to connect a relatively small group (10-20 units maximum) of mobile nodes. It is based on a token-passing scheme and is designed to manage rapid topology changes through the exchange of a matrix containing link quality among nodes. RT-WMP has an error recovery mechanism that can recover from certain types of errors without jeopardizing real-time behaviour and has a technique for reincorporating lost nodes.
Even if the RT-WMP has been designed to work in wireless multi-hop networks, it can work perfectly over wired Ethernet. However, the nodes belonging to the RT-WMP protocol must have a dedicated and isolated subnetwork to guarantee the real-time performances.




Link Quality Matrix

| | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ |
|---|---|---|---|---|---|---|
| $p_1$ | - | 88 | 0 | 79 | 0 | 0 |
| $p_2$ | 88 | - | 77 | 0 | 67 | 0 |
| $p_3$ | 0 | 77 | - | 74 | 60 | 0 |
| $p_4$ | 79 | 0 | 74 | - | 0 | 0 |
| $p_5$ | 0 | 67 | 60 | 0 | - | 80 |
| $p_6$ | 0 | 0 | 0 | 0 | 80 | - |

Fig. 4 A hypothetical situation described by the network graph and the corresponding LQM. The hops sequence of the protocol is also shown.

### 4.1 RT-WMP Overview

The protocol works in three phases (see Fig. 4): Priority Arbitration phase (PAP), Authorization Transmission Phase (ATP) and Message Transmission Phase (MTP). During the PAP, nodes reach a consensus over which of them holds the Most Priority Message (MPM) in the network at a given moment. Subsequently, in the ATP, an authorization to transmit is sent to the node which holds the highest priority message. Finally, in the MTP, this node sends the message to the destination node. To reach a consensus over which node holds the highest priority message, in the PAP a token travels through all of the nodes. The token holds information on the priority level of the MPM in the network and its owner among the set of nodes already reached by the token. The node which initiates the PAP states that the highest priority message in its own queue is the MPM in the whole network and stores this information in the token. Then it sends the token to another node, which checks the messages in its own queue. If the node verifies that it holds a message with a

higher priority level than the one carried by the token, it modifies the token data and continues the phase. The last node to receive the token, which knows the identity of the MPM holder, closes the PAP and initiates the ATP. In this phase, the node calculates a path to the MPM holder using the topology information shared among the members of the network (the Link Quality Matrix, see below) and sends an authorization message to the first node in the path. The latter will route the message to the second node in the path and so on, until the authorization reaches the MPM holder. This is when the MTP begins. The development of this phase is quite similar to the preceding one. The node that has received the authorization calculates the path to reach the destination, and sends the message to the first node of the path. The message follows the path and eventually reaches its destination. The phases repeat one after another i.e., when the MTP finishes, the node destination of the message initiates a new PAP and so on. When none of the nodes have a message to transmit, the authorization and message transmission phases are omitted and the priority arbitration phases repeat continuously.

### 4.2 The Link Quality Matrix

To describe the topology of the network, RT-WMP defines an extension of the network connectivity graph (as defined in (Facchinetti, Buttazzo, et al. (2005)) adding nonnegative values on the edges of the graph. These values are calculated as functions of the radio signal between pairs of nodes and are indicators of link quality between them. These values are represented in a matrix called the Link Quality Matrix (LQM), the elements of which describe link quality between nodes (see Fig. 4). Each column describes the links of a node with its neighbours. Nodes use this matrix to select which node to pass the token to and to take decisions on the best path to route a message from a source to a destination. All the nodes have a local copy of the LQM that is updated each time a frame is received.
Besides, every node is responsible for updating its column of the LQM (both in the local copy and the shared copy) to inform the other nodes about local topology changes.

### 4.3 Error Handling

RT-WMP is quite robust in case of node failure. The implicit acknowledgement technique used dispenses with the necessity of monitoring nodes to control the loss of the token. In RT-WMP, in fact, when a *pk* node sends a frame of any type, it listens to the channel for a timeout. The receiver *pl* node immediately processes the frame received and sends another frame to a third *pm* node. The first sender listens to such a frame as well and interprets it as an acknowledgement. If the first sender does not hear the frame within timeout, it supposes that the *pl* node has fallen or is out of its coverage range. In this case, the behaviour depends on the phase that the protocol is in. If it is in the ATP or MTP, *pk* discards the frame and starts a new PAP. However, if it is in the PAP, the *pk* node sends the token to another node to continue the PAP without jeopardizing the temporizations (see (Tardioli and Villarroel 2007) for details). Communication errors can produce another type of problem. Let us consider the situation where, in the PAP, the *pk* node sends a token to the pa node and waits for an implicit acknowledgment. Node pa processes the frame and sends the frame to node *pb*. As explained earlier, the last pass is also the acknowledgement for pk. However, if node *pb* hears the frame but *pk* does not, a token duplication occurs. In fact both nodes *pk* and *pb* continue the PAP and at that moment there are two tokens in the network. This problem

was solved introducing a serial field in the frames. In this way if a node receives frames with old serials, it discards them and informs the sender sending drop-frame information.

### 4.4 Use of RT-WMP

The RT-WMP is written in C language. However, we have developed an interface Java to C with Java Native interface technology (JNI). A dynamic library has been created and the procedures have been adapted to be used from Java using the JNI.

Fig. 8 shows the access from the coordinators executed in the virtual machines of PC1 and PC3 to a shared place implemented in the virtual machine of the PC2. The access is made locally to the *CPRI*. The calls to the methods of *CPRI* are transformed into messages in protocol RT-WMP. These messages when arriving at the PC2 involve the execution of methods of the *communication place*.

Fig. 5 shows the Unified Modelling Language (UML) Diagram of the classes written for RT-WMP communication. Two classes to represent the communication places have been implemented:

- *CommunicationplaceRtWmp:* for representation of the local instance of the object for which we want to invoke the methods.
- *CommunicationplaceRIRtWmp:* which is the remote interface of the *Communicationplace*.

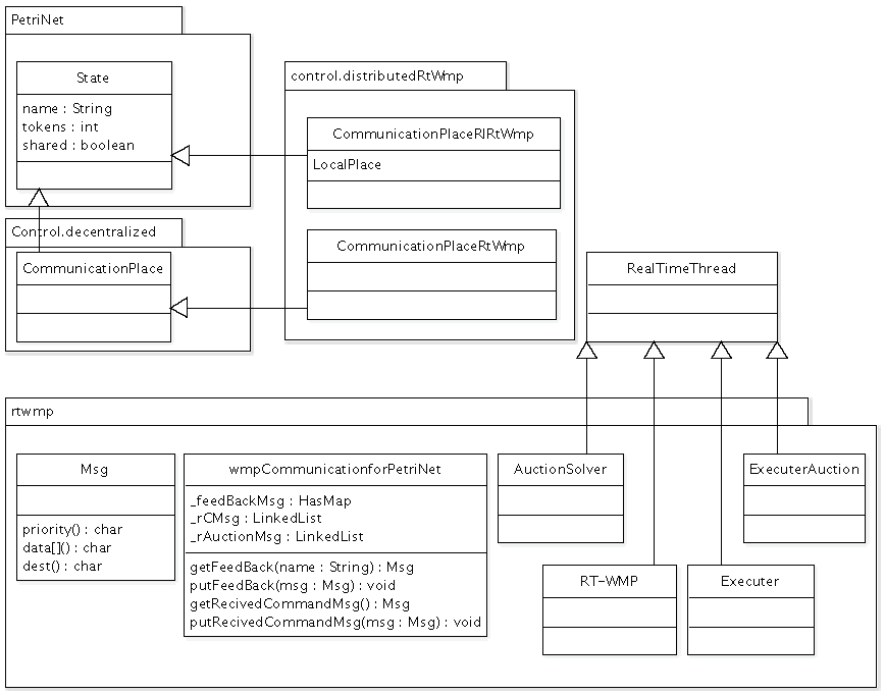



Fig. 5. RT-WMP communication Java classes.

The threads in charge of managing the communication and the execution of the remote calls are:

- *RT-WMP*. A Real-time Thread of high priority, it manages the communication in the real-time protocol.
- *Executer*. A Real-time Thread in charge of executing the remote requests.
- *ExecuterAuction*. Similar to *Executer* but specialized for the *tobidup()* remote method because that method needs more time for its execution.
- *Auction Solver*. A Real-time Thread in charge of solving conflicts.

It is necessary that the tasks in charge of managing the communication are executed at higher priority than the coordinators. In order to obtain this:

- Access to the synchronized methods of the communication places is made by means of the priority ceiling protocol.
- The execution of the Auction Solver is made with a higher priority than the coordinators.
- The execution of threads that deal with the communication has a priority higher than the coordinator threads.
- Threads in charge of executing the remote procedures are executed at a priority higher than the coordinators.

Thread *RT-WMP* is executed at the maximum real-time priority (in our application, 38). Threads *Executer, ExecuterAuction* and *Auction Solver* are executed with a lower priority than the *RT-WMP* threads but greater than the *Coordinators* (in our application, 35). With this communication architecture a bounded response time in communication is obtained. Fig. 6 shows the Execution priorities of the threads in the proposed architecture.

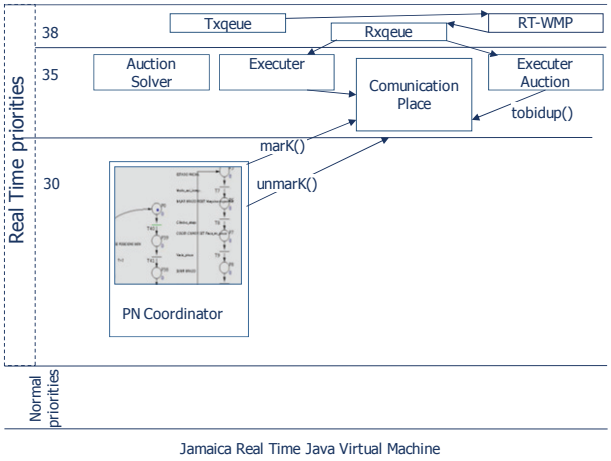



Fig. 6. Execution Priorities

## 5. Distributed Control of the Flexible Manufacturing Cell

The practical application of the technologies presented in this paper were tested using a flexible manufacturing cell installed at the Department of Computer Science and Systems Engineering at the University of Zaragoza, for research and teaching purposes. The

manufacturing cell is composed of a set of stations for the production and storage of pneumatic cylinders.
The initial control system of the cell consisted of a programmable logic controller (PLC) that controlled each station. A real-time industrial net connected all of the PLCs.
We have installed a new distributed control system based on several computers, connected by communications networks, and a fieldbus (see Fig. 7). One of the computers is equipped with a fieldbus master card, where the input/output modules of the machines are connected. The rest of the computers control the input/output hardware by Industrial Ethernet.
Stations 1 and 4 have Inline modules (Phoenix_Contact 2006) with protocol Interbus, the master of the bus being a Computer (PC 1) with the CIF50-IBM card of Hilscher (Hilscher 2007). In the rest of the cell, Industrial Ethernet is supported. In Stations 2, 3 and in the transport, the Advantys (Schneider_Electric. 2006) input/output modules communicate in the protocol Modbus TCP over Ethernet. For Ethernet communication two segmented subnetworks are arranged. The first one is used for communication with input/output modules and the second one for real-time communication among computers in RT-WMP protocol.
Each station of the cell has a read/write head of the pallet memory that is connected to a product identifier module (Pepperl&Fuchs IVI-KHD2-4HRX), (Pepperl&fuchs 2006). The product identifier module is a resource shared by all stations. Communication with the identifier module is achieved using a serial port. Access to the module must be protected from concurrent access.
In our implementations, we used the Real-time Java Virtual Machine JamaicaVM v2.7 (Aicas, 2007). The target hardware was four computers with Pentium IV processors at 1.7 GHz, running Red Hat Linux 2.4.

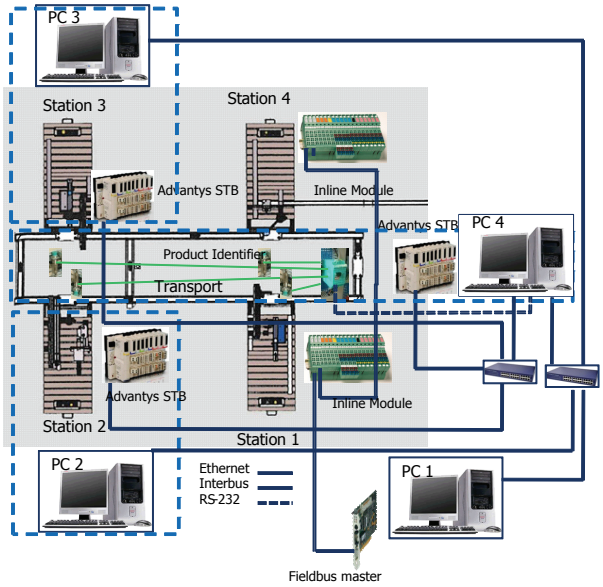



Fig. 7. Hardware Architecture

### 5.1 Software Control Architecture

The PN that models the desired behavior of the cell has been subdivided into several subnets by following local access to the input/output modules criterion, and the local control of the machines criterion. This subdivision leads to a set of subnets and to a set of shared places that model the communications. The subnets have been implemented in several computers. Each computer is in charge of control of a part of the Cell. In each computer one or more coordinators are in charge of executing the corresponding subnet (see Fig. 8).

The execution of the coordinator takes place in a high priority thread within the Jamaica Real-Time Java Virtual Machine. PNs communicate by means of shared places (i.e. the communication places). For example, the access to the product identifier is modeled by means of a shared place that is implemented as a local communication place in the PN executed by the Cell Coordinator and as a remote communication place in the rest of the PNs.

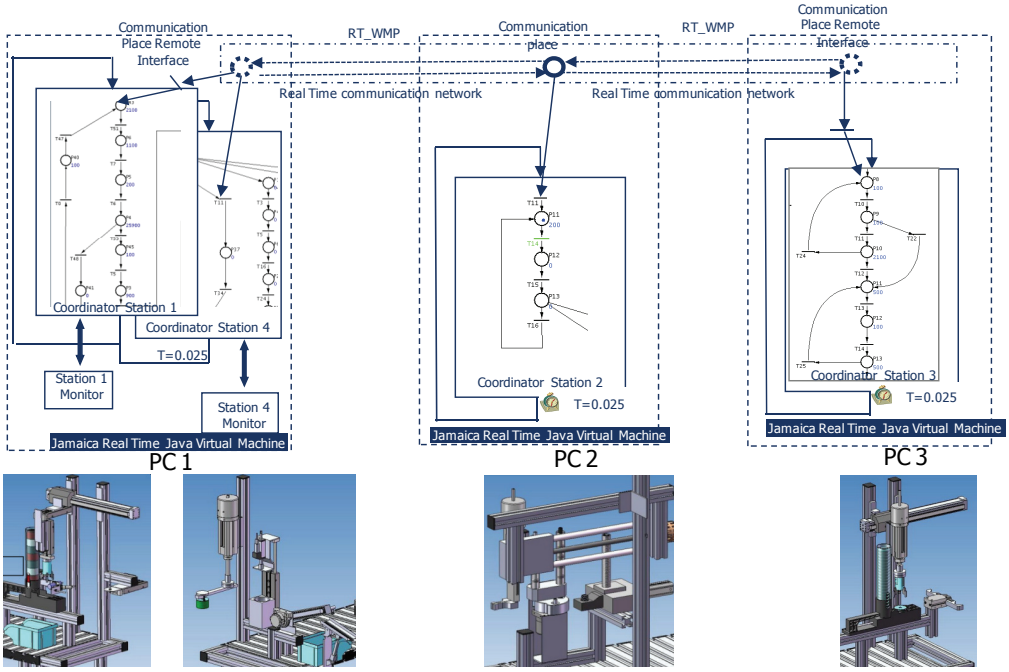



Fig. 8. Software Control Architecture

### 5.2 Communication Control Architecture

Fig. 9 shows the UML sequence diagram (Douglass 2004) of the tobidup() communication and remote execution. When in the execution of a coordinator it is desired to unmark a communication place, the method *tobidup*() of the *CommunicationplaceRIRtWmp* is called. (?)This creates a message and puts it in the transmission queue of the RT-WMP. This message is received by the RT-WMP of another node. The *Executer* reads the message,

searches the corresponding *CommunicationplaceRtWmp* and executes the request method. Finally it sends a feedback message to the *CommunicationplaceRIRtWmp*. The time necessary to unmark a communication place from a remote machine is between 10 and 12 ms.

### 5.3 Control deadlines

We have also tested the communication time among computers and the input/output modules in industrial Ethernet. This time is between 6 and 7 ms. The data transmission time in the Interbus is deterministic and can be known using the characteristics that the manufacturer provides. In this case the data transmission time is 1.6 ms.
With the communication times indicated, a control period of almost 25 ms is possible. This control period is the period of the coordinators (periodic Real-time Threads).

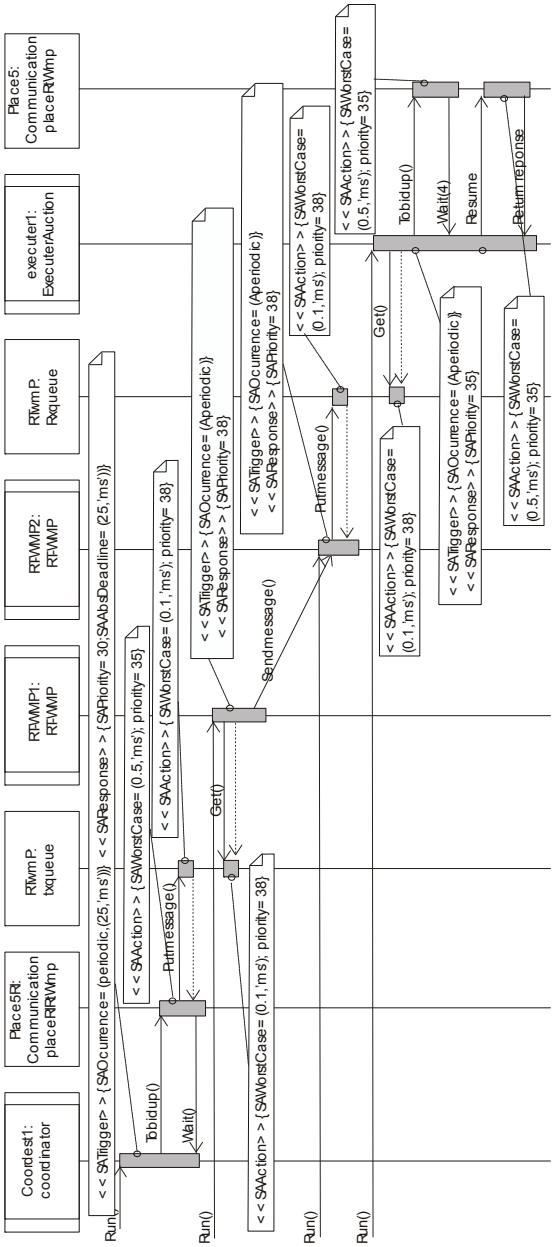



Fig. 9. Communication UML sequence diagram

The proposed concurrent structure and priorities guarantee that the controllers execute forever using updated input data and allow real-time analysis of the thread set. Following a rate monotonic approach, all of the local controller's threads run within their period (the control period) if:

$$C_{coord} + t_{bus} + t_{com} <= T \quad (1)$$

Where $T$ is the control period, $C_{coord}$ is the WCET of the local(s) coordinator(s), $t_{bus}$ is the WCET for reading the inputs and writing the outputs in the fieldbus, $t_{com}$ is the WCET for communicating with a remote communication place.
The WCET $t_{bus}$ in the fieldbus Ethernet is 7 ms. In the bus Interbus it is 1.6 ms. The WCET for communication is 12 ms. In the PC1(fieldbus Interbus):

$$C_{coordest1} + C_{coordest4} + t_{bus} + t_{com} = 1+1+1.6+12=15.6 <= 25 \text{ ms.}$$

In the PC2, PC3 and PC4 (fieldbus Ethernet):

$$C_{coordest2} + t_{bus} + t_{com} = 1.2+7+12=20.2 <= 25 \text{ ms. (PC2)}$$
$$C_{coordest3} + t_{bus} + t_{com} = 1.1+7+12=20.1 <= 25 \text{ ms. (PC3)}$$
$$C_{coordtransport} + t_{bus} + t_{com} = 0.9+7+12=19.9 <= 25 \text{ ms. (PC4)}$$

If the previous condition is fulfilled, the worst-case response time for events in the system can be calculated as:

$$t_r \leq 2T + t_{bus} \quad (2)$$

That is, the response time ($t_r$) has a bound related to the control period ($T$) and the reading-writing time of the bus ($t_{bus}$). An example of system response time to an incoming event is presented in Fig. 10
In our application, the control period is 25 ms, enough for the dynamics of the controlled system. With Ethernet, the maximum read-write time is 7 ms; therefore, the response time of the real-time control will be:

$$t_r \leq 2 * T + t_{bus} = 2 * 0.025 + 0.007 = 0.057 \text{ s}$$

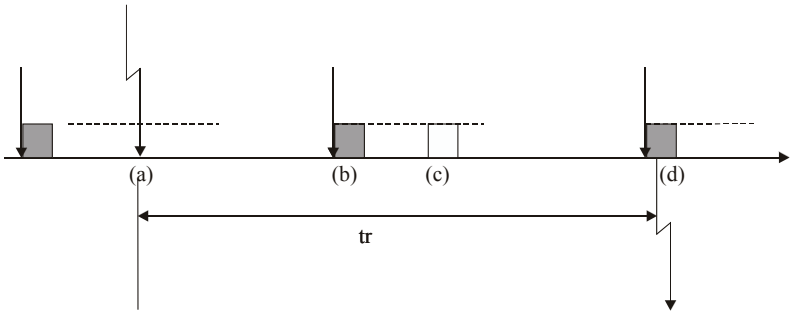



Fig. 10. System control time response:

a) An event happens in the system (e.g., a pallet arrives at a station).
b) The event is copied to the memory image of the control system.
c) A local controller reads the event and establishes the reaction as changes on the memory image of outputs.
d) The outputs are established in the system through the fieldbus.

## 6. Conclusions

In this paper a new PN implementation technique called *distributed coordinators* has been developed. This technique involves the use of centralized, decentralized and distributed implementations.

A platform that allows the implementation of distributed PNs has been developed. For this purpose, a set of Java Classes has been designed to implement synchronous and asynchronous communication paradigms. These classes implement the communications places and their remote interfaces: *communications places remote interface*, the *auction communication place* and the *Auction solver*. We propose a method to implement simple conflicts in distributed PN. In future works we intend to study multiple and coupled conflicts.

To grant the real-time execution of the remote calls to the methods of the remote communication places, a new software architecture has been implemented on top of the real-time protocol RT-WMP. In this way the system is analyzable from the real-time perspective.

All of the techniques and technologies presented in this work have been evaluated in a practical application: the distributed control of a flexible manufacturing cell. The control system of the cell is currently running without problems. This work, which is an extension of our previous research, provides the basis for further investigations into PN software implementations, languages, and real-time execution platforms.

## 7. References

Aicas, G. (2007). JamaicaVM Realtime Java Technology. http://www.aicas.com/.

Bollella, G. and J. Gosling ( 2000). "The Real-time Specification for Java." Computer 33(6): 47-54.

Borg, A. and A. Wellings (2003). A real-time RMI framework for the RTSJ. 15th Euromicro Conference on Real-time Systems (ECRTS'03).

Brams, G. V. (1982). Reseaux de Petri: Theorie et Practique, Vols. I and II, Masson.

Briz, J. L. and J. M. Colom (1994). "Implementation of Weighted Place/Transition Nets based on Linear Enabling Functions." Application and Theory of Petri nets 815: 99–118.

Bruno, G. and G. Marchetto (1986). "Process-translatable Petri nets for the rapid Prototyping of Process-Control Systems." Ieee Transactions on Software Engineering 12(2): 346-357.

Colom, J. M., M. Silva, et al. (1986). "On software implementation of Petri nets and colored Petri nets using high-level concurrent languages." Seventh European Workshop on applications and theory of Petri nets, Oxford, July 86: 207-241.

Douglass, B. P. (2004). Real-time UML: advances in the UML for real-time systems, Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA.

Facchinetti, T., G. Buttazzo, et al. (2005). Resource reservation and connectivity tracking to support real-time communication among mobile units. EURASIP J. Wirel. Commun. Netw., 5(5):712–730, 2005.

García, F. J. and J. L. Villarroel (1998). Decentralized Implementation of Real-Time Systems Using Time Petri nets. Application to Mobile Robot Control. 5th IFAC Workshop on Algorithms and Architectures for Real-time Control Cancun, Mexico.

Hilscher. (2007). "PC cards. http://www.hilscher.com."

Martínez, J. M., M. G. Harbour, et al. (2003). RT-EP: Real-Time Ethernet Protocol for Analyzable distributed Applications on a Minimum Real-Time POSIX Kernel.

Murata, T. (1989). "Petri nets- Properties, Analysys and applications." Proceedings of the Ieee 77(4): 541-580.

Pepperl&fuchs. (2006). "IVI-KHD2-4HRX DataSheet. http://www.pepperl-fuchs.com." 2006.

Phoenix_Contact. (2006). "Inline Modules. www.phoenixcontact.com. ."

Piedrafita, R. and J. L. Villarroel (2006 a). Implementation of time Petri nets in real-time Java. Proceedings of the 4th international workshop on Java technologies for real-time and embedded systems.

Piedrafita, R. and J. L. Villarroel (2006 b). Petri nets and Java. Real-Time Control of a flexible manufacturing cell. Emerging Technologies and Factory Automation, 2006. ETFA'06. IEEE Conference on.

Schneider_Electric. (2006). "Advantys STB. http://www.telemecanique.com."

Tardioli, D. and J. L. Villarroel (2007). Real-time communications over 802.11: RT-WMP.

Taubner, D. (1988). "On the implementation of Petri nets." Lecture Notes in Computer Science 340: 418-439.

Villarroel, J. L. (1990). Integración Informática del Control de Sistemas Flexibles de Fabricación. Tesis Doctoral. Ingeniería Eléctrica e Informática, Universidad de Zaragoza.

# Hybrid state Petri nets which have the analysis power of stochastic hybrid systems and the formal verification power of automata

Mariken H.C. Everdij and Henk A.P. Blom
*National Aerospace Laboratory NLR*
*The Netherlands*

## 1. Introduction

For a large range of complex applications, governments and industries invest in the development of innovative new systems existing of many distributed components that interact in a dynamic way with many uncertainties. Before any such system can be introduced into practice, an evaluation needs to have shown that both the system and the way it is used in its new context realizes the applicable objectives. If the new complex system is in its interactions similar to a previous system, such investigation can be done by analysis judgement of capable and experienced experts who judge local behaviour and implicitly assume that the interactions are working as before. If the complex system is very different from the old system, then this expert judgement approach falls short. A valuable alternative is to develop a mathematical model that incorporates the interactions, analyse this model, mobilise domain experts to evaluate where the model is representative for reality and where it needs improvement, and learn to understand how the real system works by learning how the model works. This requires a growing need for modelling and analysis of stochastic hybrid systems.
*Petri nets*, e.g. (David & Alla, 1994), have shown to be useful for developing models of various complex applications. Typical Petri net features are concurrency and synchronisation mechanism, hierarchical and modular construction, and natural expression of causal dependencies, in combination with graphical and equational representation. Numerous extensions to the basic formalism have been developed that combine different modelling features in an integrated way, including various hybrid state Petri net versions, e.g. (Giua, 1999), which combine discrete and continuous system aspects.
As a powerful class of models that support stochastic analysis, (Davis, 1984; 1993) introduced *piecewise deterministic Markov processes* (PDPs) as the most general class of continuous-time hybrid state Markov processes which include both discrete and continuous processes, except diffusion. In (Bujorianu & Lygeros, 2003; Hu et al., 2000) the PDPs have been defined as stochastic hybrid automata. Subsequently, diffusion by means of Brownian motion has been incorporated (Bujorianu & Lygeros, 2006). This way, a formal connection is established between stochastic hybrid processes that are supported by powerful stochastic analysis tools (Davis, 1993; Elliott, 1982; Elliott et al., 1995) and the automata formalism to develop formal verification tools (Frehse, 2008; Kwiatkowska et al., 2004; Labinaz et al., 1997).

In order to combine the advantages of the Petri net modelling formalisms and those of the Markovian analysis formalism, (Malhotra & Trivedi, 1994) and (Muppala et al., 2000) started the development of establishing formal connections between Petri nets and stochastic processes. Their result is a hierarchy of various dependability models based on their modelling power. At the left-hand-side of this power hierarchy are Petri net models, with *generalised stochastic Petri nets* (GSPN) at the bottom, and *deterministic and stochastic Petri nets* (DSPN) at the top. At the right-hand-side of this power hierarchy are *Markov chains* at the bottom and *semi-Markov processes* at the top. Arrows between different formalisms indicate that mappings exist, i.e. that the elements of one formalism can be represented in terms of the elements of the other formalism, such that the executions, i.e. their solutions as a stochastic process, are equivalent. In a series of studies (Everdij & Blom, 2003; 2005; 2006) developed an extension of this power hierarchy in probabilistic modelling, see Fig. 1.

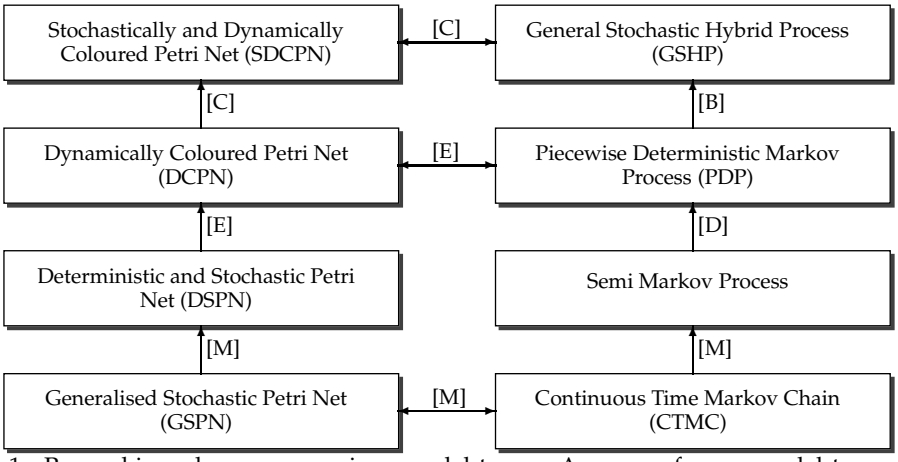



Fig. 1. Power hierarchy among various model types. An arrow from a model to another model indicates that the second model has more modelling power than the first model. The [M] arrows have been established in (Malhotra & Trivedi, 1994; Muppala et al., 2000). The [D] arrow is established in (Davis, 1984). The [B] arrow is established in (Bujorianu & Lygeros, 2006) and in (Blom, 2003). The [E] arrows are established in (Everdij & Blom, 2003; 2005). The [C] arrows are established in (Everdij & Blom, 2006) and the current chapter.

At the left hand side of this power hierarchy, we extended DSPN to *dynamically coloured Petri nets* (DCPN) and further to *stochastically and dynamically coloured Petri nets* (SDCPN). At the right hand side of the power hierarchy we extended semi Markov processes to *piecewise deterministic Markov processes* (PDP) and further to *general stochastic hybrid processes* (GSHP). In addition we showed effective ways how a DCPN can be mapped into PDP and the other way around, and how SDCPN can be mapped into GSHP and the other way around. DCPN and SDCPN are hybrid Petri net classes in which the tokens have Euclidean-valued colours that change through time (*dynamically*) while the tokens reside in their place. For DCPN, these colours follow ordinary differential equations, for SDCPN, the colours follow stochastic differential equations. The specific strength of (S)DCPN is their compositional specification power, which makes available a hierarchical modelling approach that separates local modelling issues from global modelling issues. This is illustrated for a large distributed example in air

traffic management (Everdij et al., 2006), which covers many distributed agents each of which interacts in a dynamic way with the others. Through a series of studies (Strubbe & Van der Schaft, 2005) developed a powerful compositional specification approach for automaton of PDP type (i.e. without Brownian motion), but for the complex air traffic management example (S)DCPN was shown to be better at compositional specification (Strubbe & Van der Schaft, 2004, Section 5.2).
For the mappings developed in (Everdij & Blom, 2006) between SDCPN and GSHP we made use of the *general stochastic hybrid system* (GSHS) theoretical setting developed by (Bujorianu & Lygeros, 2006), where GSHP is defined as an execution of a GSHS. More specifically, this means that SDCPN and GSHS are *bisimilar* in the sense that executions of SDCPN and GSHS yield GSHPs which are probabilistically equivalent, see e.g. (Bujorianu et al., 2005; Van der Schaft, 2004). Because of this bisimilarity, each formalism can take advantage of the strengths of both of them (Everdij & Blom, 2008).
Although the progress in the development of GSHP as an execution of a GSHS has led to significant increase of available stochastic analysis tools, there are some remaining issues to be addressed:

- *Jump linear systems* are not well covered, which unfortunately excludes most existing work on stochastic hybrid systems.
- *Semi-martingale property* of GSHS execution is unknown, which prohibits the use of Itô's differentiation rule for semi-martingales.

In the current chapter, these issues are further developed by considering GSHP not only as GSHS executions, but also as solutions of *hybrid stochastic differential equations* (HSDE). The HSDE approach towards studying GSHP has been developed in a series of complementary studies (Blom, 2003; Blom et al., 2003; Krystul, 2006; Krystul et al., 2007). The aim of this chapter is to characterise the relations between SDCPN, GSHP, HSDE and GSHS and to show that SDCPN, GSHS and HSDE are bisimilar. Fig. 2 shows the relations between the formalisms, and the key tools available for each of them.
With these relations, the properties and advantages of the various approaches come within reach of each other. The compositional specification power of SDCPN makes it relatively easy to develop a model for a complex system with multiple interactions. Subsequently, in the analysis stage three alternative approaches can be taken. The first is direct execution of SDCPN and evaluation through e.g. Monte Carlo simulation. The second is mapping the SDCPN into a GSHS and evaluating its execution, with the advantages of connection to formal methods in automata theory and to optimal control theory (Bujorianu & Lygeros, 2004). The third is mapping the SDCPN into HSDE and evaluating its solution, with the advantages of stochastic analysis for semi-martingales (Elliott, 1982; Elliott et al., 1995). With the GSHP resulting from any of these three means, properties become available such as convergence of discretisation, existence of limits, existence of event probabilities, strong Markov properties, and reachability analysis (Bujorianu & Lygeros, 2006; Davis, 1993; Ethier & Kurtz, 1986).
The organisation of this chapter is as follows. Section 2 defines SDCPN and the related SDCPN process. Section 3 defines GSHS and its GSHS execution. Section 4 defines HSDE and its stochastic process solution. Section 5 shows that SDCPN, GSHS and HSDE are bisimilar. Section 6 gives an example SDCPN. Section 7 presents this SDCPN example by an HSDE and by a GSHS. Section 8 gives conclusions. The appendices provide proof for the theorems in Section 5.

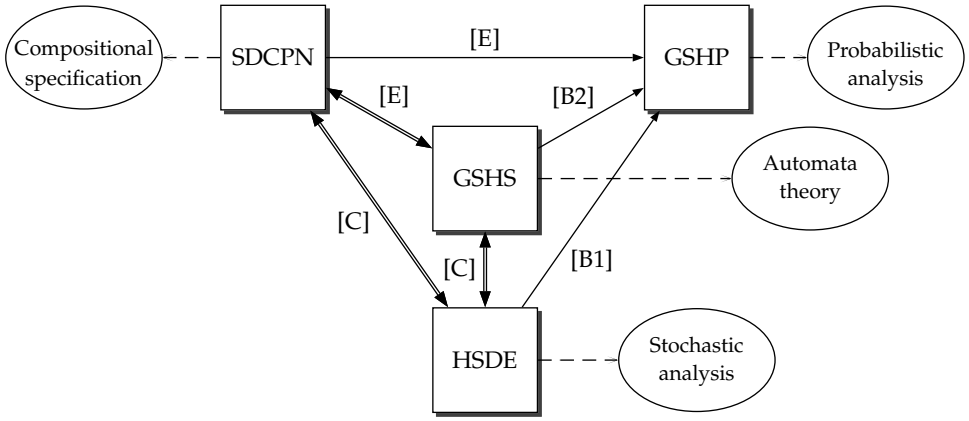



Fig. 2. Relationship between SDCPN, GSHS, GSHP and HSDE, and their key properties and advantages. The [B1] arrow is established in (Blom, 2003). The [B2] arrow is established in (Bujorianu & Lygeros, 2006). The [E] arrows are established in (Everdij & Blom, 2006). The [C] arrows are established in the current chapter, with bisimilarity relations having two-directional arrows.

## 2. SDCPN

This section presents a definition of *stochastically and dynamically coloured Petri net* (SDCPN).

**Definition 2.1** (Stochastically and dynamically coloured Petri net)**.** *An SDCPN is a collection of elements* $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{W}, \mathcal{G}, \mathcal{D}, \mathcal{F})$*, together with an SDCPN execution prescription which makes use of a sequence* $\{U_i; i = 0, 1, \ldots\}$ *of independent uniform* $U[0,1]$ *random variables, of independent sequences of mutually independent standard Brownian motions* $\{B_t^{i,P}; i = 1, 2, \ldots\}$ *of appropriate dimensions, one sequence for each place* $P$*, and of five rules R0–R4 that solve enabling conflicts.*

The formal SDCPN definition provided below is organised as follows: Section 2.1 defines the SDCPN elements $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{W}, \mathcal{G}, \mathcal{D}, \mathcal{F})$. Section 2.2 explains the SDCPN execution, which makes use of the rules R0–R4. Section 2.3 explains how the SDCPN execution defines a unique stochastic process.

### 2.1 SDCPN elements

The SDCPN elements $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{W}, \mathcal{G}, \mathcal{D}, \mathcal{F})$ are defined as follows:

- $\mathcal{P}$ is a finite set of places.
- $\mathcal{T}$ is a finite set of transitions which consists of 1) a set $\mathcal{T}_G$ of guard transitions, 2) a set $\mathcal{T}_D$ of delay transitions and 3) a set $\mathcal{T}_I$ of immediate transitions.
- $\mathcal{A}$ is a finite set of arcs which consists of 1) a set $\mathcal{A}_O$ of ordinary arcs, 2) a set $\mathcal{A}_E$ of enabling arcs and 3) a set $\mathcal{A}_I$ of inhibitor arcs.

- $\mathcal{N} : \mathcal{A} \rightarrow \mathcal{P} \times \mathcal{T} \cup \mathcal{T} \times \mathcal{P}$ is a node function which maps each arc $A \in \mathcal{A}$ to a pair of ordered nodes $\mathcal{N}(A)$, where a node is a place or a transition[1]. The place of $\mathcal{N}(A)$ is denoted by $P(A)$, the transition of $\mathcal{N}(A)$ is denoted by $T(A)$, such that for all $A \in \mathcal{A}_E \cup \mathcal{A}_I$: $\mathcal{N}(A) = (P(A), T(A))$ and for all $A \in \mathcal{A}_O$: either $\mathcal{N}(A) = (P(A), T(A))$ or $\mathcal{N}(A) = (T(A), P(A))$. Further notation:
  - $A(T) = \{A \in \mathcal{A} \mid T(A) = T\}$ denotes the set of arcs connected to transition $T$,
    $A_{in}(T) = \{A \in A(T) \mid \mathcal{N}(A) = (P(A), T)\}$ is the set of input arcs of $T$,
    $A_{out}(T) = \{A \in A(T) \mid \mathcal{N}(A) = (T, P(A))\}$ is the set of output arcs of $T$,
    $A_{in,O}(T) = A_{in}(T) \cap \mathcal{A}_O$ is the set of ordinary input arcs of $T$,
    $A_{in,OE}(T) = A_{in}(T) \cap \{\mathcal{A}_E \cup \mathcal{A}_O\}$ is the set of input arcs of $T$ that are either ordinary or enabling, and
  - $P(\mathcal{A}^{\subset}) = \{P(A); A \in \mathcal{A}^{\subset}\}$ is the multi-set of places connected to the subset of arcs $\mathcal{A}^{\subset} \subset \mathcal{A}$.

  Finally, $\{A_i \in \mathcal{A}_I \mid \exists A \in \mathcal{A}, A \neq A_i : \mathcal{N}(A) = \mathcal{N}(A_i)\} = \emptyset$, i.e., if an inhibitor arc points from a place $P$ to a transition $T$, there is no other arc from $P$ to $T$.
- $\mathcal{S} \subset \{\mathbb{R}^0, \mathbb{R}^1, \mathbb{R}^2, \ldots\}$ is a finite set of colour types, with $\mathbb{R}^0 \triangleq \emptyset$.
- $\mathcal{C} : \mathcal{P} \rightarrow \mathcal{S}$ is a colour type function which maps each place $P \in \mathcal{P}$ to a specific colour type in $\mathcal{S}$. Each token in $P$ is to have a colour in $\mathcal{C}(P)$. Since $\mathcal{C}(P) \in \{\mathbb{R}^0, \mathbb{R}^1, \ldots\}$, there exists a function $n : \mathcal{P} \rightarrow \mathbb{N}$ such that $\mathcal{C}(P) = \mathbb{R}^{n(P)}$. If $\mathcal{C}(P) = \mathbb{R}^0 \triangleq \emptyset$ then a token in $P$ has no colour. Further notation: if $P(\mathcal{A}^{\subset})$ contains more than one place, e.g., $P(\mathcal{A}^{\subset}) = \{P_{i_1}, \ldots, P_{i_k}\}$, then $\mathcal{C}(P(\mathcal{A}^{\subset}))$ is defined by $\mathcal{C}(P_{i_1}) \times \cdots \times \mathcal{C}(P_{i_k})$.
- $\mathcal{I} : \mathbb{N}^{|\mathcal{P}|} \times \mathcal{C}(\mathcal{P})^{\mathbb{N}} \rightarrow [0, 1]$ is a probability measure, which defines the initial marking of the net: for each place it defines a number $\geq 0$ of tokens initially in it and it defines their initial colours. Here, $\mathbb{N}^{|\mathcal{P}|} \triangleq \{(m_1, \ldots, m_{|\mathcal{P}|}); m_i \in \mathbb{N}, m_i < \infty, i = 1, \ldots, |\mathcal{P}|\}$ and $\mathcal{C}(\mathcal{P})^{\mathbb{N}} \triangleq \{\mathcal{C}(P_1)^{m_1} \times \cdots \times \mathcal{C}(P_{|\mathcal{P}|})^{m_{|\mathcal{P}|}}; m_i \in \mathbb{N}, m_i < \infty, i = 1, \ldots, |\mathcal{P}|\}$, where $\mathcal{C}(P_i)^{m_i} \triangleq \mathbb{R}^{m_i n(P_i)}$ for all $i = 1, \ldots, |\mathcal{P}|$, where $\mathcal{P}$ is denoted $\mathcal{P} = \{P_1, \ldots P_{|\mathcal{P}|}\}$. It is assumed that all tokens in a place are distinguishable by a unique identification tag which translates to a unique ordering/listing of tokens per place.
- $\mathcal{V} = \{\mathcal{V}_P; P \in \mathcal{P}, \mathcal{C}(P) \neq \mathbb{R}^0\}$ is a set of token colour functions. For each place $P \in \mathcal{P}$ for which $\mathcal{C}(P) \neq \mathbb{R}^0$, it contains a function $\mathcal{V}_P : \mathcal{C}(P) \rightarrow \mathcal{C}(P)$ that defines the drift coefficient of a differential equation for the colour of a token in place $P$.
- $\mathcal{W} = \{\mathcal{W}_P; P \in \mathcal{P}, \mathcal{C}(P) \neq \mathbb{R}^0\}$ is a set of token colour matrix functions. For each place $P \in \mathcal{P}$ for which $\mathcal{C}(P) \neq \mathbb{R}^0$, it contains a measurable mapping $\mathcal{W}_P : \mathcal{C}(P) \rightarrow \mathbb{R}^{n(P) \times h(P)}$ that defines the diffusion coefficient of a stochastic differential equation for the colour of a token in place $P$, where $h : \mathcal{P} \rightarrow \mathbb{N}$. It is assumed that $\mathcal{W}_P$ and $\mathcal{V}_P$ satisfy conditions that ensure a probabilistically unique solution of each stochastic differential equation.[2]

[1] The SDCPN arcs have no arc weights, but this node function definition leaves the freedom to define multiple arcs between the same pair of transition and place or place and transition (except if an inhibitor arc is involved).

[2] In the earlier definition by (Everdij & Blom, 2006) it was assumed that $\mathcal{V}_P$ and $\mathcal{W}_P$ satisfy local Lipschitz condition. This condition has now been relaxed to probabilistic uniqueness of solution of the related stochastic differential equation(s).

- $\mathcal{G} = \{\mathcal{G}_T; T \in \mathcal{T}_G\}$ is a set of transition guards. For each $T \in \mathcal{T}_G$, it contains a transition guard $\mathcal{G}_T$, which is an open subset in $\mathcal{C}(P(A_{in,OE}(T)))$ with boundary $\partial\mathcal{G}_T$. If $\mathcal{C}(P(A_{in,OE}(T))) = \mathbb{R}^0$ then $\partial\mathcal{G}_T = \emptyset$.[3] There is no requirement that $\mathcal{G}_T$ be connected.
- $\mathcal{D} = \{\mathcal{D}_T; T \in \mathcal{T}_D\}$ is a set of transition delay rates. For each $T \in \mathcal{T}_D$, it contains a locally integrable transition delay rate $\mathcal{D}_T : \mathcal{C}(P(A_{in,OE}(T))) \to \mathbb{R}^+$. If $\mathcal{C}(P(A_{in,OE}(T))) = \mathbb{R}^0$ then $\mathcal{D}_T$ is a constant function. [4]
- $\mathcal{F} = \{\mathcal{F}_T; T \in \mathcal{T}\}$ is a set of firing measures. For each $T \in \mathcal{T}$, it contains a firing measure $\mathcal{F}_T : (\{0,1\}^{|A_{out}(T)|} \times \mathcal{C}(P(A_{out}(T)))) \times \mathcal{C}(P(A_{in,OE}(T))) \to [0,1]$, which generates the number and colours of the tokens produced when transition $T$ fires, given the value of the vector $\in \mathcal{C}(P(A_{in,OE}(T)))$ that collects all input tokens: For each output arc ($\in A_{out}(T)$), zero or one token is produced, and if the colours of the tokens produced are collected in a vector, this vector is $\in \mathcal{C}(P(A_{out}(T)))$. For each fixed $H \subset \mathcal{C}(P(A_{out}(T)))$, $\mathcal{F}_T(H;\cdot)$ is measurable. For any $c \in \mathcal{C}(P(A_{in,OE}(T)))$, $\mathcal{F}_T(\cdot;c)$ is a probability measure. Here, $\{0,1\}^{|A_{out}(T)|} \triangleq \{(e_1,\ldots,e_{|A_{out}(T)|}); e_i \in \{0,1\}, i = 1,\ldots,|A_{out}(T)|\}$.

For the places, transitions and arcs, the graphical notation is as in Figure 3.

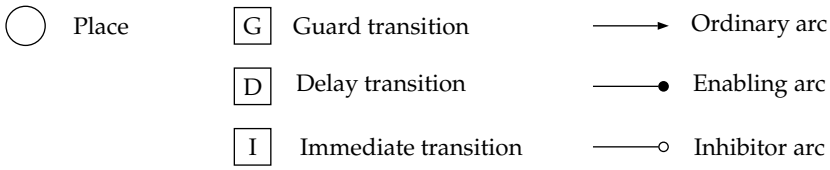



Fig. 3. Graphical notation for places, transitions and arcs in an SDCPN

### 2.2 SDCPN execution

The execution of an SDCPN provides a series of increasing stopping times, $0 = \tau_0 < \tau_1 < \tau_2 < \cdots$, with for $t \in (\tau_k, \tau_{k+1})$ a fixed number of tokens per place and per token a colour which is the solution of a stochastic differential equation. It uses a sequence $\{U_i; i = 0,1,\ldots\}$ of independent uniform $U[0,1]$ random variables, and independent sequences of mutually independent standard Brownian motions $\{B_t^{i,P}; i = 1,2,\ldots\}$ of appropriate dimensions, one sequence for each place $P$.

#### Initiation

The probability measure $\mathcal{I}$ characterises an initial marking at $\tau_0$, i.e. it gives each place $P \in \mathcal{P}$ zero or more tokens and gives each token in $P$ a colour in $\mathcal{C}(P)$, i.e. a Euclidean-valued vector. Define the inverse of $\mathcal{I}$ by a measurable function $\mathcal{I}^{inv} : [0,1] \to \mathbb{N}^{|\mathcal{P}|} \times \mathcal{C}(\mathcal{P})^{\mathbb{N}}$ such that $\mu_L\{u \mid \mathcal{I}^{inv}(u) \in H\} = \mathcal{I}(H)$, for $H$ Borel measurable and $\mu_L$ the Lebesgue measure. Then the initial marking is a hybrid random vector characterised by $(M_0, C_0) = \mathcal{I}^{inv}(U_0)$. Here, $M_0$ is a $|\mathcal{P}|$-dimensional vector of non-negative integers, the $i$th component $M_{i,0}$ of which denotes

[3] In earlier SDCPN definitions, the transition guard was defined as a Boolean function that evaluated to True if the boundary of an open subset was hit by the input token colours. Without losing generality, the transition guard is now defined to be the open subset itself.

[4] In earlier SDCPN definitions, the transition delay was defined as a probability distribution function that made use of an integrable transition delay rate. Without losing generality, the transition delay is now defined to be the delay rate itself.

the number of tokens initially in place $P_i$, $i = 1, \ldots |\mathcal{P}|$, and $C_0$ is a $\sum_{i=1}^{|\mathcal{P}|} M_{i,0} n(P_i)$-dimensional Euclidean-valued random vector which provides the colours of the initial tokens. If $M_{1,0} \geq 1$ then the first $n(P_1)$ components of $C_0$ are assigned to the first token in $P_1$. If $M_{1,0} \geq 2$ then the next $n(P_1)$ components of $C_0$ are assigned to the second token in $P_1$, etc., until all tokens in $P_1$ have their assigned colour. The following components of $C_0$ are assigned to tokens in places $P_2, \ldots, P_{|\mathcal{P}|}$ in the same way. If $\mathcal{C}(P) = \mathbb{R}^0$ then the tokens in $P$ get no colour.

#### Token colour evolution

For each token in each place $P$ for which $\mathcal{C}(P) \neq \mathbb{R}^0$: if the colour of this token is equal to $C_0^P$ at time $t = \tau_0$, and if this token is still in this place at time $t > \tau_0$, then the colour $C_t^P$ of this token equals the probabilistically unique solution of the stochastic differential equation $dC_t^P = \mathcal{V}_P(C_t^P)dt + \mathcal{W}_P(C_t^P)dB_t^{i,P}$ with initial condition $C_{\tau_0}^P = C_0^P$, and with $\{B_t^{i,P}\}$ an $h(P)$-dimensional standard Brownian motion. The first token, if any, in place $P$ uses Brownian motion $\{B_t^{1,P}\}$; the second token, if any, uses $\{B_t^{2,P}\}$, etc. Each token in a place for which $\mathcal{C}(P) = \mathbb{R}^0$ remains without colour.

#### Transition enabling

A transition $T$ is *pre-enabled* if it has at least one token per incoming ordinary and enabling arc in each of its input places and has no token in places to which it is connected by an inhibitor arc. For each transition $T$ that is pre-enabled at $\tau_0$, consider one token per ordinary and enabling arc in its input places and write $C_t^T \in \mathcal{C}(P(A_{in,OE}(T)))$, $t \geq \tau_0$, as the column vector containing the colours of these tokens; $C_t^T$ evolves through time according to its corresponding token colour functions of the places in $P(A_{in,OE}(T))$. If this vector is not unique (i.e., if one input place contains several tokens per arc), all possible such vectors are executed in parallel. Hence, a transition can be pre-enabled by multiple combinations of input tokens in parallel.
A transition $T$ is *enabled* if it is pre-enabled and a second requirement holds true. For $T \in \mathcal{T}_I$, the second requirement automatically holds true at the time of pre-enabling. For $T \in \mathcal{T}_G$, the second requirement holds true when $C_t^T \in \partial\mathcal{G}_T$. For $T \in \mathcal{T}_D$, the second requirement holds true at $t = \tau_0 + \sigma_1^T$, where $\sigma_1^T$ is generated from a probability distribution function $D_T(t - \tau_0) = 1 - \exp(-\int_{\tau_0}^t \mathcal{D}_T(C_s^T)ds)$, i.e. $\sigma_1^T = D_T^{inv}(U)$, where $D_T^{inv}$ is the inverse of $D_T(t - \tau_0)$ defined by $D_T^{inv}(u) = \inf\{t - \tau_0 \mid \exp(-\int_{\tau_0}^t \mathcal{D}_T(C_s^T)ds) \leq u\}$, with $\inf\{\ \} = +\infty$. Each delay transition uses one new uniform random variable $U \sim \mathcal{U}[0,1]$ (per vector of input tokens) each time it becomes pre-enabled to determine its time of enabling.
In the case of competing enablings, the following rules apply:

- R0 The firing of an immediate transition has priority over the firing of a guard or a delay transition.
- R1 If one transition becomes enabled by two or more sets of input tokens at exactly the same time, and the firing of any one set will not disable one or more other sets, then it will fire these sets of tokens independently, at the same time.
- R2 If one transition becomes enabled by two or more sets of input tokens at exactly the same time, and the firing of any one set disables one or more other sets, then the set that is fired is selected randomly, with the same probability for each set.
- R3 If two or more transitions become enabled at exactly the same time and the firing of any one transition will not disable the other transitions, then they will fire at the same time.

R4 If two or more transitions become enabled at exactly the same time and the firing of any one transition disables some other transitions, then each combination of transitions that can fire independently without leaving enabled transitions gets the same probability of firing.

By these rules and their combinations, if a transition is enabled in a particular set of tokens, then it is either fired or it is disabled (in this set of tokens) by the firing of another transition.

#### Transition firing

If $T$ is enabled, suppose this occurs at time $\tau_1$ and in a particular vector of token colours $C^T_{\tau_1}$, it removes one token per arc in $A_{in,O}(T)$ corresponding with $C^T_{\tau_1}$ from each of its input places (i.e. tokens are not removed along enabling arcs). Next, $T$ produces zero or one token along each output arc: If $(e^T_{\tau_1}, a^T_{\tau_1})$ is a random hybrid vector generated from probability measure $\mathcal{F}_T(\cdot; C^T_{\tau_1})$, then vector $e^T_{\tau_1} \in \{0,1\}^{|A_{out}(T)|}$ is an $|A_{out}(T)|$-dimensional vector of zeros and ones, where the $i$th vector element corresponds with the $i$th outgoing arc of transition $T$. An output place gets a token iff it is connected to an arc that corresponds with a vector element 1. Moreover, $a^T_{\tau_1} \in \mathcal{C}(P(A_{out}(T)))$ specifies the colours of the produced tokens, i.e. if the first 1 in $e^T_{\tau_1}$ corresponds with an arc from $T$ to $P_j$, then the first $n(P_j)$ elements in vector $a^T_{\tau_1}$ are assigned to the token produced in output place $P_j$. The remaining elements in $a^T_{\tau_1}$ are assigned to other tokens in the same way. The random hybrid vector from $\mathcal{F}_T(\cdot; C^T_{\tau_1})$ is characterised by defining the inverse of $\mathcal{F}_T(\cdot; C^T_{\tau_1})$ as a measurable function $\mathcal{F}^{inv}_T : [0,1] \times \mathcal{C}(P(A_{in,OE}(T))) \to \{0,1\}^{|A_{out}(T)|} \times \mathcal{C}(P(A_{out}(T)))$ such that $\mu_L\{u \mid \mathcal{F}^{inv}_T(u,c) \in H\} = \mathcal{F}_T(H;c)$ for $H$ in the Borel set of $\{0,1\}^{|A_{out}(T)|} \times \mathcal{C}(P(A_{out}(T)))$ and $\mu_L$ is the Lebesgue measure. Then $(e^T_{\tau_1}, a^T_{\tau_1}) = \mathcal{F}^{inv}_T(U, C^T_{\tau_1})$. Each firing transition uses one new uniform random variable $U \sim U[0,1]$ per firing to determine its output tokens.

#### Execution from first transition firing onwards

At $t = \tau_1$, zero or more transitions are pre-enabled (if this number is zero, no transitions will fire anymore). If these include immediate transitions, then these are fired without delay, but with use of rules R0–R4. If after this, still immediate transitions are enabled, then these are also fired, and so forth, until no more immediate transitions are enabled. Each of the immediate transitions that fire uses their firing measure and one uniform random variable (per firing) to determine the number and colours of their output tokens. Next, the SDCPN is executed in the same way as described above for the situation from $\tau_0$ onwards.

In order to keep track of the identity of individual tokens, the tokens in a place are ordered according to the time at which they entered the place, or, if several tokens are produced for one place at the same time, according to the order within the set of arcs $\mathcal{A} = \{A_1, \ldots, A_{|\mathcal{A}|}\}$ along which these tokens were produced (the firing measure produces zero or one token along each output arc). If due to rule R1, a transition fires two or more tokens along one arc at the same time, their assigned order is according to the colours they have (smallest colour first). If under these conditions, two tokens have exactly the same colour, they are indistinguishable and the marking will not be dependent on their order.

### 2.3 SDCPN stochastic process

The marking of the SDCPN is given by the numbers of tokens in the places and the associated colour values of these tokens. Due to the uniquely defined order of the tokens, the marking is unique except possibly when one or more transitions fire (particularly, immediate transitions

fire without delay hence a sequence of immediate transitions firing will generate a sequence of markings at the same time instant). The SDCPN marking at each time instant can be mapped to a probabilistically unique SDCPN stochastic process $\{M_t, C_t\}$ as follows: For any $t \geq \tau_0$, let a token distribution be characterised by the vector $M'_t = (M'_{1,t}, \ldots, M'_{|\mathcal{P}|,t})$, where $M'_{i,t} \in \mathbb{N}$ denotes the number of tokens in place $P_i$ at time $t$ and $1, \ldots, |\mathcal{P}|$ refers to a unique ordering of places adopted for SDCPN. At times $t \in (\tau_{k-1}, \tau_k)$ when no transition fires, the token distribution is unique and the SDCPN discrete process state $M_t$ is defined to be equal to $M'_t$. The associated colours of these tokens are gathered in a column vector $C_t$ which first contains all colours of tokens in place $P_1$, next (i.e. below it) all colours of tokens in place $P_2$, etc, until place $P_{|\mathcal{P}|}$, where $1, \ldots, |\mathcal{P}|$ refers to a unique ordering of places adopted for SDCPN. Within a place the colours of the tokens are ordered according to the unique ordering of tokens within their place defined for SDCPN (see under SDCPN execution above).
If at time $t = \tau_k$ one or more transitions fire, then the set of applicable token distributions is collected in $\widetilde{M}_{\tau_k} = \{M'_{\tau_k} \mid M'_{\tau_k}$ is a token distribution at time $\tau_k\}$, and the SDCPN discrete process state at time $\tau_k$ is defined by $M_{\tau_k} = \{M'_{\tau_k} \mid M'_{\tau_k} \in \widetilde{M}_{\tau_k}$ and no transitions are enabled in $M'_{\tau_k}\}$. In other words, $M_{\tau_k}$ is defined to be the token distribution that occurs after all transitions that fire at time $\tau_k$ have been fired. The associated colours of these tokens are gathered in a column vector $C_{\tau_k}$ in the same way as described above. This construction ensures that the process $\{M_t, C_t\}$ has limits from the left and is continuous from the right, i.e., it satisfies the càdlàg property. If at a time $t$ when one or more transitions fire, the process $\{M_t\}$ jumps to the same value again, and only $C_t$ makes a jump, then the càdlàg property for $\{C_t\}$ (hence for $\{M_t, C_t\}$) is still maintained due to the timing construction of $\{M_t\}$ above and the direct coupling of $\{C_t\}$ with $\{M_t\}$.

## 3. GSHS

This section presents, following (Bujorianu & Lygeros, 2006), a definition of *general stochastic hybrid system* (GSHS) and its execution.

**Definition 3.1** (General stochastic hybrid system)**.** *A GSHS is an automaton (*$\mathbf{K}$, $d$, $\mathcal{X}$, $f$, $g$, Init, $\lambda$, $Q$*), where*

- $\mathbf{K}$ *is a countable set.*
- $d : \mathbf{K} \to \mathbb{N}$ *maps each* $\theta \in \mathbf{K}$ *to a natural number.*
- $\mathcal{X} : \mathbf{K} \to \{E_\theta; \theta \in \mathbf{K}\}$ *maps each* $\theta \in \mathbf{K}$ *to an open subset* $E_\theta$ *of* $\mathbb{R}^{d(\theta)}$*. With this, the hybrid state space is given by* $E \triangleq \{\{\theta\} \times E_\theta; \theta \in \mathbf{K}\}$.
- $f : E \to \{\mathbb{R}^{d(\theta)}; \theta \in \mathbf{K}\}$ *is a vector field.*
- $g : E \to \{\mathbb{R}^{d(\theta)\times h}; \theta \in \mathbf{K}\}$ *is a matrix field, with* $h \in \mathbb{N}$.
- Init: $\mathcal{B}(E) \to [0, 1]$ *is an initial probability measure, with* $\mathcal{B}(E)$ *the Borel* $\sigma$*-algebra on* $E$.
- $\lambda : E \to \mathbb{R}^+$ *is a jump rate function.*
- $Q : \mathcal{B}(E) \times (E \cup \partial E) \to [0, 1]$ *is a GSHS transition measure, where* $\partial E \triangleq \{\{\theta\} \times \partial E_\theta; \theta \in \mathbf{K}\}$ *is the boundary of* $E$*, in which* $\partial E_\theta$ *is the boundary of* $E_\theta$.

**Definition 3.2** (GSHS execution)**.** *A stochastic process* $\{\theta_t, X_t\}$ *is called a* GSHS execution *if there exists a sequence of stopping times* $0 = \tau_0 < \tau_1 < \tau_2 \cdots$ *such that for each* $k \in \mathbb{N}$*:*

- $(\theta_0, X_0)$ *is an* $E$*-valued random variable extracted according to probability measure* Init.

- *For $t \in [\tau_k, \tau_{k+1})$, $\theta_t = \theta_{\tau_k}$ and $X_t = X_t^k$, where for $t \geq \tau_k$, $X_t^k$ is a solution of the stochastic differential equation $dX_t^k = f(\theta_{\tau_k}, X_t^k)dt + g(\theta_{\tau_k}, X_t^k)dB_t^{\theta_{\tau_k}}$ with initial condition $X_{\tau_k}^k = X_{\tau_k}$, and where $\{B_t^\theta\}$ is $h$-dimensional standard Brownian motion for each $\theta \in \mathbf{K}$.*
- *$\tau_{k+1} = \tau_k + \sigma_k$, where $\sigma_k$ is chosen according to a survivor function given by $F(t) = \mathbf{1}_{(t<\tau^*)} \exp(-\int_0^t \lambda(\theta, X_s^k)ds)$. Here, $\tau^* = \inf\{t > \tau_k \mid X_t^k \in \partial E_{\theta_{\tau_k}}\}$ and $\mathbf{1}$ is indicator function.*
- *The probability distribution of $(\theta_{\tau_{k+1}}, X_{\tau_{k+1}})$, i.e. the hybrid state right after the jump, is governed by the law $Q(\cdot; (\theta_{\tau_k}, X_{\tau_{k+1}-}))$.*

(Bujorianu & Lygeros, 2006) show that under assumptions G1-G4 below, a GSHS execution is a strong Markov Process and has the càdlàg property (right continuous with left hand limits).

G1 $f(\theta, \cdot)$ and $g(\theta, \cdot)$ are Lipschitz continuous and bounded. This yields that for each initial state $(\theta, x)$ at initial time $\tau$ there exists a pathwise unique solution $X_t$ to $dX_t = f(\theta, X_t)dt + g(\theta, X_t)dB_t$, where $\{B_t\}$ is $h$-dimensional standard Brownian motion.

G2 $\lambda : E \to \mathbb{R}^+$ is a measurable function such that for all $\xi \in E$, there is $\epsilon(\xi) > 0$ such that $t \to \lambda(\theta_t, X_t)$ is integrable on $[0, \epsilon(\xi))$.

G3 For each fixed $A \in \mathcal{B}(E)$, the map $\xi \to Q(A; \xi)$ is measurable and for any $(\theta, x) \in E \cup \partial E$, $Q(\cdot; \theta, x)$ is a probability measure.

G4 If $N_t = \sum_k \mathbf{1}_{(t \geq \tau_k)}$, then it is assumed that for every starting point $(\theta, x)$ and for all $t \in \mathbb{R}^+$, $\mathbb{E}N_t < \infty$. This means, there will be a finite number of jumps in finite time.

## 4. HSDE

This section presents, following (Blom, 2003) and (Blom et al., 2003), a definition of *hybrid stochastic differential equation* (HSDE) and gives conditions under which the HSDE has a pathwise unique solution. This pathwise unique solution is referred to as *HSDE solution process* or GSHP. The basic advantage of using HSDE in defining a GSHP over using GSHS is that with the HSDE approach the spontaneous jump mechanism is explicitly built on an underlying stochastic basis, whereas in GSHS the execution itself imposes an underlying stochastic basis. The differences are further discussed in Section 4.3.
For the HSDE setting we start with a complete stochastic basis $(\Omega, \Im, \mathbb{F}, \mathbb{P}, \mathbb{T})$, in which a complete probability space $(\Omega, \Im, \mathbb{P})$ is equipped with a right-continuous filtration $\mathbb{F} = \{\Im_t\}$ on the positive time line $\mathbb{T} = \mathbb{R}^+$. This stochastic basis is endowed with a probability measure $\mu_{\theta_0, X_0}$ for the initial state, an independent $h$-dimensional standard Wiener process $\{W_t\}$ and an independent homogeneous Poisson random measure $p_P(dt, dz)$ on $\mathbb{T} \times \mathbb{R}^{d+1}$.

**Definition 4.1** (Hybrid stochastic differential equation)**.** *An HSDE on stochastic basis $(\Omega, \Im, \mathbb{F}, \mathbb{P}, \mathbb{T})$, is defined as a set of equations (1)-(8) in which a collection of elements ($\mathbb{M}$, $E$, $f$, $g$, $\mu_{\theta_0, X_0}$, $\Lambda$, $\psi$, $\rho$, $\mu$, $p_P$, $\{W_t\}$) appear.*

This section is organised as follows: Section 4.1 explains the elements and the equations (1)-(8) that define HSDE. Section 4.2 shows that under a number of HSDE conditions H1-H8, the HSDE has a pathwise unique solution which is a semi-martingale. Section 4.3 discusses the differences between GSHP as solution of HSDE and GSHP as execution of GSHS.

### 4.1 HSDE elements and equations

This section presents the elements and equations that define a HSDE on a hybrid state space. The elements $(\mathbb{M}, E, f, g, \mu_{\theta_0,X_0}, \Lambda, \psi, \rho, \mu, p_P, \{W_t\})$ are defined as follows:

- $\mathbb{M} = \{\vartheta_1, \ldots, \vartheta_N\}$ is a finite set, $N \in \mathbb{N}, 1 \leq N < \infty$.
- $E = \{\{\theta\} \times E_\theta; \theta \in \mathbb{M}\}$ is the hybrid state space, where for each $\theta \in \mathbb{M}$, $E_\theta$ is an open subset of $\mathbb{R}^n$ with boundary $\partial E_\theta$. The boundary of $E$ is $\partial E = \{\{\theta\} \times \partial E_\theta; \theta \in \mathbb{M}\}$.
- $f : \mathbb{M} \times \mathbb{R}^n \to \mathbb{R}^n$ is a measurable mapping.
- $g : \mathbb{M} \times \mathbb{R}^n \to \mathbb{R}^{n\times h}$ is a measurable mapping.
- $\mu_{\theta_0,X_0} : \Omega \times \mathcal{B}(E) \to [0,1]$ is a probability measure for the initial random variables $\theta_0$, $X_0$, which are defined on the stochastic basis; $\mu_{\theta_0,X_0}$ is assumed to be invertible.
- $\Lambda : \mathbb{M} \times \mathbb{R}^n \to [0,\infty)$ is a measurable mapping.
- $\psi : \mathbb{M} \times \mathbb{M} \times \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$ is a measurable mapping such that $x + \psi(\vartheta, \theta, x, \underline{z}) \in E_\vartheta$ for all $x \in E_\theta$, $\underline{z} \in \mathbb{R}^d$, and $\vartheta, \theta \in \mathbb{M}$.
- $\rho : \mathbb{M} \times \mathbb{M} \times \mathbb{R}^n \to [0,\infty)$ is a measurable mapping such that $\sum_{i=1}^N \rho(\vartheta_i, \theta, x) = 1$ for all $\theta \in \mathbb{M}, x \in \mathbb{R}^n$.
- $\mu : \Omega \times \mathbb{R}^d \to [0,1]$ is a probability measure which is assumed to be invertible.
- $p_P : \Omega \times \mathbb{T} \times \mathbb{R}^{d+1} \to \{0,1\}$ is a homogeneous Poisson random measure on the stochastic basis, independent of $(\theta_0, X_0)$. The intensity measure of $p_P(dt, dz)$ equals $dt \cdot \mu_L(dz_1) \cdot \mu(d\underline{z})$, where $z = \mathrm{Col}\{z_1, \underline{z}\}$ and $\mu_L$ is the Lebesgue measure.
- $W : \Omega \times \mathbb{T} \to \mathbb{R}^h$ such that $\{W_t\}$ is an $h$-dimensional standard Wiener process on the stochastic basis, and independent of $(\theta_0, X_0)$ and $p_P$.

Using these elements, the HSDE process $\{\theta_t^*, X_t^*\}$ is defined as follows:

$$\theta_t^* = \theta_t^k \text{ for all } t \in [\tau_k^b, \tau_{k+1}^b), k = 0, 1, 2, \ldots \tag{1}$$

$$X_t^* = X_t^k \text{ for all } t \in [\tau_k^b, \tau_{k+1}^b), k = 0, 1, 2, \ldots \tag{2}$$

Hence $\{\theta_t^*, X_t^*\}$ consists of a concatenation of processes $\{\theta_t^k, X_t^k\}$ which are defined by (3)-(8) below. If the system (1)-(8) has a solution in probabilistic sense, then the process $\{\theta_t^*, X_t^*\}$ is referred to as *HSDE solution process* or *GSHP*.

$$d\theta_t^k = \sum_{i=1}^N (\vartheta_i - \theta_{t-}^k) p_P(dt, (\Sigma_{i-1}(\theta_{t-}^k, X_{t-}^k), \Sigma_i(\theta_{t-}^k, X_{t-}^k)] \times \mathbb{R}^d) \tag{3}$$

$$dX_t^k = f(\theta_t^k, X_t^k)dt + g(\theta_t^k, X_t^k)dW_t + \int_{\mathbb{R}^d} \psi(\theta_t^k, \theta_{t-}^k, X_{t-}^k, \underline{z}) p_P(dt, (0, \Lambda(\theta_{t-}^k, X_{t-}^k)] \times d\underline{z}) \tag{4}$$

with $\theta_0^0 = \theta_0$, $X_0^0 = X_0$ and with $\Sigma_0$ through $\Sigma_N$ measurable mappings satisfying, for $\theta \in \mathbb{M}$, $\vartheta_j \in \mathbb{M}$, $x \in \mathbb{R}^n$:

$$\Sigma_i(\theta, x) = \begin{cases} \Lambda(\theta, x) \sum_{j=1}^i \rho(\vartheta_j, \theta, x) & \text{if } i > 0 \\ 0 & \text{if } i = 0 \end{cases} \tag{5}$$

In addition, for $k = 0, 1, 2, \ldots$, with $\tau_0^b = 0$:

$$\tau_{k+1}^b \triangleq \inf\{t > \tau_k^b \mid (\theta_t^k, X_t^k) \in \partial E\} \tag{6}$$

$$\mathbb{P}\{\theta_{\tau_{k+1}^b}^{k+1} = \vartheta, X_{\tau_{k+1}^b}^{k+1} \in A \mid \theta_{\tau_{k+1}^b-}^k = \theta, X_{\tau_{k+1}^b-}^k = x\} = Q(\{\vartheta\} \times A; \theta, x) \tag{7}$$

for $A \in \mathcal{B}(\mathbb{R}^n)$, where $Q$ is given by

$$Q(\{\vartheta\} \times A; \theta, x) = \rho(\vartheta, \theta, x) \int_{\mathbb{R}^d} \mathbf{1}_A(x + \psi(\vartheta, \theta, x, \underline{z})) \mu(d\underline{z}) \tag{8}$$

### 4.2 HSDE solution

This subsection shows that under a set of sufficient conditions H1-H8, the HSDE (1)-(8) has a pathwise unique solution. Note that the existence of a pathwise unique solution guarantees the existence of a unique solution in probabilistic sense.

**Proposition 4.1.** *Let conditions H1-H8 below hold true. Let* $(\theta_0^*(\omega), X_0^*(\omega)) = (\theta_0, X_0) \in E$ *for all* $\omega$. *Then for every initial condition* $(\theta_0, X_0)$, *(1)-(8) has a pathwise unique solution* $\{\theta_t^*, X_t^*\}$ *which is càdlàg and adapted and is a semi-martingale assuming values in the hybrid state space E.*

**H1** For all $\theta \in \mathbb{M}$ there exists a constant $K(\theta)$ such that for all $x \in \mathbb{R}^n$, $|f(\theta,x)|^2 + \|g(\theta,x))\|^2 \leq K(\theta)(1+|x|^2)$, where $|a|^2 = \sum_i (a_i)^2$ and $||b||^2 = \sum_{i,j}(b_{ij})^2$.

**H2** For all $r \in \mathbb{N}$ and for all $\theta \in \mathbb{M}$ there exists a constant $L_r(\theta)$ such that for all $x$ and $y$ in the ball $B_r = \{z \in \mathbb{R}^n \mid |z| \leq r+1\}$, $|f(\theta,x) - f(\theta,y)|^2 + \|g(\theta,x) - g(\theta,y)\|^2 \leq L_r(\theta)|x-y|^2$.

**H3** For each $\theta \in \mathbb{M}$, the mapping $\Lambda(\theta,\cdot) : \mathbb{R}^n \to [0,\infty)$ is continuous and bounded, with upper bound a constant $C_\Lambda$.

**H4** For all $(\theta,\vartheta) \in \mathbb{M}^2$, the mapping $\rho(\vartheta,\theta,\cdot) : \mathbb{R}^n \to [0,\infty)$ is continuous.

**H5** For all $r \in \mathbb{N}$ there exists a constant $M_r(\theta)$ such that

$$\sup_{|x|\leq r} \int_{\mathbb{R}^d} |\psi(\vartheta,\theta,x,\underline{z})| \mu(d\underline{z}) \leq M_r(\theta), \text{ for all } \vartheta,\theta \in \mathbb{M}$$

**H6** $|\psi(\theta,\theta,x,\underline{z})| = 0$ or $> 1$ for all $\theta \in \mathbb{M}$, $x \in \mathbb{R}^n$, $\underline{z} \in \mathbb{R}^d$

**H7** $\{(\theta_t^*, X_t^*)\}$ hits the boundary $\partial E$ a finite number of times on any finite time interval

**H8** $|\vartheta_i - \vartheta_j| > 1$ for $i \neq j$, with $|\cdot|$ a suitable metric well defined on $\mathbb{M}$.

(Blom, 2003) has used (Lepeltier & Marchal, 1976) to prove a version of Proposition 4.1 where $E = \mathbb{M} \times \mathbb{R}^n$, i.e. there are no boundaries with instantaneous jumps. Subsequently, (Blom et al., 2003) have proven the proposition under H1-H8 and the additional condition that $\{\tau_k^b\}$ is a sequence of predictable stopping times. (Krystul, 2006; Krystul & Blom, 2005) have shown that this additional condition can be removed. An overview of various HSDE versions is given in (Krystul et al., 2007).

### 4.3 Discussion of HSDE versus GSHS

HSDE and GSHS have a lot of similarities. Both concatenate different solutions of SDEs with hybrid jumps at each moment of switching to another SDE. Hence the differences are of a rather technical nature. This section collects these technical differences between GSHS and its GSHP execution, versus HSDE and its GSHP solution:

1. For GSHS, the discrete state space is a countable space of discrete variables. For HSDE, the discrete state space is a finite set.
2. For GSHS, the continuous state is Euclidean with a dimension dependent on $\theta$. For HSDE, the continuous state is Euclidean with constant dimension $n$.
3. The times of spontaneous jump of the GSHS execution are driven by a survivor function which imposes a stochastic basis. For HSDE, the times of spontaneous jumps are driven by a Poisson random measure endowed upon a given stochastic basis.
4. For GSHS, the size of jump is driven by a transition measure $Q$. For HSDE, the jump size is determined by probability measure $\mu$ and measurable mappings $\psi$ and $\rho$.

5. GSHS involves $|\mathbf{K}|$ Brownian motions. HSDE involves one Wiener process only.
6. For GSHS, the drift and diffusion coefficient are assumed (globally) Lipschitz and bounded. For HSDE, the drift and dissusion coefficient are locally Lipschitz and are allowed to grow with the continuous state.

For 1) and 2), GSHS has as advantage of being more general than HSDE. HSDE however has significant advantages regarding issues 3)-6): Regarding 3)-5), HSDE has the advantage that this allows to establish the *semi-martingale property*. Regarding 6), HSDE removes the particular restriction of GSHS which excludes *jump linear systems*.

## 5. SDCPN, GSHS and HSDE are bisimilar

This section shows that for each SDCPN there exists a GSHS which is bisimular, and there exists a HSDE which is bisimular. This is shown in the four theorems below.

**Theorem 5.1.** *Consider an arbitrary GSHS* ($\mathbf{K}$, $d$, $\mathcal{X}$, $f$, $g$, Init, $\lambda$, $Q$) *with a finite domain* $\mathbf{K}$. *If for each* $\theta$ *and initial value* $X_0$, *the stochastic differential equation* $dX_t = f(\theta, X_t)dt + g(\theta, X_t)dB_t$ *has a unique solution in probabilistic sense, then this GSHS can be mapped into an SDCPN* ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) *satisfying R0-R4. If the resulting SDCPN is executed on a probability space endowed with standard Brownian motion (one for each place), then the resulting SDCPN process and the GSHS execution are probabilistically equivalent.*

*Proof.* See (Everdij & Blom, 2006). □

**Theorem 5.2.** *Consider an arbitrary SDCPN* ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) *satisfying R0-R4. If in the initial marking no immediate transition is enabled, and if the number of tokens remains finite for* $t \to \infty$, *then this SDCPN can be mapped into a GSHS* ($\mathbf{K}$, $d$, $\mathcal{X}$, $f$, $g$, Init, $\lambda$, $Q$). *If the original SDCPN is executed on a probability space endowed with Brownian motion (one for each place) then the resulting GSHS execution and the SDCPN process are probabilistically equivalent.*

*Proof.* See (Everdij & Blom, 2006). □

**Theorem 5.3** (HSDE into SDCPN)**.** *Consider an arbitrary HSDE (1)-(8) with elements* ($\mathbb{M}$, $E$, $f$, $g$, $\mu_{\theta_0,X_0}$, $\Lambda$, $\psi$, $\rho$, $\mu$, $p_P$, $\{W_t\}$). *If for each* $\theta$ *the stochastic differential equation* $dX_t = f(\theta, X_t)dt + g(\theta, X_t)dW_t$ *has a unique solution in probabilistic sense and if* $\Lambda$ *is bounded, then the elements of this HSDE can be mapped into an SDCPN* ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) *satisfying R0–R4. If the resulting SDCPN is executed on a probability space endowed with sequences of standard Brownian motions (one sequence for each place), then the resulting SDCPN process and the HSDE solution process are probabilistically equivalent.*

*Proof.* See Appendix A. □

**Theorem 5.4** (SDCPN into HSDE)**.** *Consider an arbitrary SDCPN* ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) *satisfying R0–R4. If in the initial marking no immediate transition is enabled, if the delay rates* $\mathcal{D}_T$ *are bounded, and if the number of tokens remains finite for* $t \to \infty$, *then this SDCPN can be mapped into a HSDE (1)-(8) with elements* ($\mathbb{M}$, $E$, $f$, $g$, $\mu_{\theta_0,X_0}$, $\Lambda$, $\psi$, $\rho$, $\mu$, $p_P$, $\{W_t\}$). *If the original SDCPN is executed on a probability space which is endowed with sequences of standard Brownian motions (one sequence for each place), then the resulting HSDE solution process and the SDCPN process are probabilistically equivalent.*

*Proof.* See Appendix B. □

Theorems 5.1 and 5.2 imply that SDCPN and GSHS are bisimilar. Theorems 5.3 and 5.4 imply that SDCPN and HSDE are bisimilar. The implications are that GSHS and HSDE are also bisimilar and that the strengths of all three formalisms come within reach of each other. The use of this bisimilarity is illustrated by an example in the following two sections.

## 6. SDCPN example

To illustrate the advantages of SDCPN when modelling a complex system, consider a simplified model of the evolution of an aircraft in one sector of airspace. The deviation of this aircraft from its intended path is affected by its engine system and its navigation system. Each of these aircraft systems can be in either *Working* (functioning properly) or *Not working* (operating in some failure mode). Both systems switch between these modes independently and with exponentially distributed sojourn times, with finite rates $\delta_3$ (engine repaired), $\delta_4$ (engine fails), $\delta_5$ (navigation repaired) and $\delta_6$ (navigation fails), respectively. If both systems are *Working*, the aircraft evolves in *Nominal* mode and the position $Y_t$ and velocity $S_t$ of the aircraft are determined by $dX_t = \mathcal{V}_1(X_t)dt + \mathcal{W}_1 dW_t$, where $X_t = (Y_t, S_t)'$. If either one, or both, of the systems is *Not working*, the aircraft evolves in *Non-nominal* mode and the position and velocity of the aircraft are determined by $dX_t = \mathcal{V}_2(X_t)dt + \mathcal{W}_2 dW_t$. The factors $\mathcal{W}_1$ and $\mathcal{W}_2$ are determined by wind fluctuations. Initially, the aircraft has position $Y_0$ and velocity $S_0$, while both its systems are *Working*. The evaluation of this process may be stopped when the aircraft has *Landed*, i.e. its vertical position and velocity are equal to zero.

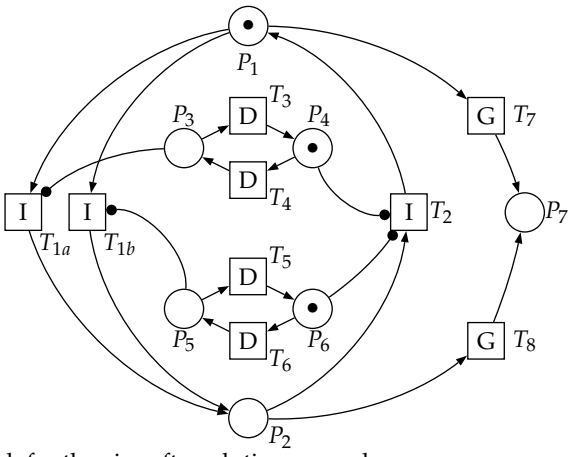



Fig. 4. SDCPN graph for the aircraft evolution example

Fig. 4 shows the SDCPN graph for this example, where,

- $P_1$ denotes aircraft evolution *Nominal*, i.e. evolution is according to $\mathcal{V}_1$ and $\mathcal{W}_1$.
- $P_2$ denotes aircraft evolution *Non-nominal*, i.e. evolution is according to $\mathcal{V}_2$ and $\mathcal{W}_2$.
- $P_3$ and $P_4$ denote engine system *Not working* and *Working*, respectively.
- $P_5$ and $P_6$ denote navigation system *Not working* and *Working*, respectively.
- $P_7$ denotes the aircraft has landed.
- $T_{1a}$ and $T_{1b}$ denote a transition of aircraft evolution from *Nominal* to *Non-nominal*, due to engine system or navigation system *Not working*, respectively.

- $T_2$ denotes a transition of aircraft evolution from *Non-nominal* to *Nominal*, due to engine system and navigation system both *Working* again.
- $T_3$ through $T_6$ denote transitions between *Working* and *Not working* of the engine and navigation systems.
- $T_7$ and $T_8$ denote transitions of the aircraft landing.

The graph in Fig. 4 completely defines SDCPN elements $\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$ and $\mathcal{N}$, where $\mathcal{T}_G = \{T_7, T_8\}$, $\mathcal{T}_D = \{T_3, T_4, T_5, T_6\}$ and $\mathcal{T}_I = \{T_{1a}, T_{1b}, T_2\}$. The other SDCPN elements are specified below:

$\mathcal{S}$: Two colour types are defined; $\mathcal{S} = \{\mathbb{R}^0, \mathbb{R}^6\}$.

$\mathcal{C}$: $\mathcal{C}(P_1) = \mathcal{C}(P_2) = \mathcal{C}(P_7) = \mathbb{R}^6$, i.e. tokens in $P_1$, $P_2$ and $P_7$ have colours in $\mathbb{R}^6$; the colour components model the 3-dimensional position and 3-dimensional velocity of the aircraft. $\mathcal{C}(P_3) = \mathcal{C}(P_4) = \mathcal{C}(P_5) = \mathcal{C}(P_6) = \mathbb{R}^0 \triangleq \emptyset$.

$\mathcal{I}$: Place $P_1$ initially has a token with colour $X_0 = (Y_0, S_0)'$, with $Y_0 \in \mathbb{R}^2 \times (0, \infty)$ and $S_0 \in \mathbb{R}^3 \setminus \mathrm{Col}\{0, 0, 0\}$. Places $P_4$ and $P_6$ initially each have a token without colour.

$\mathcal{V}$, $\mathcal{W}$: The token colour functions for places $P_1$, $P_2$ and $P_7$ are determined by $(\mathcal{V}_1, \mathcal{W}_1)$, $(\mathcal{V}_2, \mathcal{W}_2)$, and $(\mathcal{V}_7, \mathcal{W}_7)$, respectively, where $(\mathcal{V}_7, \mathcal{W}_7) = (0, 0)$. For places $P_3$ – $P_6$ there is no token colour function.

$\mathcal{G}$: Transitions $T_7$ and $T_8$ have a guard defined by $\mathcal{G}_{T_7} = \mathcal{G}_{T_8} = \mathbb{R}^2 \times (0, \infty) \times \mathbb{R}^2 \times (0, \infty)$.

$\mathcal{D}$: The jump rates for transitions $T_3$, $T_4$, $T_5$ and $T_6$ are $\mathcal{D}_{T_3}(\cdot) = \delta_3$, $\mathcal{D}_{T_4}(\cdot) = \delta_4$, $\mathcal{D}_{T_5}(\cdot) = \delta_5$ and $\mathcal{D}_{T_6}(\cdot) = \delta_6$.

$\mathcal{F}$: Each transition has a unique output place, to which it fires a token with a colour (if applicable) equal to the colour of the token removed.

## 7. Mapping of SDCPN example to HSDE and GSHS

Next we transform the SDCPN of Section 6 into an HSDE. The first step is to construct the state space $\mathbb{M}$ for the HSDE discrete process $\{\theta_t\}$. This is done by identifying the SDCPN *reachability graph*. Nodes in the reachability graph provide the number of tokens in each of the SDCPN places. Arrows connect these nodes as they represent transitions firing. The SDCPN of Fig. 4 has seven places hence the reachability graph for this example has elements that are vectors of length 7. These nodes, excluding the nodes that enable immediate transitions, form the HSDE discrete state space.

The reachability graph is shown in Fig. 5, with nodes that form the HSDE discrete state space in Bold typeface, i.e. $\mathbb{M} = \{V_1, \ldots, V_8\}$, with $V_1 = (1,0,0,1,0,1,0)$, $V_2 = (0,1,1,0,0,1,0)$, $V_3 = (0,1,1,0,1,0,0)$, $V_4 = (0,1,0,1,1,0,0)$, $V_5 = (0,0,0,1,0,1,1)$, $V_6 = (0,0,1,0,0,1,1)$, $V_7 = (0,0,1,0,1,0,1)$, $V_8 = (0,0,0,1,1,0,1)$. Since initially there is a token in places $P_1$, $P_4$ and $P_6$, the HSDE initial mode equals $\theta_0 = V_1 = (1,0,0,1,0,1,0)$. The HSDE initial continuous state value equals the vector containing the initial colours of all initial tokens. Since the initial colour of the token in Place $P_1$ equals $X_0$, and the tokens in places $P_4$ and $P_6$ have no colour, the HSDE initial continuous state value equals $\mathrm{Col}\{X_0, \emptyset, \emptyset\} = X_0$. The HSDE drift coefficient $f$ is given by $f(\theta, \cdot) = \mathcal{V}_1(\cdot)$ for $\theta = V_1$, $f(\theta, \cdot) = \mathcal{V}_2(\cdot)$ for $\theta \in \{V_2, V_3, V_4\}$, and $f(\theta, \cdot) = 0$ otherwise. For the diffusion coefficient, $g(\theta, \cdot) = \mathcal{W}_1$ for $\theta = V_1$, $g(\theta, \cdot) = \mathcal{W}_2$ for $\theta \in \{V_2, V_3, V_4\}$, and $g(\theta, \cdot) = 0$ otherwise. The hybrid state space is given by $E = \{\{\theta\} \times E_\theta; \theta \in \mathbb{M}\}$, where for $\theta \in \{V_1, V_2, V_3, V_4\}$: $E_\theta = \mathbb{R}^2 \times (0, \infty) \times \mathbb{R}^2 \times (0, \infty)$ and for $\theta \in \{V_5, V_6, V_7, V_8\}$: $E_\theta = \mathbb{R}^6$. Always two delay transitions are pre-enabled: either $T_3$ or $T_4$ and either $T_5$ or $T_6$. This yields $\Lambda(V_1, \cdot) = \Lambda(V_5, \cdot) = \delta_4 + \delta_6$, $\Lambda(V_2, \cdot) = \Lambda(V_6, \cdot) = \delta_3 + \delta_6$, $\Lambda(V_3, \cdot) = \Lambda(V_7, \cdot) = \delta_3 + \delta_5$,

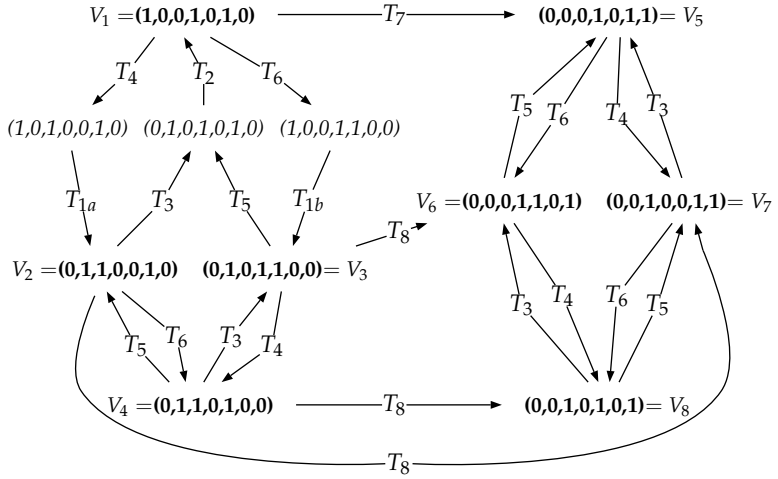



Fig. 5. Reachability graph for the SDCPN of Fig. 4. The nodes in bold type face correspond with the elements of the HSDE discrete state space $\mathbb{M}$.

$\Lambda(V_4,\cdot) = \Lambda(V_8,\cdot) = \delta_4 + \delta_5$. For the determination of elements $\psi$, $\rho$ and $\mu$, we first construct a probability measure $P_Q$, by making use of the reachability graph, the sets $\mathcal{D}$, $\mathcal{G}$ and $\mathcal{F}$ and the rules R0–R4. In Table 1, $P_Q(\theta', x'; \theta, x) = p$ denotes that if $(\theta, x)$ is the value of the HSDE state before the hybrid jump, then, with probability $p$, $(\theta', x')$ is the value of the HSDE state immediately after the jump. Since the continuous valued process jumps to the same value with probability 1, we find that $\psi(V^i, V^j, x, \underline{z}) = 0$ for all $V^i$, $V^j$, $x$, $\underline{z}$. Moreover, $\rho(V^i, V^j, x) = P_Q(V^i, x, V^j, x)$ and $\mu$ may be any given invertible probability measure.

Table 1. Example probability measure for size of jump

| | | |
|---|---|---|
| For $x \notin \partial E_{V_1}$: | $P_Q(V_2, x; V_1, x) = \frac{\delta_4}{\delta_4+\delta_6}$, | $P_Q(V_4, x; V_1, x) = \frac{\delta_6}{\delta_4+\delta_6}$ |
| For $x \in \partial E_{V_1}$: | $P_Q(V_5, x; V_1, x) = 1$ | |
| For $x \notin \partial E_{V_2}$: | $P_Q(V_3, x; V_2, x) = \frac{\delta_6}{\delta_3+\delta_6}$, | $P_Q(V_1, x; V_2, x) = \frac{\delta_3}{\delta_3+\delta_6}$ |
| For $x \in \partial E_{V_2}$: | $P_Q(V_6, x; V_2, x) = 1$ | |
| For $x \notin \partial E_{V_3}$: | $P_Q(V_4, x; V_3, x) = \frac{\delta_3}{\delta_3+\delta_5}$, | $P_Q(V_2, x; V_3, x) = \frac{\delta_5}{\delta_3+\delta_5}$ |
| For $x \in \partial E_{V_3}$: | $P_Q(V_7, x; V_3, x) = 1$ | |
| For $x \notin \partial E_{V_4}$: | $P_Q(V_3, x; V_4, x) = \frac{\delta_4}{\delta_4+\delta_5}$, | $P_Q(V_1, x; V_4, x) = \frac{\delta_5}{\delta_4+\delta_5}$ |
| For $x \in \partial E_{V_4}$: | $P_Q(V_8, x; V_4, x) = 1$ | |
| For all $x$: | $P_Q(V_6, x; V_5, x) = \frac{\delta_4}{\delta_4+\delta_6}$, | $P_Q(V_8, x; V_5, x) = \frac{\delta_6}{\delta_4+\delta_6}$ |
| For all $x$: | $P_Q(V_7, x; V_6, x) = \frac{\delta_6}{\delta_3+\delta_6}$, | $P_Q(V_5, x; V_6, x) = \frac{\delta_3}{\delta_3+\delta_6}$ |
| For all $x$: | $P_Q(V_8, x; V_7, x) = \frac{\delta_3}{\delta_3+\delta_5}$, | $P_Q(V_6, x; V_7, x) = \frac{\delta_5}{\delta_3+\delta_5}$ |
| For all $x$: | $P_Q(V_7, x; V_8, x) = \frac{\delta_4}{\delta_4+\delta_5}$, | $P_Q(V_5, x; V_8, x) = \frac{\delta_5}{\delta_4+\delta_5}$ |

With this, the SDCPN of the aircraft evolution example is uniquely mapped to an HSDE. If in addition, we want to make use of the HSDE properties of Proposition 4.1, i.e. the resulting HSDE solution process being adapted and a semi-martingale, we need to make sure that

HSDE conditions H1-H8 are satisfied. It is shown below that they are, under the following sufficient condition D1 for the example SDCPN.

**D1** For $P \in \{P_1, P_2\}$, there exist $K_P^v$, $L_P^v$, $K_P^w$ and $L_P^w$ such that for all $c, a \in \mathcal{C}(P)$,
$|\mathcal{V}_P(c)|^2 \leq K_P^v(1+|c|^2)$ and $|\mathcal{V}_P(c) - \mathcal{V}_P(a)|^2 \leq L_P^v|c-a|^2$ and
$\|\mathcal{W}_P(c)\|^2 \leq K_P^w(1+|c|^2)$ and $\|\mathcal{W}_P(c) - \mathcal{W}_P(a)\|^2 \leq L_P^w|c-a|^2$.

We verify that under condition D1, HSDE conditions H1-H8 hold true in this example.

**H1:** From the construction of $f$ and $g$ above we have for $\theta = V_1$: $|f(\theta,x)|^2 + \|g(\theta,x)\|^2 = |\mathcal{V}_1(x)|^2 + \|\mathcal{W}_1(x)\|^2 \leq K_{P_1}^v(1+|x|^2) + K_{P_1}^w(1+|x|^2) = K(\theta)(1+|x|^2)$, with $K(\theta) = (K_{P_1}^v + K_{P_1}^w)$. For $\theta = V_2, V_3, V_4$ the verification is with replacing $\mathcal{V}_1$, $\mathcal{W}_1$ by $\mathcal{V}_2$, $\mathcal{W}_2$.

**H2:** From the construction of $f$ and $g$ above we have for $\theta = V_1$: $|f(\theta,x) - f(\theta,y)|^2 + \|g(\theta,x) - g(\theta,y)\|^2 = |\mathcal{V}_1(x) - \mathcal{V}_1(y)|^2 + \|\mathcal{W}_1(x) - \mathcal{W}_1(y)\|^2 \leq L_{P_1}^v|x-y|^2 + L_{P_1}^w|x-y|^2 = L_r(\theta)|x-y|^2$ with $L_r(\theta) = L_{P_1}^v + L_{P_1}^w$. For $\theta = V_2, V_3, V_4$ replace $\mathcal{V}_1$, $\mathcal{W}_1$ by $\mathcal{V}_2$, $\mathcal{W}_2$.

**H3:** Since $\delta_3$–$\delta_6$ are constant, for all $\theta$, $\Lambda(\theta,\cdot)$ is bounded and continuous, with upper bound $C_\Lambda = \max\{\delta_4+\delta_6, \delta_3+\delta_6, \delta_3+\delta_5, \delta_4+\delta_5\}$.

**H4:** Since for all $\theta, \vartheta$, $P_Q(\vartheta,\cdot;\theta,x)$ is constant, we find $\rho(\vartheta,\theta,x) = P_Q(\vartheta,x,\theta,x)$ is continuous.

**H5 and H6:** These are satisfied due to $\psi(V^i, V^j, x, \underline{z}) = 0$ for all $V^i$, $V^j$, $x$, $\underline{z}$.

**H7:** This condition holds due to $\delta_3$–$\delta_6$ being finite and the fact that in this SDCPN example, there is no firing sequence of more than one guard transition.

**H8:** This condition holds for all $V_1, \ldots, V_8$, with metric $|a|^2 = \sum_i (a_i)^2$.

Thanks to this bisimilarity mapping we can now use HSDE tools to analyse the GSHP that is defined by the execution of the SDCPN model for the example.
In (Everdij & Blom, 2008) we showed how the SDCPN for the aircraft evolution example above is mapped to a GSHS. The main difference is that the GSHS transition measure $Q$ is defined by the probability measure $P_Q$ in Table 1 and that GSHS does not use elements $\psi$, $\rho$ and $\mu$, but apart of these details the differences with the mapping of SDCPN elements into HSDE elements are small. Thanks to this bisimilarity mapping, we can also use the automata framework to analyse the GSHS that is defined by the SDCPN model.

## 8. Conclusions

In order to combine the compositional specification power of Petri nets with the analysis power of Markov processes, (Malhotra & Trivedi, 1994) and (Muppala et al., 2000) developed a power hierarchy of dependability models. In (Everdij & Blom, 2003; 2005), the power hierarchy was extended with *dynamically coloured Petri nets* (DCPN) and *piecewise deterministic Markov processes* (PDP). In (Everdij & Blom, 2006), this power hierarchy was further extended by *stochastically and dynamically coloured Petri nets* (SDCPN) and *general stochastic hybrid process* (GSHP).
In this chapter the power-hierarchy has been further deepened by studying various ways to develop GSHP. We started in Section 2 by defining SDCPN and the resulting SDCPN process. In Section 3 we studied GSHP as an execution of a *general stochastic hybrid system* (GSHS). In Section 4 we defined GSHP as a solution of a *hybrid stochastic differential equation* (HSDE) and explained the differences between GSHS and HSDE. Next, in Section 5 we showed that GSHS, HSDE and SDCPN are bisimilar. In Sections 6-7, the results were illustrated with an aircraft

evolution example. The bisimilarities between SDCPN, GSHS and HSDE mean that each of them inherits the strengths of the other two formalisms. This has been depicted in Fig. 2 in the introduction. Hence, analysis tools designed for GSHS, HSDE and GSHP and their properties become available for SDCPN. Examples of GSHP properties are convergence in discretisation, existence of limits, existence of event probabilities, strong Markov properties, reachability analysis. Examples of GSHS features are their connection to formal methods in automata theory and optimal control theory. Examples of HSDE features are stochastic analysis tools for semi-martingales. At the same time, numerous SDCPN features such as natural expression of causal dependencies, concurrency and synchronisation mechanism, hierarchical and modular construction, and graphical representation become available when modelling GSHS, HSDE and GSHP through SDCPN. And these complementary advantages of SDCPN, GSHS, HSDE and GSHP perspectives tend to increase with the complexity of the system considered.

## 9. References

Blom, H. (2003). Stochastic hybrid processes with hybrid jumps, *Proceedings IFAC conference on analysis and design of hybrid system (ADHS), Saint-Malo, Brittany, France*, pp. 361–365.

Blom, H., Bakker, G., Everdij, M. & Van der Park, M. (2003). Stochastic analysis background of accident risk assessment for air traffic management, *Hybridge report*, D2.2. http://hosted.nlr.nl/public/hosted-sites/hybridge/.

Bujorianu, M. & Lygeros, J. (2003). Reachability questions in piecewise deterministic Markov processes, *in* O. Maler & A. Pnueli (eds), *Proceedings 6th international workshop on hybrid systems: computation and control (HSCC), Prague, Czech Republic*, Vol. 2623 of *Lecture notes in computer science (LNCS)*, Springer, pp. 126–140.

Bujorianu, M. & Lygeros, J. (2004). General stochastic hybrid systems: modelling and optimal control, *Proceedings 43rd conference on decision and control (CDC), Nassau, Bahamas*.

Bujorianu, M. & Lygeros, J. (2006). Toward a general theory of stochastic hybrid systems, *in* H. Blom & J. Lygeros (eds), *Stochastic hybrid systems: theory and safety critical applications*, Vol. 337 of *Lectures notes in control and information sciences (LNCIS)*, Springer, pp. 3–30.

Bujorianu, M., Lygeros, J. & Bujorianu, M. (2005). Different approaches on bisimulation for stochastic hybrid systems, *in* M. Morari & L. Thiele (eds), *Proceedings 8th international workshop on hybrid systems: computation and control (HSCC), Zürich, Switzerland*, Vol. 3414 of *Lecture notes in computer science (LNCS)*, pp. 198–214.

David, R. & Alla, H. (1994). Petri nets for modeling of dynamic systems - a survey, *Automatica* **30**(2): 175–202.

Davis, M. (1984). Piecewise deterministic Markov processes: a general class of non-diffusion stochastic models, *Journal royal statistical society (B)* **46**(3): 353–388.

Davis, M. (1993). *Markov models and optimization*, Vol. 49 of *Monographs on statistics and applied probability*, Chapman and Hall, London.

Elliott, R. (1982). *Stochastic calculus and applications*, Vol. 18 of *Applications of mathematics: Stochastic modelling and applied probability*, Springer-Verlag.

Elliott, R., Aggoun, L. & Moore, J. (1995). *Hidden Markov models: estimation and control*, Vol. 29 of *Applications of mathematics: stochastic modelling and applied probability*, Springer-Verlag.

Ethier, S. & Kurtz, T. (1986). *Markov processes, characterization and convergence*, Wiley series in probability and mathematical statistics, John Wiley & Sons, New York.

Everdij, M. & Blom, H. (2003). Petri nets and hybrid state Markov processes in a power-hierarchy of dependability models, *Proceedings IFAC conference on analysis and design of hybrid system (ADHS), Saint-Malo, Brittany, France*, pp. 355–360.

Everdij, M. & Blom, H. (2005). Piecewise deterministic Markov processes represented by dynamically coloured Petri nets, *in* S. Jacka (ed.), *Stochastics: an international journal of probability and stochastic processes*, Vol. 77, number 1, Taylor & Francis, pp. 1–29.

Everdij, M. & Blom, H. (2006). Hybrid Petri nets with diffusion that have into-mappings with generalised stochastic hybrid processes, *in* H. Blom & J. Lygeros (eds), *Stochastic hybrid systems: theory and safety critical applications*, Vol. 337 of *Lectures notes in control and information sciences (LNCIS)*, Springer, pp. 31–63.

Everdij, M. & Blom, H. (2008). Enhancing hybrid state Petri nets with the analysis power of stochastic hybrid processes, *in* B. Lennartson, M. Fabian, K. Åkesson, A. Giua & R. Kumar (eds), *Proceedings 9th International Workshop on Discrete Event Systems (WODES), Goeteborg, Sweden*, pp. 400–405.

Everdij, M., Klompstra, M., Blom, H. & Klein Obbink, B. (2006). Compositional specification of a multi-agent system by stochastically and dynamically coloured Petri nets, *in* H. Blom & J. Lygeros (eds), *Stochastic hybrid systems: theory and safety critical applications*, Vol. 337 of *Lectures notes in control and information sciences (LNCIS)*, Springer, pp. 325–350.

Frehse, G. (2008). PHAVer: algorithmic verification of hybrid systems past HyTech, *International journal on software tools for technology transfer* **10**(3): 263–279.

Giua, A. (1999). Bibliography on hybrid Petri nets. http://bode.diee.unica.it/∼hpn/.

Hu, J., Lygeros, J. & Sastry, S. (2000). Towards a theory of stochastic hybrid systems, *in* N. Lynch & B. Krogh (eds), *Proceedings 3rd international workshop on hybrid systems: computation and control (HSCC), Pittsburgh, Pennsylvania, USA*, Vol. 1790 of *Lecture notes in computer science (LNCS)*, Springer Verlag, pp. 160–173.

Krystul, J. (2006). *Modelling of stochastic hybrid systems with applications to accident risk assessment*, PhD thesis, University of Twente, The Netherlands.

Krystul, J. & Blom, H. (2005). Generalised stochastic hybrid processes as strong solutions of stochastic differential equations, *Hybridge report*, D2.3. http://hosted.nlr.nl/public/hosted-sites/hybridge/.

Krystul, J., Blom, H. & Bagchi, A. (2007). *Stochastic hybrid systems*, number 24 in *Control engineering series*, Taylor and Francis / CRC Press, chapter 2: Stochastic differential equations on hybrid state spaces, pp. 15–45.

Kwiatkowska, M., Norman, G. & Parker, D. (2004). Probabilistic symbol model checking with PRISM: a hybrid approach, *International journal on software tools for technology transfer* **6**(2): 128–142.

Labinaz, G., Bayoumi, M. & Rudie, K. (1997). A survey of modeling and control of hybrid systems, *Annual reviews of control* **21**: 79–92.

Lepeltier, J. & Marchal, B. (1976). Problème des martingales et équations différentielles stochastiques associées à un opérateur intégro-différentiel, *Annales de l'Institute Henri Poincaré* **Section B - XII**(1): 43–103.

Malhotra, M. & Trivedi, K. (1994). Power-hierarchy of dependability-model types, *IEEE transactions on reliability* **R-43**(3): 493–502.

Muppala, J., Fricks, R. & Trivedi, K. (2000). Techniques for system dependability evaluation, *in* W. Grasman (ed.), *Computational probability*, Kluwer academic publishers, The Netherlands, pp. 445–480.

Strubbe, S. & Van der Schaft, A. (2004). Semantics, bisimulation and interaction-structures for the CPDP model, *Hybridge report*, D4.3. http://hosted.nlr.nl/public/hosted-sites/hybridge/.

Strubbe, S. & Van der Schaft, A. (2005). Bisimulation for communicating piecewise deterministic Markov processes (CPDPs), *in* M. Morari & L. Thiele (eds), *Proceedings 8th international workshop on hybrid systems: computation and control (HSCC), Zürich, Switzerland*, Vol. 3414 of *Lecture notes in computer science (LNCS)*, pp. 623–639.

Van der Schaft, A. (2004). Equivalence of dynamical systems by bisimulation, *IEEE transactions on automatic control* **49**(12): 2160–2172.

## Appendix A: Proof of Theorem 5.3

Consider an arbitrary HSDE (1)-(8) with elements ($\mathbb{M}$, $E$, $f$, $g$, $\mu_{\theta_0,X_0}$, $\Lambda$, $\psi$, $\rho$, $\mu$, $p_P$, $\{W_t\}$). We assume that the stochastic differential equations defined by $f$ and $g$ have probabilistically unique solutions and that $\Lambda$ is bounded. First, we characterise SDCPN elements ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) in terms of HSDE elements ($\mathbb{M}$, $E$, $f$, $g$, $\mu_{\theta_0,X_0}$, $\Lambda$, $\psi$, $\rho$, $\mu$, $p_P$, $\{W_t\}$). The thus constructed SDCPN is referred to as SDCPN$^{HSDE}$. Subsequently, we show that the SDCPN$^{HSDE}$ stochastic process is probabilistically equivalent to the stochastic process defined by the original HSDE.

### A.1 Construction of SDCPN$^{HSDE}$ elements

We provide an into-mapping that characterises SDCPN elements ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$) in terms of HSDE elements ($\mathbb{M}$, $E$, $f$, $g$, $\mu_{\theta_0,X_0}$, $\Lambda$, $\psi$, $\rho$, $\mu$, $p_P$, $\{W_t\}$).

$\mathcal{P}$ $= \{P_\theta; \theta \in \mathbb{M}\}$. Hence, for each $\theta \in \mathbb{M}$, there is one place $P_\theta$. The places are ordered $P_{\vartheta_1}, \ldots, P_{\vartheta_N}$ according to $\mathbb{M} = \{\vartheta_1, \ldots, \vartheta_N\}$.

$\mathcal{T}$ $= \mathcal{T}_G \cup \mathcal{T}_D \cup \mathcal{T}_I$, with $\mathcal{T}_I = \emptyset$, $\mathcal{T}_G = \{T^G_\theta; \theta \in \mathbb{M}\}$, $\mathcal{T}_D = \{T^D_\theta; \theta \in \mathbb{M}\}$. Hence, for each $\theta \in \mathbb{M}$ there is one guard transition $T^G_\theta$ and one delay transition $T^D_\theta$.

$\mathcal{A}$ $= \mathcal{A}_O \cup \mathcal{A}_E \cup \mathcal{A}_I$, with $|\mathcal{A}_I| = 0$, $|\mathcal{A}_E| = 0$, and $|\mathcal{A}_O| = 2N + 2N^2$, where $N = |\mathbb{M}|$. Hence, there are no inhibitor arcs or enabling arcs in this SDCPN$^{HSDE}$ constructed, and the number of ordinary arcs is $2N + 2N^2$.

$\mathcal{N}$: The node function maps each arc in $\mathcal{A} = \mathcal{A}_O$ to a pair of nodes. These connected pairs of nodes are: $\{(P_\theta, T^G_\theta); \theta \in \mathbb{M}\} \cup \{(P_\theta, T^D_\theta); \theta \in \mathbb{M}\} \cup \{(T^G_\theta, P_\vartheta); \theta, \vartheta \in \mathbb{M}\} \cup \{(T^D_\theta, P_\vartheta); \theta, \vartheta \in \mathbb{M}\}$. Hence, each place $P_\theta$ ($\theta \in \mathbb{M}$) has two outgoing arcs: one to guard transition $T^G_\theta$ and one to delay transition $T^D_\theta$. Each transition has $N$ outgoing arcs: one arc to each place in $\mathcal{P}$.

$\mathcal{S}$ $= \{\mathbb{R}^n\}$.

$\mathcal{C}$: For all $\theta \in \mathbb{M}$, $\mathcal{C}(P_\theta) = \mathbb{R}^n$

$\mathcal{I}$: For all $\theta_0 \in \mathbb{M}$ and $X_0 \in \mathcal{C}(P_{\theta_0}) = \mathbb{R}^n$, $\mathcal{I}(M^{\theta_0}, X_0) = \mu_{\theta_0,x_0}(\theta_0, X_0)$, where $M^\theta$ is the $|\mathcal{P}|$-dimensional vector that has a one at the element corresponding to place $P_\theta$ and zeros elsewhere.

$\mathcal{V}$: For all $\theta \in \mathbb{M}$, $\mathcal{V}_{P_\theta}(\cdot) = f(\theta, \cdot)$.

$\mathcal{W}$: For all $\theta \in \mathbb{M}$, $\mathcal{W}_{P_\theta}(\cdot) = g(\theta, \cdot)$.

$\mathcal{G}$: For all $\theta \in \mathbb{M}$, $\mathcal{G}_{T^G_\theta} = E_\theta$.

$\mathcal{D}$: For all $\theta \in \mathbb{M}$, $\mathcal{D}_{T_\theta^D}(\cdot) = \Lambda(\theta, \cdot)$. Since we assumed that $\Lambda$ is bounded, e.g. $\Lambda(\theta, \cdot) \leq C_\Lambda$, we find that $\mathcal{D}_{T_\theta^D}(\cdot)$ is bounded as well, and its upperbound is $C_\delta = C_\Lambda$.

$\mathcal{F}$: Define for particular transition $T$, $e^{\vartheta'}$ as the vector of length $N$ containing a one at the component corresponding with the arc from transition $T$ to place $P_{\vartheta'}$ and zeros elsewhere. Then for all $\theta \in \mathbb{M}$, and for $T \in \{T_\theta^G, T_\theta^D\}$, $\mathcal{F}_T(e^{\vartheta'}, x'; x) = F_T^Q(\vartheta', x'; \theta, x)$, for all $x \in E_\theta \cup \partial E_\theta$, $\vartheta' \in \mathbb{M}$ and $x' \in E_{\vartheta'}$, where $F_T^Q$ is defined through

$$F_T^Q(\{\vartheta'\} \times A'; \theta, x) = \rho(\vartheta', \theta, x) \int_{\mathbb{R}^d} \mathbf{1}_{A'}(x + \psi(\vartheta', \theta, x, \underline{z})) \mu(d\underline{z}) \tag{9}$$

### A.2 Probabilistic equivalence

Next, we show that the SDCPN$^{HSDE}$ stochastic process is probabilistically equivalent to the stochastic process defined by the original HSDE. This is done by showing: Equivalence of initial states; Equivalence of continuous evolution until first jump; Equivalence of time of jumps; Equivalence of size of jumps; Equivalence of processes after the first jump.

**Equivalence of initial states:**

The initial marking of the SDCPN$^{HSDE}$ is defined by $\mathcal{I}(M^{\theta_0}, X_0) = \mu_{\theta_0, X_0}(\theta_0, X_0)$, where $M^\theta$ is the $N$-dimensional vector that has a one at the element corresponding to place $P_\theta$ and zeros elsewhere. Therefore, with probability $\mathcal{I}(M^{\theta_0}, X_0)$, at time $t = \tau_0$ there is one token in place $P_{\theta_0}$ which has colour $X_0$. The initial state of the HSDE is $(\theta_0, X_0)$ with probability $\mu_{\theta_0, X_0}(\theta_0, X_0)$. Due to the mapping between the places $P_\theta \in \mathcal{P}$ and the modes $\theta \in \mathbb{M}$, the initial states of SDCPN$^{HSDE}$ and HSDE are probabilistically equivalent.

**Equivalence of continuous evolution until first jump:**

The continuous part of the SDCPN$^{HSDE}$ stochastic process equals the vector that collects all token colours. Since there is only one token in the constructed SDCPN$^{HSDE}$ at all times, this vector equals the colour of this single token. Until the first jump, this colour follows the stochastic differential equation $dC_t^{P_{\theta_0}} = \mathcal{V}_{P_{\theta_0}}(C_t^{P_{\theta_0}})dt + \mathcal{W}_{P_{\theta_0}}(C_t^{P_{\theta_0}})dW_t^{P_{\theta_0}}$ which has probabilistically unique solution $C_t^{P_{\theta_0}}$.

In the original HSDE solution process, the continuous process until the first jump follows stochastic differential equation $dX_t^0 = f(\theta_t^0, X_t^0)dt + g(\theta_t^0, X_t^0)dW_t + \int_{\mathbb{R}^d} \psi(\theta_t^0, \theta_{t-}^0, X_{t-}^0, \underline{z}) p_P(dt, (0, \Lambda(\theta_{t-}^0, X_{t-}^0)] \times d\underline{z})$ where $d\theta_t^0 = \sum_{i=0}^N (\vartheta_i - \theta_{t-}^0) p_P(dt, (\Sigma_{i-1}(\theta_{t-}^0, X_{t-}^0), \Sigma_i(\theta_{t-}^0, X_{t-}^0)] \times \mathbb{R}^d)$. Until the first jump, the Poisson terms in the stochastic differential equations above are equal to zero. What remains is: $d\theta_t^0 = 0$ and $dX_t^0 = f(\theta_t^0, X_t^0)dt + g(\theta_t^0, X_t^0)dW_t$, which are assumed to have a probabilistically unique solution $\theta_t^0$ and $X_t^0$.

Due to equivalence of initial states $M^{\theta_0} \equiv \theta_0$ and $C_0 = X_0$, equivalence of drift coefficients $\mathcal{V}_{P_{\theta_0}}(\cdot) = f(\theta_0, \cdot)$ and equivalence of diffusion coefficients $\mathcal{W}_{P_{\theta_0}}(\cdot) = g(\theta_0, \cdot)$, as long as no jumps occur, we derive that for $t \geq \tau_0 = 0$, $M^{\theta_t} = \theta_t^0$ and $X_t^0 = C_t^{P_{\theta_0}}$.

**Equivalence of time of jumps:**

For the SDCPN$^{HSDE}$, for each arbitrary place in which the initial token may reside, two transitions are pre-enabled: a guard transition and a delay transition. If either of them becomes enabled and fires, then the other becomes disabled. The time until the guard transition is enabled is $t_*(M^{\theta_0}, C_0) \triangleq \inf\{t - \tau_0 > 0 \mid C_t^{P_{\theta_0}} \in \partial\mathcal{G}_{T_{\theta_0}^G}\}$. The time until the delay transition is

enabled is $\sigma_1^{T_{\theta_0}^D} = D_{T_{\theta_0}^D}^{inv}(U_1)$, with $D_{T_{\theta_0}^D}^{inv}(u) = \inf\{t - \tau_0 \mid \exp(-\int_{\tau_0}^{t} \mathcal{D}_{T_{\theta_0}^D}(C_s^{P_{\theta_0}})ds) \leq u\}$ and $U_1 \sim U[0,1]$.

For HSDE, from Equation (6), using $k = 0$ and $\tau_0^b = \tau_0$, the time at which the continuous state first hits the boundary of its state space is $\tau_1^b \triangleq \inf\{t > \tau_0 \mid (\theta_t^0, X_t^0) \in \{\{\theta\} \times \partial E_\theta; \theta \in \mathbb{M}\}\}$. It is easily seen that as long as $\theta_t^0 = \theta_0$, then due to $X_t^0 = C_t^{P_{\theta_0}}$ and the equality $\partial \mathcal{G}_{T_{\theta_0}^G} = \partial E_{\theta_0}$, we have that $\inf\{t > \tau_0 \mid (\theta_t^0, X_t^0) \in \{\{\theta\} \times \partial E_\theta; \theta \in \mathbb{M}\}\} = \tau_0 + \inf\{t - \tau_0 > 0 \mid C_t^{P_{\theta_0}} \in \partial \mathcal{G}_{T_{\theta_0}^G}\}$, hence $\tau_1^b = \tau_0 + t_*(M^{\theta_0}, C_0)$. However, there is a possibility that at some time $\tau_1^p < \tau_1^b$, the HSDE solution process state makes a jump due to the Poisson random measure generating a point: Consider Equations (3) and (4), for $k = 0$. A jump is generated when $\sum_{i=1}^{N}(\vartheta_i - \theta_{t-}^0)p_P(dt, (\Sigma_{i-1}(\theta_{t-}^0, X_{t-}^0), \Sigma_i(\theta_{t-}^0, X_{t-}^0)] \times \mathbb{R}^d) \neq 0$ or when $\int_{\mathbb{R}^d} \psi(\theta_t^0, \theta_{t-}^0, X_{t-}^0, \underline{z}) p_P(dt, (0, \Lambda(\theta_{t-}^0, X_{t-}^0)] \times d\underline{z}) \neq 0$, or both. Consider the Poisson random measure in Equation (4), i.e. $p_P(dt, (0, \Lambda(\theta_{t-}^0, X_{t-}^0)] \times d\underline{z})$, which is equal to zero, except at singular times when it generates a multivariate point $(\{\tau_1^p\}, \{z_1\}, \{\underline{z}\})$. Due to the Poisson random measure being homogeneous and due to $\Lambda(\theta_{t-}^0, X_{t-}^0) \leq C_\Lambda$, the point $(\{\tau_1^p\}, \{z_1\}, \{\underline{z}\})$ is generated as follows: Generate a triple $(\varepsilon_1, \nu_1, \underline{\nu}_1)$, with $\varepsilon_1 \sim \text{Exp}(C_\Lambda)$, $\nu_1 \sim U[0, C_\Lambda]$ and $\underline{\nu}_1 \sim \mu$. Accept this triple if $\nu_1 \leq \Lambda(\theta_{\tau_0+\varepsilon_1-}^0, X_{\tau_0+\varepsilon_1-}^0)$, otherwise reject it. If it is accepted then $\tau_1^p = \tau_0 + \varepsilon_1$, $z_1 = \nu_1$ and $\underline{z} = \underline{\nu}_1$. If it is not accepted then another triple $(\varepsilon_2, \nu_2, \underline{\nu}_2)$ is generated with $\varepsilon_2 \sim \text{Exp}(C_\Lambda)$, $\nu_2 \sim U[0, C_\Lambda]$ and $\underline{\nu}_2 \sim \mu$, and this triple is accepted if $\nu_2 \leq \Lambda(\theta_{\tau_0+\varepsilon_1+\varepsilon_2-}^0, X_{\tau_0+\varepsilon_1+\varepsilon_2-}^0)$. If it is accepted then $\tau_1^0 = \tau_0 + \varepsilon_1 + \varepsilon_2$, $z_1 = \nu_2$ and $\underline{z} = \underline{\nu}_2$. If it is not accepted then another triple $(\varepsilon_3, \nu_3, \underline{\nu}_3)$ is generated, and so on. Hence if $(\varepsilon_r, \nu_r, \underline{\nu}_r)$ is the first triple that is accepted then $\tau_1^p = \tau_0 + \sum_{n=1}^{r} \varepsilon_n$ and $z_1 = \nu_r$ and $\underline{z} = \underline{\nu}_r$. The interarrival times of the triples accepted through this mechanism are exponential with intensity $\Lambda$. In addition, due to $\mathcal{D}_{T_\theta^D}(\cdot) = \Lambda(\theta, \cdot)$, we find that $\tau_1^p - \tau_0$ is probabilistically equivalent to $\sigma_1^{T_{\theta_0}^D}$. For HSDE, the time of the first jump is equal to the minimum of $\tau_1^b$ and $\tau_1^p$. Due to the reasoning above, this time of first jump is probabilistically equivalent to the time of first jump of the SDCPN$^{HSDE}$.

#### Equivalence of size of jumps

For the SDCPN$^{HSDE}$, the jump size is determined by the firing measure $\mathcal{F}_T$ of the enabled transition $T$: for all $\theta \in \mathbb{M}$ and $T \in \{T_\theta^G, T_\theta^D\}$, $\mathcal{F}_T(e^{\vartheta'}, x'; x) = F_T^Q(\vartheta', x'; \theta, x)$, for all $x \in E_\theta \cup \partial E_\theta$, $\vartheta' \in \mathbb{M}$ and $x' \in E_{\vartheta'}$, where $F_T^Q$ is defined through

$$F_T^Q(\{\vartheta'\} \times A'; \theta, x) = \rho(\vartheta', \theta, x) \int_{\mathbb{R}^d} \mathbf{1}_{A'}(x + \psi(\vartheta', \theta, x, \underline{z})) \mu(d\underline{z})$$

For HSDE, the size of jumps is generated as follows: In case of a jump generated by Poisson random measure at time $t = \tau_1^p$, the size of jump in $\{\theta_t^0\}$ is given by

$$\theta_{\tau_1^p}^0 - \theta_{\tau_1^p-}^0 = \sum_{i=1}^{N} (\vartheta_i - \theta_{\tau_1^p-}^0) p_P(dt, (\Sigma_{i-1}(\theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0), \Sigma_i(\theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0)] \times \mathbb{R}^d)$$

and the size of jump in $\{X_t^0\}$ is given by

$$X_{\tau_1^p}^0 - X_{\tau_1^p-}^0 = \int_{\mathbb{R}^d} \psi(\theta_{\tau_1^p}^0, \theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0, \underline{z}) p_P(dt, (0, \Lambda(\theta_{\tau_1^p-}^0, X_{\tau_1^p-}^0)] \times d\underline{z})$$

Now use that the Poisson random measure has generated a point $(\{\tau_1^p\}, \{z_1\}, \{\underline{z}\})$, with $z_1 = \nu_r$ and $\underline{z} = \underline{\nu}_r$ as described above. Random variable $z_1$ is used as follows: Notice that by Equation (5) and definition of $\rho$, for all $\theta \in \mathbb{M}$ and all $x \in \mathbb{R}^n$, the interval $(0, \Lambda(\theta, x)]$ is divided into subintervals $(\Sigma_{i-1}(\theta, x), \Sigma_i(\theta, x)]$, i.e. $(0, \Lambda(\theta, x)] = (\Sigma_0(\theta, x), \Sigma_1(\theta, x)] \cup (\Sigma_1(\theta, x), \Sigma_2(\theta, x)] \cup \cdots \cup (\Sigma_{N-1}(\theta, x), \Sigma_N(\theta, x)]$, where $\Sigma_0(\theta, x) = 0$ and $\Sigma_N(\theta, x) = \Lambda(\theta, x)$. The $i$th interval, i.e. $(\Sigma_{i-1}(\theta, x), \Sigma_i(\theta, x)]$ has a weight $\rho(\vartheta_i, \theta, x) = (\Sigma_i(\theta, x) - \Sigma_{i-1}(\theta, x))/\Lambda(\theta, x)$, with $\sum_{i=1}^N \rho(\vartheta_i, \theta, x) = 1$. Due to $z_1 \in (0, \Lambda(\theta^0_{\tau_1^p -}, X^0_{\tau_1^p -})]$, there exists $j \in \{1, \ldots, N\}$ such that $z_1 \in (\Sigma_{j-1}(\theta^0_{\tau_1^p -}, X^0_{\tau_1^p -}), \Sigma_j(\theta^0_{\tau_1^p}, X^0_{\tau_1^p})]$. This makes $p_P(dt, (\Sigma_{i-1}(\theta^0_{\tau_1^p -}, X^0_{\tau_1^p -}), \Sigma_i(\theta^0_{\tau_1^p -}, X^0_{\tau_1^p -})] \times \mathbb{R}^d) = 1$ if $i = j$ and $= 0$ for $i \neq j$. Therefore, $\theta^0_{\tau_1^p} - \theta^0_{\tau_1^p -} = \vartheta_j - \theta^0_{\tau_1^p -}$, i.e. at time $\tau_1^p$, $\theta_t$ jumps from $\theta^0_{\tau_1^p -} = \theta_0$ to $\theta^0_{\tau_1^p} = \vartheta_j$. Next, the random variable $\underline{z}$ is used to determine $X^0_{\tau_1^p} - X^0_{\tau_1^p -}$, i.e. in $(\{\tau_1^p\}, \{z_1\}, \{\underline{z}\})$, $p_P(dt, (0, \Lambda(\theta^0_{t-}, X^0_{t-})] \times d\underline{z}) = 1$ and is zero elsewhere. Therefore, $X^0_{\tau_1^p} - X^0_{\tau_1^p -} = \psi(\vartheta_j, \theta^0_{\tau_1^p -}, X^0_{\tau_1^p -}, \underline{z})$. This gives that at time $\tau_1^p$, $X_t$ jumps from $X^0_{\tau_1^p -}$ to $X^0_{\tau_1^p -} + \psi(\vartheta_j, \theta^0_{\tau_1^p -}, X^0_{\tau_1^p -}, \underline{z})$. From this, we find that the probability for $(\theta_t, X_t)$ to jump into $(\{\vartheta_j\}, A)$, given that the state right before the jump is $(\theta^0_{\tau_1^p -}, X^0_{\tau_1^p -})$, is equal to the probability that $z_1$ is in $(\Sigma_{j-1}(\theta^0_{\tau_1^p -}, X^0_{\tau_1^p -}), \Sigma_j(\theta^0_{\tau_1^p -}, X^0_{\tau_1^p -})]$, times the probability that $X^0_{\tau_1^p -} + \psi(\vartheta_j, \theta^0_{\tau_1^p -}, X^0_{\tau_1^p -}, \underline{z})$ is in $A$. This probability is equal to

$$\rho(\vartheta_j, \theta^0_{\tau_1^p -}, X^0_{\tau_1^p -}) \int_{\mathbb{R}^d} \mathbf{1}_A(X^0_{\tau_1^p -} + \psi(\vartheta_j, \theta^0_{\tau_1^p -}, X^0_{\tau_1^p -}, \underline{z}))\mu(d\underline{z})$$

which is equal to $Q(\{\vartheta_j\} \times A; \theta^0_{\tau_1^p -}, X^0_{\tau_1^p -})$, according to Equation (8).

For boundary hitting type of jumps, the size of jump is given by Equation (7), i.e.

$$\mathbb{P}\{\theta^1_{\tau_1^b} = \vartheta, X^1_{\tau_1^b} \in A \mid \theta^0_{\tau_1^b -} = \theta, X^0_{\tau_1^b -} = x\} = Q(\{\vartheta\} \times A; \theta, x)$$

This shows that the jump size mechanisms for Poisson random measure type of jumps and boundary hitting type of jumps are the same. Also note that for all $\vartheta'$, $x'$, $\theta$ and $x$, and $T \in \mathcal{T}_D \cup \mathcal{T}_G$, $F^Q_T(\vartheta', x'; \theta, x) = Q(\vartheta', x'; \theta, x)$. This means that the SDCPN$^{HSDE}$ state after the jump and the HSDE solution process state after the jump are probabilistically equivalent.

**Equivalence of processes after the first jump:**
From $\tau_1 = \min\{\tau_1^b, \tau_1^p\}$ onwards, the probabilistic equivalence of the HSDE and SDCPN$^{HSDE}$ processes is shown in the same way. If $\tau_1 = \tau_1^p$, then Equations (3) and (4) are used for $k = 0$; if $\tau_1 = \tau_1^b$ then these equations are used for $k = 1$. From stopping time $\tau_{n-1}$ to stopping time $\tau_n$ the HSDE solution process and the associated SDCPN$^{HSDE}$ process have probabilistically equivalent paths and probabilistically equivalent stopping times. Due to the unique definition of the SDCPN$^{HSDE}$ stochastic process at times when transitions fire, the SDCPN$^{HSDE}$ state at stopping times is also equivalent to the HSDE solution process state at the stopping times and both processes are càdlàg.
This completes the proof of Theorem 5.3.

## Appendix B: Proof of Theorem 5.4

Consider an arbitrary SDCPN $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{N}, \mathcal{S}, \mathcal{C}, \mathcal{I}, \mathcal{V}, \mathcal{W}, \mathcal{G}, \mathcal{D}, \mathcal{F})$ that satisfies rules R0–R4. It is assumed that in the initial marking no immediate transitions are enabled, that the delay

rates $\mathcal{D}_T$ are bounded, and that for $t \to \infty$, the number of tokens remains finite. First, we characterise the HSDE elements ($\mathbb{M}$, $E$, $f$, $g$, $\mu_{\theta_0,X_0}$, $\Lambda$, $\psi$, $\rho$, $\mu$, $p_P$, $\{W_t\}$), in terms of SDCPN elements, where it is assumed that $\mu$ is given. The thus constructed HSDE is referred to as HSDE$^{SDCPN}$. Subsequently, we show that the HSDE$^{SDCPN}$ solution process is probabilistically equivalent to the stochastic process defined by the original SDCPN.

### B.1 Construction of HSDE$^{SDCPN}$ elements

We provide an into-mapping that characterises HSDE$^{SDCPN}$ elements ($\mathbb{M}$, $E$, $f$, $g$, $\mu_{\theta_0,X_0}$, $\Lambda$, $\psi$, $\rho$, $\mu$, $p_P$, $\{W_t\}$) in terms of SDCPN elements ($\mathcal{P}$, $\mathcal{T}$, $\mathcal{A}$, $\mathcal{N}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{I}$, $\mathcal{V}$, $\mathcal{W}$, $\mathcal{G}$, $\mathcal{D}$, $\mathcal{F}$).

$\mathbb{M}$ The characterisation of $\mathbb{M}$ in terms of SDCPN elements is by means of the reachability graph (RG). The nodes in the RG are token distributions, written as row vectors $(m_1, \ldots, m_{|\mathcal{P}|})$, where $m_i$ is the number of tokens in place $P_i$. Arrows between nodes are labelled by transitions, and indicate how the number of tokens in the places change due to transition firings. Then $\mathbb{M}$ is composed of those nodes in the reachability graph that do not enable an immediate transition, and $N = |\mathbb{M}|$.

$E$ For each $\theta \in \mathbb{M}$, corresponding with node $m = (m_1, \ldots, m_{|\mathcal{P}|})$ in the RG, define $d(\theta) = \sum_{i=1}^{|\mathcal{P}|} m_i n(P_i)$. If under token distribution $\theta$, no guard transitions are pre-enabled, then $E_\theta = \mathbb{R}^{d(\theta)}$. If under token distribution $\theta$, one or more guard transitions are pre-enabled, then $E_\theta = \mathbb{R}^{d(\theta)} \setminus \partial E_\theta$, where $\partial E_\theta$ is constructed as follows: Without loss of generality, suppose that under token distribution $\theta$, the multi-set of pre-enabled guard transitions is $T_1, \ldots, T_k$. This set may contain one transition multiple times, if such transition evaluates multiple input token vectors in parallel. Suppose $\{P_{i_1}, \ldots, P_{i_{r_i}}\} = P(A_{in,OE}(T_i))$ are the input places of $T_i$ that are connected to $T_i$ by means of ordinary or enabling arcs. This set may contain one place multiple times if such place is connected to $T_i$ by multiple arcs (input arcs of $T_i$). Define $d_i = \sum_{j=1}^{r_i} n(P_{i_j})$, then $\partial E_\theta = \partial \mathcal{G}'_{T_1} \cup \ldots \cup \partial \mathcal{G}'_{T_k}$, where $\mathcal{G}'_{T_i} = [\mathcal{G}_{T_i} \times \mathbb{R}^{d(\theta)-d_i}] \in \mathbb{R}^{d(\theta)}$. Here $[\cdot]$ denotes a special ordering of all vector elements: Vector elements are ordered according to the unique ordering of places and to the unique ordering of tokens within their place defined for SDCPN. Finally, $E = \{\{\theta\} \times E_\theta; \theta \in \mathbb{M}\}$.

$f$ For each $\theta \in \mathbb{M}$ and $x \in E_\theta$, $f(\theta, x) = \mathrm{Col}_{i=1}^{|\mathcal{P}|} \left\{ \mathrm{Col}_{j=1}^{m_i} \{\mathcal{V}_{P_i}(c_{ij})\} \right\}$, where $x = \mathrm{Col}_{i=1}^{|\mathcal{P}|} \left\{ \mathrm{Col}_{j=1}^{m_i} \{c_{ij}\} \right\}$ and $\theta$ corresponds to $(m_1, \ldots, m_{|\mathcal{P}|})$.

$g$: For each $\theta \in \mathbb{M}$ and $x \in E_\theta$,
$g(\theta, x) = \mathrm{Row}\{\mathrm{Diag}_{i=1}^{|\mathcal{P}|} \left\{ \mathrm{Diag}_{j=1}^{m_i} \{\mathcal{W}_{P_i}(c_{ij})\} \right\}, O^{\Sigma_{i=1}^{|\mathcal{P}|}(m_i^{max}-m_i)h(P_i)}\}$, where

- $O^{\Sigma_{i=1}^{|\mathcal{P}|}(m_i^{max}-m_i)h(P_i)}$ is a square matrix of dimension $(\Sigma_{i=1}^{|\mathcal{P}|}(m_i^{max} - m_i)h(P_i)) \times (\Sigma_{i=1}^{|\mathcal{P}|}(m_i^{max} - m_i)h(P_i))$ that contains only zeros. In the $g(\theta, \cdot)$ constructed above it is put to the right of the block that contains the matrices $\mathcal{W}_{P_i}$.
- $m_i^{max} = \max_{\theta \in \mathbb{M}}\{m_i \mid \theta = (m_1, \ldots, m_{|\mathcal{P}|})\}$ is the maximum number of tokens that exists in place $P_i$. This maximum $m_i^{max}$ exists due to the condition that for $t \to \infty$ the number of tokens remains finite.

$\mu_{\theta_0,X_0}$: $\mu_{\theta_0,X_0}(M_0, C_0) = \mathcal{I}(M_0, C_0)$ for all $M_0$ and $C_0$, where $M_0 = (M_{1,0}, \ldots, M_{|\mathcal{P}|,0})$, with $M_{i,0}$ the initial number of tokens in place $P_i$, with the places ordered according to the

unique ordering adopted for SDCPN, and $C_0 \in \mathbb{R}^{d(\theta_0)}$ containing the colours of these tokens. Due to the condition that no immediate transitions are enabled in the initial marking (which prevents vanishing token distributions to be current at the initial time), the constructed $M_0$ and $C_0$ are uniquely defined.

**$\Lambda$:** For each $\theta \in \mathbb{M}$ and $x \in E_\theta$, $\Lambda(\theta, x) = \sum_{n=1}^{k} \mathcal{D}_{T_n}(c^{T_n})$, where $T_1, \ldots, T_k$ refers to the multi-set of transitions in $\mathcal{T}_D$ that, under token distribution $\theta$, are pre-enabled, and $c^{T_n}$ are the respective elements of $x$ that are used to pre-enable these transitions. This set $T_1, \ldots, T_k$ may contain one transition multiple times, if multiple input token vectors are evaluated in parallel. If the set of pre-enabled delay transitions is empty in $\theta$, then $\Lambda(\theta, \cdot) = 0$.

**$\psi$, $\rho$, $\mu$:** we make use of the assumption that $\mu$ is given. As part of the construction, define a probability measure $P_Q(\theta', A; \theta, x)$, the value of which equals the probability that if a jump occurs, and if the value of the HSDE solution process just prior to the jump is $(\theta, x)$, then the value of the HSDE solution process just after the jump is in $(\theta', A)$. Probability $P_Q(\theta', A; \theta, x)$ is characterised in terms of the SDCPN by the reachability graph (RG), elements $\mathcal{D}$, $\mathcal{G}$ and Rules R0–R4 and the set $\mathcal{F}$. This is done in four steps, precisely following the characterisation of the GSHS transition measure $Q$ in terms of SDCPN elements in the appendix of (Everdij & Blom, 2006). Next, we characterise $\psi$ and $\rho$ in terms of this result: For HSDE, due to Equation (7), the probability that given a jump from $(\theta, x)$, the state after the jump is in $(\theta', A)$ is given by $Q(\{\theta'\} \times A; \theta, x)$ hence we find that $P_Q = Q$. Here, $Q$ is given by Equation (8). From this, we find

$$\rho(\theta', \theta, x) = Q(\{\theta'\} \times \mathbb{R}^n; \theta, x)$$

Next write, for any $x'$,

$$\begin{aligned} Q(\{\theta'\}, x'; \theta, x) &= \rho(\theta', \theta, x) \cdot \mathbb{P}\{x + \psi(\theta', \theta, x, \underline{z}) = x'\} \\ &= \rho(\theta', \theta, x) \cdot \mathbb{P}\{\underline{z} = \psi^{inv}(x' - x)\} \\ &= \rho(\theta', \theta, x) \cdot \mu(\psi^{inv}(x' - x)) \end{aligned}$$

where $\psi^{inv}$ is such that $\mu_L\{u \mid \psi^{inv}(u) \in B\} = \psi(\theta', \theta, x, B)$. Therefore,

$$\mu(\psi^{inv}(x' - x)) = \frac{Q(\{\theta'\}, x'; \theta, x)}{\rho(\theta', \theta, x)}$$

and $\psi$ is finally defined by

$$\psi^{inv}(x' - x)) = \mu^{inv}\left(\frac{Q(\{\theta'\}, x'; \theta, x)}{\rho(\theta', \theta, x)}\right)$$

**$\{W_t\}$:** This is generated according to the standard mechanism to generate Wiener processes. An $h$-dimensional Wiener process is constructed by collecting a number of $h = \sum_{i=1}^{|\mathcal{P}|} m_i^{max} h(P_i)$ independent one-dimensional Wiener processes in a vector.

**Adding zeros and transforming discrete state vectors** We add a sufficient number of zeros to some of the elements in order to create a constant dimension for the HSDE$^{SDCPN}$ hybrid state space. Denote $n = \max_\theta d(\theta)$, $0^a$ as a column vector of zeros in $\mathbb{R}^a$ and $0^{a \times b}$ as a matrix of zeros in $\mathbb{R}^{a \times b}$, then $E$ is redefined as $E = \{\{\theta\} \times (E_\theta \times \mathbb{R}^{n - d(\theta)}); \theta \in \mathbb{M}\}$; $f$ is redefined as $\mathrm{Col}\{f, 0^{n-d(\theta)}\}$; $g$ is redefined as $\mathrm{Col}\{g, 0^{(n-d(\theta)) \times \Sigma_i m_i^{max} \cdot h(P_i)}\}$; $X_0$ is redefined as $\mathrm{Col}\{X_0, 0^{n-d(\theta)}\}$ and $\psi$ is redefined as $\mathrm{Col}\{\psi, 0^{n-d(\theta)}\}$.

This shows that all HSDE$^{SDCPN}$ elements can be characterised uniquely in terms of SDCPN elements.

### B.2 Probabilistic equivalence

Subsequently, we show that the solution of the constructed HSDE$^{SDCPN}$ delivers a stochastic process which is probabilistically equivalent to the process defined by the SDCPN. This is done by showing: Equivalence of initial states; Equivalence of continuous evolution until first jump; Equivalence of time of jumps; Equivalence of size of jumps; Equivalence of processes after the first jump.

**Equivalence of initial states:**
The initial HSDE$^{SDCPN}$-process state $(\theta_0, X_0)$ at $t = \tau_0$ is equivalent to the initial SDCPN state through the mapping constructed above. If $\mathcal{I}^{inv}$ denotes the inverse of $\mathcal{I}$ and $\mu^{inv}_{\theta_0,X_0}$ denotes the inverse of $\mu_{\theta_0,X_0}$, then the random variable $(M_0, C_0) = \mathcal{I}^{inv}(U)$ is equivalent to the random variable $(\theta_0, X_0) = \mu^{inv}_{\theta_0,X_0}(U)$. Due to equivalence between $\mathcal{I}$ and $\mu_{\theta_0,X_0}$, the initial states are probabilistically equivalent.

**Equivalence of continuous evolution until first jump:**
At times $t$ when no jump occurs, the HSDE$^{SDCPN}$-process evolves according to $f$ and $g$, driven by a Wiener process, and the SDCPN-process evolves according to $\mathcal{V} = \{\mathcal{V}_P; P \in \mathcal{P}\}$ and $\mathcal{W} = \{\mathcal{W}_P; P \in \mathcal{P}\}$, driven by Brownian motion. Through the mappings between $f$ and $\mathcal{V}$ and between $g$ and $\mathcal{W}$ developed above, and through the probabilistic equivalence between Brownian motions and Wiener processes, these evolutions provide probabilistically equivalent processes $X_t$ and $C_t$ for all $t > \tau_0$, until the first jump.

**Equivalence of time of jumps:**
The times of jumps are generated by forced jumps and spontaneous jumps. In SDCPN, the forced jumps are represented by guard transitions; in HSDE, the forced jumps are represented by continuous state space boundary hits. Due to the mapping between the boundary of the HSDE$^{SDCPN}$ state space $\partial E_\theta$ and the boundaries of the transition guards of the guard transitions $\{\partial\mathcal{G}_T; T \in \mathcal{T}_G\}$, the HSDE$^{SDCPN}$ forced jumps and the SDCPN forced jumps occur at the same time. The HSDE$^{SDCPN}$ spontaneous jumps are generated by a Poisson random measure that uses a rate $\Lambda$. The SDCPN spontaneous jumps are generated by the delay transitions that use rates $\{\mathcal{D}_T; T \in \mathcal{T}_D\}$. Due to the mapping between $\Lambda$ and $\{\mathcal{D}_T; T \in \mathcal{T}_D\}$, the time of spontaneous jump is according to the same rate for both HSDE$^{SDCPN}$ and SDCPN.

**Equivalence of size of jumps:**
At times when a jump occurs, the HSDE$^{SDCPN}$-process makes a jump generated by $\psi$, $\rho$ and $\mu$, while the SDCPN-process makes a jump generated by $\mathcal{F}$. Through the mapping between $\psi$, $\rho$, $\mu$ and $\mathcal{F}$, these jumps provide probabilistically equivalent processes.

**Equivalence of processes after the first jump:**
After the first jump, equivalence is shown in a similar way as above.
This completes the proof of Theorem 5.4.

13

# Reachability Analysis of Time-Critical Systems

Štefan Hudák, Štefan Korečko and Slavomír Šimoňák
*Department of Computers and Informatics*
*Faculty of Electrical Engineering and Informatics*
*The Technical University of Košice*
*Slovak Republic*

## 1. Introduction

The systems whose functionalities (i.e. semantics) are defined with respect to time and whose correctness can only be assessed by taking time into consideration are called time - critical systems. This class of systems includes embedded real - time systems such as process control systems, digital signal processing systems, patient monitoring systems, flight control systems and weapon systems. The definition of time - critical systems covers a broader class then that of real - time systems. As an example can serve a typical computer system with a mouse input device where making two clicks on the mouse (the first and the second click at the beginning and at the end of the time interval duration 2 seconds respectively), has a quite different meaning than a double clicking within the time interval duration an half of second.
There is much expectation from the application of formal description techniques (FDT) and verification techniques to that field, since often time - critical systems have severe reliability and safety requirements. The theory and techniques developed for designing and understanding sequential and concurrent systems are not immediately suitable for time - critical systems: new formalisms and new methods are needed to specify their properties and prove correctness of implementations.
Several extensions to mathematical theories, such as logic, to cope with concurrent and real-time systems have been already proposed (Olderog, 1991; Ostroff, 1989), or algebra (Baeten & Bergstra, 1991). Petri Nets (PN) have deserved great attention in the role of FDT for concurrent, and in the last past as an FDT for *time - critical systems*. Two kinds of extensions to PN formalism are relevant with respect to time-critical systems:

- extensions that add time modelling capabilities (Time PNs),
- extensions that add functional modelling capabilities.

Different PN formalisms have been introduced in the literature; for example see (Genrich, 1986; Ghezzi et al., 1991; Jensen & Kristensen, 2009). Stochastic PNs represent important extension - that serves mainly to deal with performance evaluation and not in specification and verification of time-critical issues. Function modelling capabilities have been added to PNs at so-called high-level PNs. Among them we mention Colored Petri Nets (Jensen & Kristensen, 2009), Predicate Transition Nets (Genrich, 1986; Genrich & Lautenbach, 1981),

Numerical Petri Nets (Billington et al., 1988), PROT Nets (Bruno & Marchetto, 1986), Evaluative Petri Nets (Hudák, 1980) and TER Nets (Ghezzi et al., 1991).

## 2. Reachability Problem in Petri Nets

Petri Nets are very well known FDT, and many sources in the literature treat the subject in the depth (Peterson, 1981; Reisig, 1985; Murata, 1989; Hudák, 1999).
We define PN to be a 4-tuple $N = (P, T, pre, post)$, where:

- P is a finite set of places
- T is a finite set of transitions
- $pre : P \times T \to \{0,1\}$ - preset function, $post : P \times T \to \{0,1\}$ - postset function

The functions *pre, post* define a structure on the set $P \cup T$. It is very common to represent the PN by the bipartite oriented graph (Fig. 1a). The following useful notations can be defined:

| | |
|---|---|
| ${}^\bullet t = \{p \mid pre(p,t) \neq 0\}$ the set of preconditions of $t$ | ${}^\bullet p = \{t \mid post(p,t) \neq 0\}$ |
| $t^\bullet = \{p \mid post(p,t) \neq 0\}$ the set of postconditions of $t$ | $p^\bullet = \{t \mid pre(p,t) \neq 0\}$ |

By the marking of PN $N = (P, T, pre, post)$ we mean a totally defined function

$$m: P \to \mathbf{N} \tag{1}$$

We use **N** for the set of natural numbers , i.e, **N**={0,1,2,…} and $m$ to describe the situation or configuration in PN $N$. Namely we say the condition represented by the place $p$ in PN $N$ holds iff $m(p) \neq 0$. Without loss of generality we assume that $P$ and $T$ have $k$ and $s$ elements respectively, i.e. $P = \{p_1, p_2, ..., p_k\}$, $T = \{t_1, t_2, ..., t_s\}$ and we fix some ordering of both, places and transitions from now on. Using the ordering of places we can consider $m$ to be the $k$-dimensional nonnegative integer vector, i.e. $\vec{m} \in \mathbf{N}^k$. More formally

$$\vec{m} = (m(p_1), m(p_2), ..., m(p_k))$$

And $m(p_i)$ is the value of $m$ in $p_i$, $i$ = 1,2,..., $k$, according to (1). In our example (Fig. 1a) $m(p_i) = 1$ iff $i$ = 1, or alternatively $\vec{m} = (1,0,0,0,0)$. For the simplicity we will use the denotation $m$ for both interpretations of the marking $m$. We say $t$ is enabled in $m$, and denote it $m \xrightarrow{t}$ iff for every $p \in t$, $m(p) \geq pre(p,t)$. In Fig. 1a $t_2$ is enabled in $m = (1,0,0,0,0)$ because ${}^\bullet t_2 = \{p_1\}$ and $m(p_1)$ = 1, and $pre(p_1, t_2)$ = 1. Once the transition $t$ is enabled it can fire. The effect of the firing $t$ in $m$ is the creation of a new marking $m'$ that depends on $m$ and $t$. We use a denotation

$$m \xrightarrow{t} m'$$

and $m'$ is defined in the following way:

$$m'(p) = \begin{cases} m(p) - pre(p,t) & if \quad p \in {}^\bullet t \wedge p \notin t^\bullet \\ m(p) + post(p,t) & if \quad p \in t^\bullet \wedge p \notin {}^\bullet t \\ m(p) - pre(p,t) + post(p,t) & if \quad p \in {}^\bullet t \cap t^\bullet \\ m(p) & otherwise \end{cases}$$

In PN $N$ of Fig. 1a we can write $m = (1,0,0,0,0) \xrightarrow{t_2} m' = (0,1,1,0,1)$ . Notice that transition $t_3$ will be enabled in $m'$. We say the sequence of transitions $\sigma = t_1 t_2 ... t_r$ is *admissible firing sequence* in PN $N$, provided a sequence of markings $m_0, m_1, ..., m_r$ exists and such that $m_{i-1} \xrightarrow{t_i} m_i$ , $i$ = 1, 2,..., $r$. In that case we write $m_0 \xrightarrow{\sigma} m$ or $m_0 \xrightarrow{*} m$ , when σ is immaterial. The marking $m$ is to be called *the reachable marking* in $N$ from $m_0$ (via σ). We fix the marking $m_0$ to be *the initial marking* of PN $N = (P, T, pre, post)$ and we denote it $N_0$ = ($N$, $m_0$) or $N_0 = (P, T, pre, post, m_0)$ . Given PN $N_0 = (P, T, pre, post, m_0)$ we define the set of reachable markings

$$R(N_0) = \{m \mid m_0 \xrightarrow{\sigma} m\}$$

**Reachability Problem (RP).** Given PN $N_0 = (P, T, pre, post, m_0)$ and a $k$-dimensional non-negative integer vector $q$, the problem whether $q \in R(N_0)$ is called *(the instance of) the reachability problem* of PN $N_0$ (for the state $q$).
The reachability problem can be treated in the terms of so called *vector addition systems* (VAS).

**Definition 1***: A (k-dimensional) vector addition system (VAS) $W_k$ is a couple $W_k$ = ($q_0$ ,$W$), where $q_0 \in \mathbf{N}^k$ is the initial state of $W_k$ , $W$ is a finite set of (k- dimensional) integer vectors, i.e*

$$W = \{w_i \mid \forall i (1 \le i \le s) : w_i = (a_{i_1}, a_{i_2}, ..., a_{i_k}), a_{i_j} \in \mathrm{Z} \ \ for\ all\ j = 1,2,...,k\}$$

*By a reachable state (vector) of $W_k$ we call each $q \in \mathbf{N}^k$ such that*

1. $q = q_0 + w_{i1} + ... + w_{in}$ for some integer $n \ge 0$, $w_{ij} \in \mathrm{W}$, $j$ = 1,..., $n$ and
2. for $\forall j (1 \le j \le n)$: $q_j = q_0 + w_{i1} + ... + w_{ij} \in \mathbf{N}^k$

*We call the set of all such vectors the set of reachable state vectors, or simply the reachability set of VAS $W_k$, and denote it as R($W_k$ ). More formally:*

$$R(W_k) = \{q \mid q \in \mathrm{N}^k, q = q_0 + w_{i_1} + w_{i_2} + ... + w_{i_n}, n \in \mathrm{N},$$
$$\forall j (1 \le j \le n) : w_{i_j} \in W, q_j = q_0 + w_{i_1} + w_{i_2} + ... + w_{i_j} \in \mathrm{N}^k\}$$

□

In what follows we use the words: *reachable state vector, state vector*, or *state* as synonyms. If it is clear what dimensionality $W_k$ has, we omit the index $k$, and also we say just a vector addition system instead a $k$-dimensional VAS, and the vector $w$, rather than the $k$-dimensional vector $w$.

**Definition 2:** *Given any VAS $W_k$ = ($q_0$,W), where $q_0 \in \mathbf{N^k}$ and $W \subseteq Z^k$ for some k > 0, and let $R(W_k)$ be its reachability set. For any $q \in \mathbf{N^k}$ a problem whether $q \in R(W_k)$ is called the reachability problem of VAS (with respect to q). We will occasionally use the abbreviation $RP(q,W_k)$ ) for it.* □

With a vector addition system $W_k = (q_0, W)$ we can associate a tree structure, which we call the vector state tree, and define it as follows:

**Definition 3:** *Let $W_k = (q_0, W)$ be a vector addition system with $q_0$ as its initial state vector, and W the finite set of k-dimensional integer vectors. Then by the vector state tree of VAS $W_k$, denoted by $VST_w$, we mean a double labelled oriented tree*

$$VST_w = (T_w, lab_1, lab_2, q_0),\ T_w = (V, E, r_0)$$

*is an oriented rooted tree, V - a set of vertices, $E \subseteq V \times V$ - a set of edges, $r_0 \in V$ - the root of $T_w$ and the two labelling mappings*

$$lab_1 : V \to \mathrm{N}^k,\ lab_2 : E \to W$$

*The VST is defined such that $lab_1(r_0) = q_0$ and any vertex v of $T_w$, with $lab_1(v) = q$ has a son $u \in V$ with $lab_1(u) = q'$ and $lab_2(v,u) = a$ iff $q' = q + a$ for some $a \in W$ and $q,q' \in \mathbf{N^k}$.* □

Let $N_0 = (P, T, pre, post, m_0)$ be a PN with the initial marking $m_0$. Recall $m_0$ can be represented as a *k*-dimensional nonnegative integer vector, i.e. $m_0 \in \mathbf{N^k}$ and $m_0 = (m_0(p_1),..., m_0(p_k))$. The latter assumes an ordering of places in *P* and transitions in *T*, i.e. $P = \{p_1,..., p_k\}$ and $T = \{t_1,..., t_s\}$. From the analytical properties of PNs (Reisig, 1985) we have that

$$m_0 \xrightarrow{t} m \Leftrightarrow \vec{m} = \vec{m}_0 + (\Psi(t) \times c^T)^T$$

and $\Psi(t)$ is the Parikh mapping vector of the "string" *t* over the (ordered) alphabet *T*. Recall again that any transition $t \in T$ can be represented as a *k*-dimensional integer vector

$$\vec{t} = \overrightarrow{post}(t) - \overrightarrow{pre}(t)$$

and

$$\overrightarrow{post}(t) = (post(p_1,t), post(p_2,t),..., post(p_k,t)),\ \overrightarrow{pre}(t) = (pre(p_1,t), pre(p_2,t),..., pre(p_k,t))$$

It can be easily seen that

$$m_0 \xrightarrow{t} m \Leftrightarrow \vec{m} = \vec{m}_0 + \vec{t}$$

and we can construct for PN $N_0 = (P, T, pre, post, m_0)$ the vector addition system $W_k = (q_0, W)$ and such that $q_0 = \vec{m}_0$, $W = \{\vec{t}_i \mid t_i \in T, i = 1,2,..., s\}$, and *k* = *card P*. The following result holds.

**Theorem 1.** *For any PN $N_0 = (P, T, pre, post, m_0)$ there is an vector addition system $W_k = (q_0, W)$ and such that $R(W_k) = R(N_0)$, and k = card P.* □

**Proof:** follows from the above construction.

In what follows we make use of the special symbol - $\omega$, which we add to **N**, creating by that virtue the new set $\mathbf{N}_\omega$. The ω element has the following properties: for any $a \in \mathbf{N}$: $a \leq \omega$ and

$\omega \pm a = \omega$. Having a ($k$-dimensional) vector $q = (q_1, q_2, ..., q_k)$ and a set $A \subseteq \{1,2,...,k\} = \mathrm{K}$ we denote by $\omega_A q$ a vector, that we will call $\omega$ *vector* and will be defined such that

$$\omega_A q = (a_1, a_2, ..., a_k) \text{ where } a_i = \begin{cases} \omega & \text{if } i \in A \\ q_i & \text{otherwise} \end{cases}$$

In (Hudák, 1999) a new RP algorithm was proposed, in which a core structure is a finite state automaton (fsa) of the type $M_w$. Here we can only in a very short way summarize the RP algorithm.

**RP algorithm:** Given: VAS $W_k = (q_0, W)$, $q \in \mathbf{N^k}$ - a state to be decided reachable or not;

Step 1 : Create fsa $M_w$=(Q, W, δ, $\rho_0$) ;
Step 2 : Construct $\mathrm{MILP_w}$ (A, $X_0$, B($q$), r);
Step 3 : if $\mathrm{MILP_w}$ (A, $X_0$, B(q), r) = true then go to Step 4 else go to Step 5;
Step 4 : $q \in \mathrm{R}(W_k)$. Stop.
Step 5 : $q \notin \mathrm{R}(W_k)$. Stop.

Here, $Q \subseteq \mathbf{N^k}$ is the set of states, δ: $Q \times W \to Q$ is the next-state function, $\rho_0 = \omega_A q_0$, $A \subseteq \boldsymbol{K}$ is the initial ω (macro) state. In many cases $M_w$'s state diagram contains strongly connected components (scc), i.e it has the loop structure (see Fig. 8). $\mathrm{MILP_w}$ (A, $X_0$, B($q$), r) stands for *the modified integer linear programming problem* (Hudák, 1999), and denotes a predicate that is true iff $X_0$ is the solution to the problem, $A$ is the matrix of ρ-*simple loops* (Hudák, 1999) of fsa $M_w$ (taken as integer vectors), $r$ is the number of strongly connected components of $M_w$'s state diagram on the path leading from the macrostate $\omega_B q_0$ to a macrostate $\omega_{B'} q$, provided $q$ is the state whose reachability is solved by the instance of MILP, and $B(q)$ is some linear expression free of any ρ-*simple loops* and depending on the state $q$.

The complexity of the RP algorithm was assessed (Hudák, 1999) and the worst-case upper bound of time complexity was settled. It turns out to be $O(2^{b^{2^{k+2} \times k^2}})$, provided $k$=*card* $P$, $P$ to be the set of places of PN $N_0 = (P, T, pre, post, m_0)$ and $b \geq 0$ is some constant. Fortunately, in some cases the worst-case upper bound time complexity can be lowered (Hudák, 1999). There the reader can find also the proof of lower bound on the space complexity of RP in spirit of R.J.Lipton, which had been shown to be $O(c^k)$ for some constant $c$>0.

Fsa $M_w$ is a finite state automaton with some interpretation of its states and input alphabet. That means that fsa can be thought of as $M_w$ = (M,$I$) where M = (K, Σ, δ, $r_0$) we call the *basic fsa* for $M_w$, and $I$ = (f, g, $r_0$) - be an *interpretation*:

$f : K \to \mathrm{N}_\omega^k$ - is a state labelling mapping

$g : \Sigma \to \mathrm{Z}^k$ - is an input labelling mapping

Properties of the interpretations were studied, that was inspired by V. N. Redko (Hudák, 1999).

***Problem:*** *Let M = (K,Σ, δ, $r_0$) be a fsa; is there any interpretation I = (f,g,$r_0$) for <M,I> to be the finite state automaton of the type $M_w$ for the PN $N_0 = (P, T, pre, post, m_0)$ ?*

*for <M,I> to be the finite state automaton of the type $M_w$ for the PN $N_0$ = (P,T, pre , post,$m_0$)?*

Here follow results of the study and solution to the problem settled and also some remarks, that reflect some interesting issues connected with the problem (Hudák, 1999).

1. Assume we are given a fsa $M$ =($K$, Σ, δ, $q_0$); for any such $M$ and a constant $k$ > 1 there is a tripple of mappings $I_k$ = <$f$, $g$, $r_0$>, and such that it is the interpretation for the fsa $M$, and <$M$, $I_k$> is $M_w$-type fsa for the VAS $W_k$ = <$r_0$, {$g(a)$ | $a \in \Sigma$}>. $I_k$ and VAS $W_k$ can be effectively constructed on the basis of $M$ and of the constant $k$.

2. Assume we are given a fsa of the type $M_w$ - $M' = (Q, W, \delta_w, r_0)$, $Q \subseteq \mathbf{N}^k$, $W \subseteq \mathbf{Z}^k$, $r_0 \in \mathbf{N}^k$; then on the basis of $M'$ the fsa $M = (K, \Sigma, \delta, q_0)$ and the couple of mappings $<f, g>$ can be constructed effectively; the latter is the isomorphism of fsa's $M$ and $M'$, and $<f, g, r_0>$ represents the interpretation for the fsa $M$.

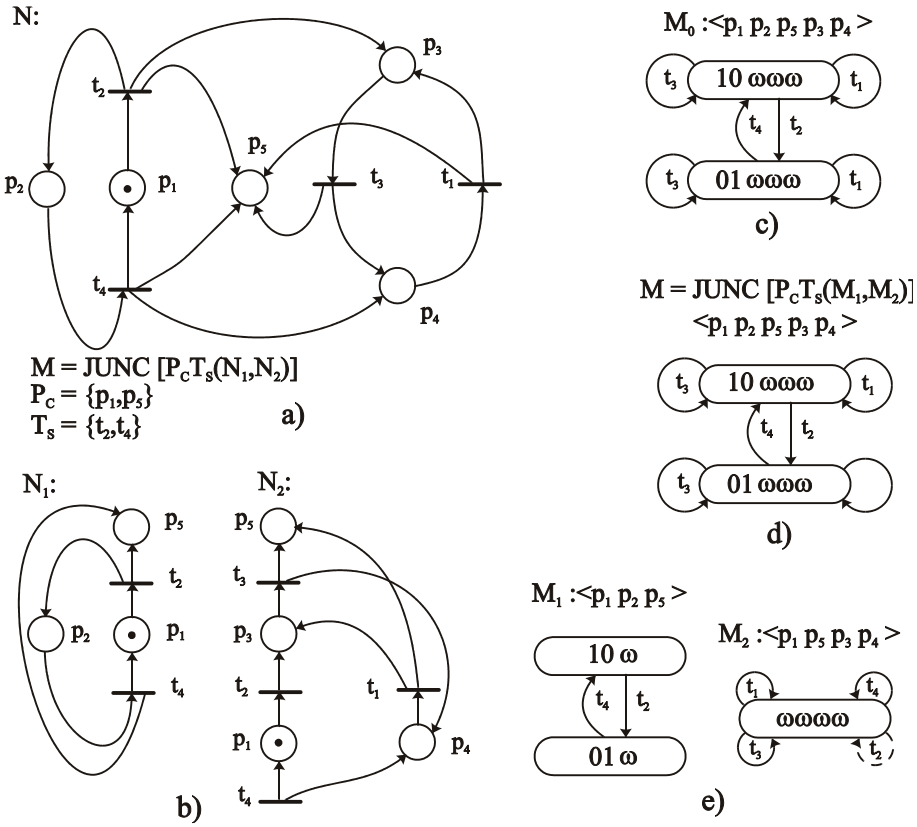



Fig. 1. De/compositional reachability analysis

### 2.1 De/compositional reachability analysis of PNs

We have pointed out that the complexity of RP is tremendous and so RP is an intractable problem. On the other hand there are some (quite practical and vital) problems for to be dealt with and for that a solving the reachability problem is an inevitable task. An example of such a problem can serve time analysis of time-critical systems (Ghezzi et al., 1994). An outcome from that situation, and that seems to be the only way, is to apply the divide and conquer approach to the RP. In (Hudák, 1994) we have proposed a new de/composition method of PNs that is suitable for the reachability analysis of Petri Nets in general case. The method is based on the RP algorithm by the first author (Hudák, 1999).

The de/composition can be accomplished in three ways: T-(P-, PT-) de/compositions, that we call T-(P-, PT-) JUNCTION respectively (Hudák, 1994).

Fig. 1 illustrates the method and results achieved. In Fig. 1a) and 1b) there are PN $N$ and the subnets $N_1$ and $N_2$, the product of PT decomposition. Fig. 1c) and e) show the fsa of type $M_w$ for the PNs $N$ and $N_1$, $N_2$ respectively. They are fsa's with ω states that says the state space is infinite. Fig. 1d) shows the result of composition based on $M_1$ and $M_2$. Comparison of the state diagrams in Fig.1c) and d) shows the two automata are isomorphic.

## 3. Time-Critical Systems

A large class of systems can be represented and understood by abstracting away from the time aspect. That is the case of sequential systems, i.e. systems whose provided functionalities - their semantics - do not depend on the speed of execution. It is also the case of properly designed concurrent systems, such as time - sharing operating systems, whose overall correctness does not depend on the speed of execution of the component processes. In these two classes of systems time affects performance, not functional correctness.
In other cases, however, time issue becomes essential. In real - time systems, correctness depends not only on the results produced by computations, but also on the time at which such results are produced. The systems may enter incorrect state if the right result is produced too early or too late with respect to certain time bounds (Ghezzi et al., 1991). For example, an aircraft should not only modify its course once a mountain appears to be on the route, but it should also do it before crashing, i.e. within a given time.
The systems whose functionalities (i.e. semantics) are defined with respect to time and whose correctness can only be assessed by taking time into consideration are called time - critical systems. This class of systems includes embedded real - time systems such as process control systems, digital signal processing systems, patient monitoring systems, flight control systems and weapon systems. The definition of time - critical systems covers a broader class then that of real - time systems. As an example can serve a typical computer system with a mouse input device where making two clicks on the mouse (the first and the second click at the beginning and at the end of the time interval duration 2 seconds respectively), has a quite different meaning than a double clicking within the time interval duration an half of second.

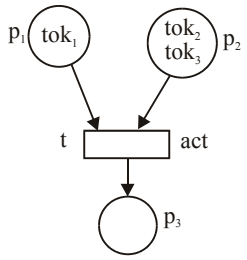
### 3.1 Environment Relationship Nets

Environment-Relationship Nets (ER nets) (Ghezzi et al., 1991) represent a very strong extension to ordinary Petri Nets, that provides means to incorporate the notion of the time into the concept. We start with some notions. ER net is a net which can be characterized as follows:

1. tokens are environments on ID (the set of identifiers) and V (the set of all values identifiers can take upon). The universal set of environments ENV = $V^{ID}$.
2. each transition is associated with an action. An action is a relationship
$$\alpha(t) \subseteq ENV^{k(t)} \times ENV^{h(t)}, \text{ and } k(t) = |{}^\bullet t|, h(t) = |t^\bullet|$$
By $\pi(t)$ we denote the projection of $\alpha(t)$ on $ENV^{k(t)}$ only, and call it *the predicate of t.*
3. a *marking* is an assignment of multisets of environments *(envs)* to places.
4. transition $t$ is enabled in a marking $m$ iff $\forall p_i \in {}^\bullet t$ there is at least one token $env_i$ and such that the tuple $<env_1, ..., env_{k(t)}> \in \pi(t)$. The tuple $<env_1, ..., env_{k(t)}>$ is called *the*

*enabling tuple for t*. Notice there can be more than one enabling tuple for $t$ in the marking $m$. The same token can belong to several enabling tuples.

5. *A firing* (of $t$ in $m$) is a triple $x =< enab, t, prod >$ such that $< enab, prod > \in \pi(t)$.
6. The occurrence of a firing $x =< enab, t, prod >$ in a marking $m$ causes a producing a new marking $m'$ for the net, obtained from the marking $m$ by removing the enabling tuple *enab* from the preset places of $t$.
7. in a natural way we define a *firing sequence* and *the sequence of admissible markings*. We may use the notation $m \xrightarrow{x} m'$ provided that $x =< enab, t, prod >$ is a firing that produces $m'$ from $m$. We may also define the set of reachable markings, boundedness, liveness and other notions that can be defined for ordinary Petri Nets.




$\mathbf{tok_1}$ = {<a,0>,<b,1>}

$\mathbf{tok_2}$ = {<a,0>,<b,2>}

$\mathbf{tok_3}$ = {<a,0>,<b,1>}

$\mathbf{act} = \{<<p_1,p_2>,p_3> |\ p_1.a = p_2.a,\ p_1.b < p_2.b,\ p_3.a = p_1.a,\ p_3.b \in \{x |\ p_1.b \le x \le p_1.b + p_2.b\}\}$

Fig. 2. ER Net

### 3.2 Time ER Nets

When we assume that each environment contains a special variable called *chronos*, whose values are of numerical type representing *the timestamp* of the environment (token), by that the time can be incorporated into the net and we obtain Time ER net. All notions introduced for ER nets remain valid, for TER net, except the firing rule; the latter has to be modified. The timestamp of the token expresses the time of the environment's creation. We denote by $env_i.chronos$ the value of the chronos of the token $env_i$. We assume the firing $x$ to be used in what follows. The firing $x$ has the structure: $x =< enab, t, prod >$, $enab =< env_1, env_2, ..., env_{k(t)} >$ $prod =< env'_1, env'_2, ..., env'_{h(t)} >$, $k(t) = |{}^\bullet t|, h(t) = |t^\bullet|$. The following are the axioms that have to be satisfied by any action (Ghezzi et al., 1991).

**Axiom 1:** *(Local monotonicity) For any firing x* $env'_j.chronos \ge env_i.chronos$, *for all* $j,i,\ 1 \le j \le h(t),\ 1 \le i \le k(t)$.

**Axiom 2:** *(Constraint on timestamps) For any firing x there is the value denoted time(x) and such that* $env'_j.chronos = time(x)$, *for all* $1 \le j \le h(t)$. *The value time(x) is called the time of the firing.*

**Axiom 3***: (Firing sequence monotonicity) For any firing sequence* $s = x_1 x_2 ... x_{|s|}$, $time(x_i) \le time(x_j)$, *if* $i < j,\ 1 \le i,j \le |s|$.

For an ER net satisfying Axiom 2 we will write *prod.chronos* to denote the value *time(x)* the all chronos have in *prod*. An ER net where all environments contain chronos and that satisfies Axioms 1 and 2 is called *Time ER (TER) net*. Given ER net that satisfies Axiom 2, we will call the firing sequence $s = x_1 x_2 ... x_{|s|}$, to be *the time ordered firing sequence* iff for each $i,j,$

$i < j$, $time(x_i) \leq time(x_j)$. We will call two firing sequences $s$, $s'$ to be equivalent iff $s$ is a permutation of $s'$. The following result can be proven (Ghezzi et al., 1991)

**Theorem 2.** *Let E be an ER net satisfying Axioms 1 and 2; for each firing sequence s with the initial marking $m_0$ there exists a firing sequence s' equivalent to s that is time ordered.*

Given a transition $t$ and the enabling tuple *enab* of the TER net, we can define *the set of possible firing times*, denoted *f-time*:

$$f-time(enab) = \{t \mid < enab, prod > \in \alpha(t), t = prod.chronos\}$$

Let $s = x_1 x_2 ... x_{|s|}$, be a firing sequence of a TER net with the initial marking $m_0$, and $m_0 \xrightarrow{s} m$ and $m_0 \xrightarrow{x_1 x_2 ... x_i} m_i$ ; the firing sequence $s$ is *strong* iff it is time ordered, and for each $t' \in T$ and for each $m_i, (1 \leq i \leq |s|-1)$ there exists no tuple $enab_i^{'}$ for $t'$ in $m_i$ and such that $time(x_{i+1}) > \sup(f-time(enab_i^{'}))$ . Another axiom can be stated to hold in TER nets.

**Axiom 3':** *All firing sequences are strong.*

TER net that satisfies Axioms 1,2 and 3' is called *the strong* TER net (STER net).

Finally, when we assume that the only type of tokens is chronos then we obtain *Time Basic (TB) nets*. In the class of TB nets we can distinguish two significant classes:

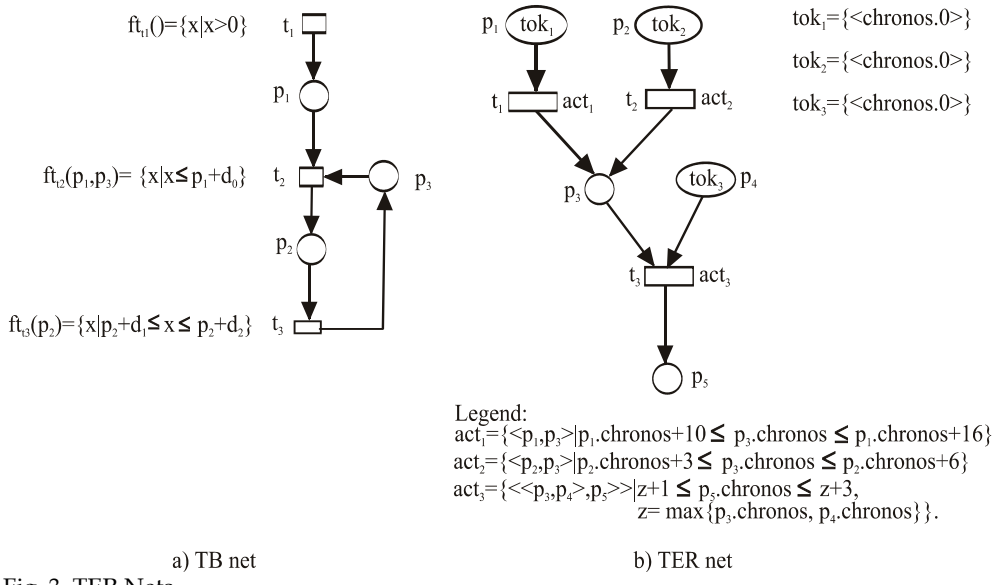
1. *weak-time semantics* WTS TB nets are those TB nets that satisfy Axiom 1,2, and 3,
2. *strong-time semantics* STS TB nets are those TB nets that satisfy Axiom 1,2 and 3'.

Fig. 3 shows an example of a TB net and a TER net. Another example of a TB net can be found in Fig. 7, where the net specifies a time-critical system, called 'the voice station', whose operation is governed by a simple protocol. The latter is formally described altogether by the $tf_t$ predicates and by the structure of the TB net itself.

The modelling power of TER nets is such that it covers all known extensions of Petri Nets including those which incorporate the concept of the time (e.g. MF nets)(Ghezzi et al., 1991). The tradeoff between modelling power and the tractability of problems that of our concern should be taken into account. The greater is the modelling power, less is a chance to solve interesting problems: reachability, liveness, boundedness and others.

## 4. Time Reachability Analysis of TB Nets

There was pointed out that time-critical systems represent an important class of discrete systems for which the reachability analysis with respect to time is an inevitable task to be done during design and analysis of such the systems. We call the problem in that case *time reachability problem* or *time reachability analysis* (TRA) *problem*. In practice (Ghezzi et al., 1994), to cope with the problem mostly simulation techniques are used.




a) TB net b) TER net

Fig. 3. TER Nets

We have started the development of a new methodology of the reachability analysis, based on the new algorithm to solve the reachability problem by the first author (Hudák, 1981; Hudák, 1999), and the de/compositional approach to the problem (Hudák, 1994). Because of the specification of time critical systems by TB nets and that the task of TRA is inevitable in the design of such the systems, the new methodology presupposes an extension of the original results on de/compositional reachability analysis to TB nets.

The new methodology of the time reachability analysis consists of following steps.

1. developing a new semantics of TB nets based on time intervals (TI) rather than on time points (Ghezzi et al., 1994; Hudák, 1996),
2. extending new RP algorithm to TB (TER) nets in a way to be able to use the properties of fsa's of the type $M_w$,
3. exploring properties of Time Interval Profiles (TIP) as a suitable generalization of enabling tuples in TI semantics,
4. creating a new methodology for TRA of time-critical systems specified by TB nets that would have to be based on "the loop-spectral analysis" of TB nets (Hudák & Teliopoulos, 1998b).

### 4.1 Time Interval Semantics of TB Nets

In this section we deal with the time interval semantics of TB nets. We want to extend the solution of the reachability problem (RP) (for Petri Nets) to Time Environment Relationships Nets (TER), particularly to TB nets. Basic property of TB nets is that their tokens (timestamps) have the individuality (Ghezzi et al., 1991).

For the purpose to deal with the Time Reachability Problem (TRP) we will make use of several interpretations of tokens (Fig. 4)

- the case a) is the usual interpretation of the token as being the chronos (Ghezzi et al., 1991),
- the cases b) and c) will be used when we deal with the reachability issue itself, especially in a constructing of the finite state automaton (fsa) of type $M_w$ (Hudák, 1996) for TRP, which we will denote as $M_{\tau w}$.

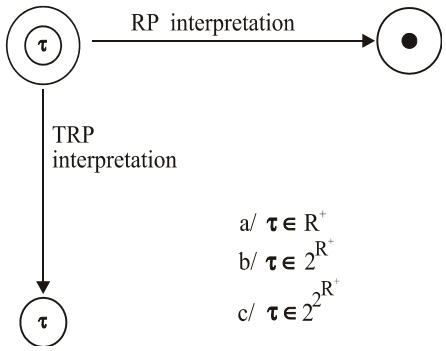



Fig. 4. Token Interpretation

In (Hudák, 1996) we deal with TB's semantics based on TIs. Any token (chronos) τ in TI semantics is considered to be a TI $\tau = [\tau i, \tau a] \subseteq \mathbf{R}^+$. Besides the set operations $\cap$, $\cup$, $()^C$ a new operations "+" and "·" have been defined (Hudák, 1996). Given $\tau = [\tau i, \tau a] \subseteq R^+$ we define the operations:

1. $0 + \tau = \tau + 0 = \tau$ where 0 stands for empty TI.
2. $c + \tau = [\tau i + c, \tau a + c]$, $c \cdot \tau = [\tau i \cdot c, \tau a \cdot c]$ for any suitable constant $c \in \mathbf{R}$
3. $\tau' + \tau'' = \tau \Leftrightarrow_{df} \tau = \{\theta \in R^+ \mid \theta = \theta' + \theta'', \theta' \in \tau', \theta'' \in \tau''\}$
4. following are true statements: $\tau' + \tau'' = \tau'' + \tau'$, $(\theta + \tau) + \tau' = \theta + (\tau + \tau')$

In TI semantics we introduce the concept of TI relations that can hold between TIs (Fig. 5). Given any transition $t \in T$ of a TB net with the enabling predicate $tf_t$ (Ghezzi et al., 1994), in TI semantics we replace any enabling tuple $en = (m(p_1), m(p_2), ..., m(p_{|\bullet t|}))$ with a corresponding collection of TIs that is called Time Interval Profile (TIP). In (Hudák, 1996) properties of TI operations were studied. For a reason that will be clear soon, a crucial issue to be dealt with was the property of TI expression

$$(\tau_1 + A) \cap (\tau_2 + B) \tag{2}$$

The study yielded several important results. Given $en = TIP(\tau_1, \tau_2, ..., \tau_{|\bullet t|})$ and $tf_t$-enabling predicate, we were able to prove the canonical representation exists for any *en* and $tf_t$ (*en*).

**Theorem 3.** *(canonical representation of $tf_t$) Given TB net $\tau N_0 = (P, T, \theta, pre, post, tf, q_0)$ and let $tf_t$ (en) be in the form $(\tau_1 + A) \cap (\tau_2 + B)$, then $tf_t$ (en) has for given en the unique representation*

$$tf_t(en) = \tau en + tf_t(0)$$

*where τen is a TI that depends on en , and $tf_t(0)$ is a TI that depends only on the structure of TB net and does not depend on en.* □

To put it in another way, any *t*-generated TI $\tau_t$ can be represented as a sum of two TIs: τ*en*-the determinate TI that depends on TIP *en* in question and on *t* (or $tf_t$) and a constant TI $tf_t(0)$, which depends only on the structure of the TB net in question.

The 'anatomy' of τ*en* was studied in (Hudák & Teliopoulos, 1998b). There was discovered that any τ*en* can be classified as belonging exclusively to one of the two possible classes: mono-generated τ*en*s (i.e. $\tau en = [\tau_t i, \tau_t a]$ and $\tau_t \in en$) and non mono-generated τ*en*s (i.e. $\tau en = [\tau_{t_1} i, \tau_{t_2} a]$ and $\tau_{t_1}, \tau_{t_2} \in en$ ).

| i | $Rel_i$ representation | | Formal characterization |
|---|---|---|---|
| | Graphical | Symbolic | |
| 1 | { [ } ] | $\tau <_{sq} \tau'$ | $(\tau i < \tau' i) \wedge (\tau' i < \tau a < \tau' a)$ |
| 2 | [ ] { } | $\tau' \sqsubseteq_a \tau$ | $(\tau i < \tau' i) \wedge (\tau a = \tau' a)$ |
| 3 | { [ ] } | $\tau' \sqsubset \tau$ | $(\tau i < \tau' i) \wedge (\tau' a < \tau a)$ |
| 4 | [ ] { } | $\tau \sqsubseteq_i \tau'$ | $(\tau i = \tau' i) \wedge (\tau a < \tau' a)$ |
| 5 | [ ] { } | $\tau' \sqsubseteq_i \tau$ | $(\tau i = \tau' i) \wedge (\tau' a < \tau a)$ |
| 6 | [ { } ] | $\tau \sqsubset \tau'$ | $(\tau' i < \tau i) \wedge (\tau a < \tau' a)$ |
| 7 | [ ] { } | $\tau \sqsubseteq_a \tau'$ | $(\tau' i < \tau i) \wedge (\tau a = \tau' a)$ |
| 8 | [ { ] } | $\tau' <_{sq} \tau$ | $(\tau' i < \tau i) \wedge (\tau' a < \tau a)$ |
| 9 | { } [ ] | $\tau \leq \tau'$ | $(\tau a < \tau' i)$ |

{ } $\tau = \{\tau i, \tau a\}$ interval [ ] $\tau' = \{\tau' i, \tau' a\}$ interval

Fig. 5. Relations defined on TI variables

Given TB net $\tau N_0 = (P, T, \theta, pre, post, tf, q_0)$ we call Petri net $N_0 = (P, T, pre, post, m_0)$ *basic* PN for TB net $\tau N_0$. In extending the RP algorithm to TB nets we have chosen an approach to create the fsa $M_w$ (in the case of TB nets we denote it $M_{\tau w}$), in such a way that $M_{\tau w}$ will be isomorphic with the corresponding fsa $M_w$ for the basic PN in question. As we already mentioned, fsa's $M_{\tau w}$ and $M_w$ will contain scc's. That implies that TIP's properties on ρ-loops are of very importance. The properties were studied in (Hudák & Teliopoulos, 1998b) and can be characterized as follows.

There are few assumptions taken for loops:

($a_1$) the initial marking $\mu_0 = (\rho_0, \tau_{\rho_0})$ to be such that $\rho_0(p) \leq 1$,

($a_2$) $post(p,t) \leq 1$ for any *p*, *t* of $\tau N_0$, i.e. upon *t*-firing exactly one TB token can be generated,

($a_3$) $\mu_0$ is a live marking,

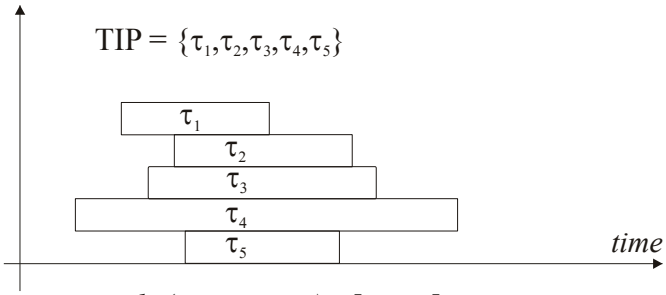
($a_4$) the existence of $en_t^{(0)}$ - $t$'s enabling tuple upon its first firing, as the first transition of $\sigma_t$, ($a_5$) in the loop σ, the only generators of TI tokens in any $p \in {}^{\bullet}t_i$, where $t_i$ is from $\sigma_t$ (i.e. $t_i \in T(\sigma_t)$), are again the transitions of $\sigma_t$.

**Theorem 4.** *(characterization of $tf_t(en)$)*

*Let $\sigma_t = t_1 \cdots t_r$ be a $\rho$ - simple loop satisfying the assumptions $a_1$-$a_5$ For any $i \geq 0$, $\forall \tau \in T(\sigma_t)$*

$$tf_t(en^{(i+1)}) = tf_t(en^{(i)}) + tf_t(0)$$
$$tf_\ell(0) = tf_{t_1}(0) + \ldots + tf_{t_r}(0)$$

*provided $T(\sigma_t) = \{t_i \mid \sigma_t(t_i) \geq 1\}$ and $\sigma_t(t_i)$ is the number of occurrences of $t_i$ in $\sigma_t$.*




$\tau = dti(\tau_1,\tau_2,\tau_3,\tau_4,\tau_5) = [ei, ea]$
$ei = \tau_5 i$, $ea = \tau_4 a$
$\tau = [\tau_5 i, \tau_4 a]$

Fig. 6. TIP and determinate time interval

**Corollary 1.** *For any $n>0$, $t \in \sigma_t = t\, t_1 \cdots t_r$*

$$tf_t(en^{(n)}) = tf_t(en^{(0)}) + n \cdot tf_\ell(0),$$
$$tf_t(en^{(n)}) = \tau en^{(0)} + n \cdot tf_\ell(0) \quad \text{and}$$
$$tf_t(en^{(\omega)}) = \tau en^{(0)} + \omega \cdot tf_\ell(0)$$

**Lemma 1.** *Let $p_1,p_2,\ldots,p_n$ be a sequence of places of TB net $\tau N = (P,T,\theta,pre,post,m_0,tf)$, $t_1,t_2,\ldots,t_{n-1}$ be a sequence of transitions from T and such that $p_i \in {}^{\bullet}t_i, p_{i+1} \in t_i^{\bullet}$ $\sigma$, $q$, $q'$- be a firing sequence and two markings respectively and such that $\sigma = x_1 x_2 \ldots x_n$, $q \xrightarrow{\sigma_t} q'$, $x_i = < en_i, t_i, prod_i >$, $i = 1,2,\ldots,n$. Let further $\tau_i$ be a chronos with the time interval (TI) interpretation (we propose to call it TI token, TI chronos, or simply TI) in $p_i$ in the state $q$ and such that $\tau_i = en(p_i)$ ($\tau_i$ is dwelling in $p_i$ in the state $q$) and $\tau_{t_i}$ be a $t_i$ - generated TI upon $t_i$ firing, $i=1,\ldots,n-1$ and $en_j(p_{j+1}) = \tau_{t_j}$, $j=1,\ldots,n-1$.*

*Then it is true that $\tau_1 \prec \tau_{t_1}, \tau_{t_{j-1}} \prec \tau_{t_j}$, $\quad i = 1,2,\ldots,n-1 \prec \in \{<_{sq}, \leq\}$ and $\tau_{t_j} \in \tau_{q_j}(p_{j+1})$*

*Provided that*

$q_j = [q\sigma_j]$, $\sigma_j = x_1 x_2 \ldots x_j$, *and* $[q\sigma_j] = \vec{q} + \overrightarrow{t(x_1)} + \ldots + \overrightarrow{t(x_j)t(x_j)} = \vec{t_j} \leftrightarrow x_j = < en, t_j, prod >$

**Corollary 2.** *Under conditions of Lemma 1 the following holds: for all i>0:* $\tau_t^{(i)} \prec \tau_{t_r}^{(i)}$.

**Lemma 2.** *Given a* $\rho_0$ *-loop in* $M_{\pi w}$ *for the TB net* $\pi N_0 = (N_0, tf, \theta)$, $\sigma_t = t_1 \cdots t_r$, $t_i \in T$, *i=1,...,r, and two TI tokens* $\tau_1', \tau_2'$ *belonging to* $en_t^{(0)} = TIP(\tau_1, \tau_2, ...\tau_t)$, *and* $\tau_j'$ *is* $t'_j$*-generated, TI chronos, i.e.* $\tau_j' = \tau_{t_j'}$ $t_j' \in T(\sigma_t)$, *j=1,2,…,r.*

*Suppose now that* $\tau_1' \prec \tau_2'$ *then* $\tau_{t_1'}^{(i+1)} \prec \tau_{t_2'}^{(i+1)}$ *for any i > 0 and* $\prec \in \{<_{sq}, \geq\}$.

What the results in the above assertions say is that:

- given the loop structure of particular scc in $M_w$ we are able to compute and thus predict the TIs which will be created upon passing the (stabile and selffeeded) loops (Theorem 4 and Corollary 1).
- relation $\prec \in \{<_{sq}, \leq\}$ between TIs will be preserved on the (stabile and selffeeded) loops of any scc of the fsa $M_w$ (Lemma 1, Lemma 2, and Corollary 2).

### 4.2 On the Way to Loop-Spectral Time Reachability Analysis

In the state diagram of fsa $M_w$ $\rho$ - simple loops play profound role in the reachability analysis of the ordinary PN as the results of this work demonstrate. The role of $\rho$ - simple loops grows even more in the issue of TRA of TB nets.

We distinguish two subclasses of loops in $M_{\pi w}$: *selffeeded* and *stabile* loops. The loop is selffeeded one if in a *t* - firing (*t* belongs to the loop) *t* "consumes" only tokens that was created solely by firings of loop's transitions. A loop can be called stabile if at any *t*-firing (*t* belongs to the loop) all tokens at precondition places are uniformly generated, i.e. at any *t*-firing at each repetition *t* consumes tokens from the same generators, i.e. transitions that generated tokens consumed by *t*. There is a strong relation between the two types of loops (Hudák & Teliopoulos, 1998b). Each loop becomes stabile after some initialization, after that some TIP (we call it initial) is reached which starts stable part of computation.

After defining initial TIP for any loop on a path we can define the TIs of any chronos (tokens) in which it can exist in the future. The structure of those TIs reminds very much spectral image of time sequences (Hudák & Teliopoulos, 1998b). Results achieved, however show that once $M_{\pi w}$ has been constructed we can predict precisely future of any token and discover perhaps a moment when it disappears because of the emptiness or dummy feature of its TI. For any TI $\tau_t$ (*t* - generated TI) we can construct a formula for the TI to be calculated. The RP algorithm works almost in the same way in the case of TB nets as it does in the case of ordinary Petri Nets.

## 5. TRA by example

We are now going to demonstrate TRA based on our approach. As an example we have chosen the TB net for the voice station (Ghezzi et al., 1994) which is depicted in Fig. 7. An interested reader can find a full description of the voice station in (Ghezzi et al., 1994).

### 5.1 The voice station

The voice station consists of three parts: coder, decoder and medium to transmit voice signals formatted into packets. Here we abstract from the structure of the packets, the nature of the medium, and the decoder. We only concentrate on the signal transmission, i.e. the voice coder and the part of the interface that transmits the packets.

The voice coder produces packets of constant length at a constant rate -one packet each 10 milliseconds. Once a new packet is produced and the station interface is ready to accept a new packet, the packet becomes ready to be transmitted. The reaction time of the interface, which takes ready packet and puts it in the transmission buffer, waiting to be transmitted on the medium, is assumed to be 0.2 milliseconds. The station interface waits for a transmission token (i.e. the interface is based on a token transmission protocol, e.g. a token ring) which is available for transmitting the packet already prepared in the buffer, 0.1 milliseconds after its arrival in the interface.

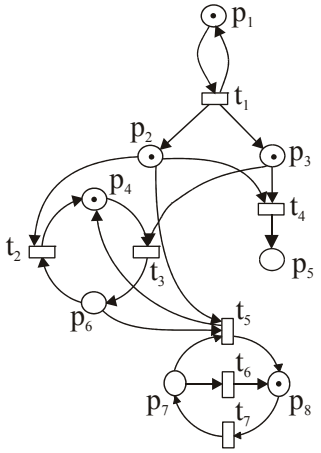
A packet is transmitted if it is in the buffer and the transmission token is available. Otherwise, after waiting for ready packet for 0.15 milliseconds, the transmission token goes to the next station of the ring. If a packet is not transmitted by the time a new packet is produced 10 milliseconds after the not transmitted packet was produced, the latter is discarded and the new packet is considered for transmission.

### 5.2 TB net of the voice station

In Fig. 7 the voice station is modelled by TB net and in that strong time semantics is assumed. The voice coder is modelled by transition $t_1$ and places $p_1$, $p_2$ and $p_3$. The token (chronos) in p1 records the time at which the last packet has been created by the coder. A token in $p_3$ represents the packet produced by the coder. A token in place $p_2$ records the packet creation time. Its timestamp is used by transition $t_2$ to determine whether the packet can still be processed or must be discarded. Transition $t_1$ fires 10 milliseconds after the production of a token in place $p_1$, i.e. 10 milliseconds after the production of the last packet.

The station interface is modelled by transitions $t_2$, $t_3$ and $t_4$ and places $p_4$, $p_5$ and $p_6$. A token in place $p_4$ represents the interface ready to accept a new packet from the voice coder; a token in place $p_6$ represents the interface ready to transmit a packet on the transmission medium. The firing of transition $t_3$ represents the station interface accepting (place $p_4$) a new packet produced by the coder (place $p_3$) and putting the newly produced packet in the transmission buffer (place $p_6$), ready to be transmitted. Transition $t_3$ can fire 0.2 milliseconds after both the packet has been produced and the interface is ready to accept a new packet. This can occur only within 10 milliseconds after the packet has been produced (10 milliseconds after place the packet has been produced); otherwise the packet is no longer considered for transmission and is discarded (transition $t_4$). The tokens stored in place $p_5$ represent all the messages that could not be transmitted. Transition $t_2$ represents the station interface becoming ready to accept a new packet from the coder if the current packet to be transmitted cannot be transmitted because its validity time elapsed. The transition token flowing through the station is modelled by transitions $t_5$ and $t_6$ and places $p_7$ and $p_8$. A token in $p_7$ represents the availability of a transmission token at the station interface. If a packet is ready for transmission ($p_6$ is marked) 0.1 millisecond after the arrival of the transmission token in the interface, $t_5$ fires, i.e. the packet is transmitted and the interface becomes ready for accepting a new packet. If no packet is ready for transmission, transmission token exits

the interface 0.15 milliseconds after its arrival, which is represented by $t_6$ and its $tf_{t6}$ function. Transition $t_7$ models the return of the transmission token from the ring.




Time-Functions:

$tf_{t1}(p_1) = (\tau = p_1+10)$
$tf_{t2}(p_2,p_6) = (\tau = p_2+10 \text{ and } \tau \geq p_6)$
$tf_{t3}(p_3,p_4) = (\tau = \max(p_3,p_4)+0.2 \text{ and } \tau<p_3+10)$
$tf_{t4}(p_2,p_3) = (\tau = p_3+10 \text{ and } p_2=p_3)$
$tf_{t5}(p_2,p_6,p_7) = (\tau = p_7+0.1 \text{ and } \tau \geq p_2 \text{ and } \tau \geq p_6)$
$tf_{t6}(p_7) = (\tau = p_7+0.15)$
$tf_{t7}(p_8) = (p_8 \leq \tau \leq p_8+8)$

Initial marking:

$m_0(p_1) = \{18\}$ $m_0(p_4) = \{10\}$
$m_0(p_2) = \{18\}$ $m_0(p_8) = \{10\}$
$m_0(p_3) = \{18\}$

Fig. 7. TB net of the voice station

The fsa of the type $M_w$ for the basic PN of the TB net of the voice station is depicted in Fig. 8. In (Hudák & Teliopoulos, 1997) it was demonstrated how *dti τen* can be created and that it can be either mono-, or non mono-generated TI.

In the rest of this section we will use the notation $\vec{x}$ for an interval from *0* to *x* , $T(\sigma)$ for a set of transitions of a sequence $\sigma$ and $P(\sigma)$ for a set of places connected with $\sigma$ . More precisely

$$T(\sigma) = \{t \mid t \in T \wedge \sigma = t_1 t_2 \ldots t_{|\sigma|} \wedge \exists t_i (t_i = t \wedge i \in 1\ldots|\sigma|)\},$$

$$P(\sigma) = \{p \mid p \in P \wedge \exists t (t \in T(\sigma) \wedge p \in {}^\bullet t \cup t^\bullet)\}.$$

Let us consider now the situation in the voice station and look at how *τens* are generated. In the Fig. 8 we can see *selffeeded* loops $\sigma_t$ there, e.g. $\sigma_t = t_7 t_1 t_3 t_5 t_7 t_1 t_3 t_5 \ldots$. Several remarks are in the order now as far as ρ- simple loops are concerned:

1. $\sigma_t$ is selffeeded loop; that simply means that for any $p \in P(\sigma_t)$ and such that $p \in t_j^\bullet$ there exists a transition $t_i \in T(\sigma_t)$ and $p \in {}^\bullet t_i$, and also it holds that for any $p \in P(\sigma_t)$ and such that $p \in {}^\bullet t_i$, there exists a transition $t_j \in T(\sigma_t)$ such that $p \in t_j^\bullet$.
2. let $en_t^{(0)} = TIP(\ldots,\tau_1,\ldots,\tau_2,\ldots)$ denotes that $\tau_1, \tau_2$ belong to $en_t^{(0)}$ and for any $\tau \in TIP(\ldots,\tau,\ldots)$ it is true that $\tau$ is *t'*- generated for some $t' \in T(\sigma_t)$;
3. we say $\sigma_t = t t_1 \ldots t_r$ forms *a chain*, if there exists a sequence of places $p_1, p_2, \ldots, p_r$ such that $p_{i-1} \in {}^\bullet t_i \cap t^\bullet{}_{i-1}$; in that case

$$\tau^{(n)} = \tau en^{(0)} + tf_{\sigma_t}(0) \quad \text{and}$$
$$tf_{\sigma_t}(0) = tf_t(0) + tf_{t_1}(0) + \ldots + tf_{t_r}(0)$$

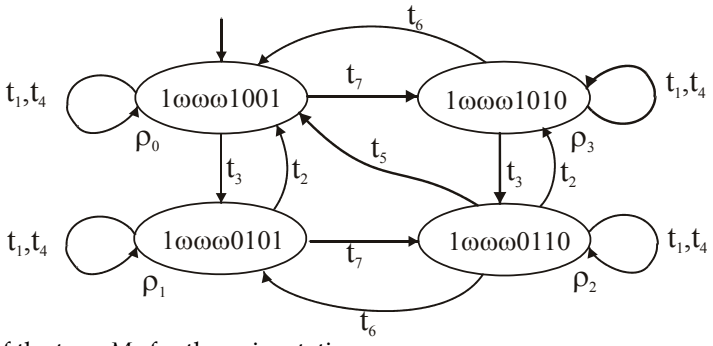



Fig. 8. FSA of the type $M_w$ for the voice station

We assume first, that $\sigma_t$ is not a chain. However, according to the remark 1, we can create a chain $\sigma_t^{(1)} = t' t_1' t_2' ... t_{r1}'$ to be a part of $\sigma_t$. Let further $\tau_1, \tau_2$ be $t_1'$ - and $t_1''$-generated *dti*s respectively. Then two different properties the dtis may have:

**i.** $t_1', t_1'' \in T(\sigma_t^{(1)})$ or

**ii.** $t_1', t_1''$ belong to different chains.

In the *i.* case if $t_1'$ precedes $t_1''$ then $\tau_1^{(1)} \prec \tau_2^{(1)}$ and also $\tau_1^{(i)} \prec \tau_2^{(i)}$ for any $i > 1$. The latter follows from Lemma 1, and Lemma 2 guarantees that TIP's relations are preserved on loops. So in any chain containing $t'_i$

$$\tau_{t_i'}^{(n)} = \tau en_{t_i'}^{(n-1)} + tf_{t_i'}^{(0)}$$

In Fig. 9 we demonstrate how the limits of *dti* $\tau en$ can be created. Any $\tau$ generated in a chain preserves the relation $\prec \in \{<_{sq}, \leq\}$ between $\tau$ and the *TI*s generated in the chain in the previous times. We can see that TI limits' generation schema can be quite complicated (see Fig. 9). As we go through the chain, the situation may get even more complicated (Fig. 10). In spite of some irregularity of the situation from the start, the process of calculation of TIs in the TB net of voice station, as Fig. 10 shows, $\tau en_{t_i'}$ changes after $t_{i-1}'$ firing and what will determine the $\tau_{t_i'}$ TI to be generated upon next $t_i'$ firing. It seems reasonably to count separately the changes of $\tau eni$ and $\tau ena$. We have to realize that the case *d*) in Fig. 9 shows the typical situation in evaluating $\tau en^{(n-1)}$ for the n-th firing.

The same phenomenon can be discovered in ii) case either; for any TIP on the loop $\sigma_t$ (selffeedable) every TI $\tau$ "arriving" at *p* will preserve $\prec \in \{<_{sq}, \leq\}$ with its predecessor. So

$$\tau_j^{(i)} \prec \tau_j^{(i+1)} \quad j = 1,2$$

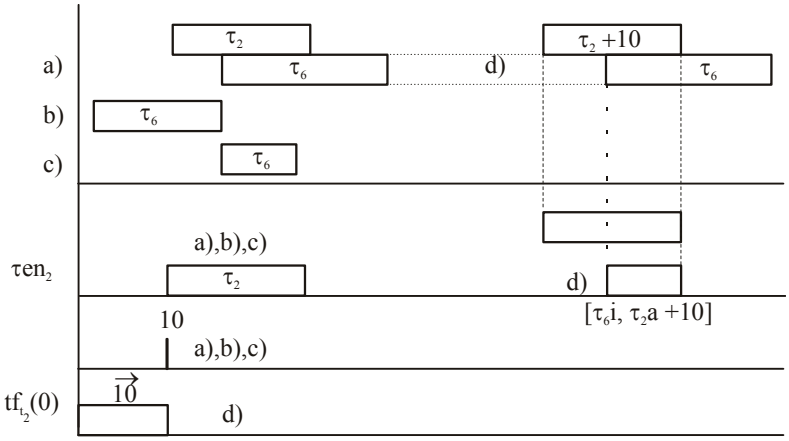



Fig. 9. TIP based calculation of dti $\tau en$ for $t_2$

Now what we have to cope with is the problem what kind of relations we can expect between $\tau_1^{(i)}$ and $\tau_2^{(i)}$. We have to explore the following:

1. that chain approach to estimate interval limits is well founded, and
2. that selffeedable loop $\sigma_t$ guaranties the stabile regime i.e. if $\sigma_t = tt_1...t_r$ is selffeedable loop, and

$$\tau_t^{(n)} = [\tau en_\alpha i + n \cdot \tau eni(tf_{\sigma_t}(0)), \tau en_\alpha a + n \cdot \tau ena(tf_{\sigma_t}(0))]$$

where $\tau eni(tf_{\sigma_t}(0))$ - expresses the contribution to the left limit of $\tau en_t$ by $\sigma_t$ -loop, and the same meaning wrt the right limit of $\tau en_t$ is attached to the $\tau ena(tf_{\sigma_t}(0))$ (see below ) .

Let us compute TI $\tau_4$ that can be $t_2$ -or $t_5$ -generated (Fig.7). We choose first for $\tau_4$ to be $t_5$ -generated. According to Theorem 3 (Hudák, 1996), and Fig.7 we may write

$$\tau_4 = \tau en(\tau_2, \tau_6, \tau_7) + tf_{t_5}(0)$$

so

$$\tau en(\tau_2, \tau_6, \tau_7) = [\tau_6 i, \tau_7 a], \quad tf_{t_5}(0) = \overrightarrow{0.1}$$

TI $\tau en$ will be determined (in the case of $\tau_4$ as being $t_5$-generated) by $\tau_6 i$ and $\tau_7 a$ as far as the lower and upper bound of $\tau en$ is concerned respectively. We are able to create a chain of transitions leading to creation of $\tau_6$ (and thus $\tau_6 i$ ). Then chain will be $\{t_1 t_3\}$ . To generate $\tau_7$ (and thus $\tau_7 a$ ) $t_7$ has to be fired so the chain for $\tau_7$ is $\{t_7\}$ .

To make creation of $\tau_6$ and $\tau_7$ repeatedly the loop $\sigma_t = t_1 t_3 t_7 t_5$ has to be executed repeatedly in the stable regime. From $\sigma_t$ only the part $t_1 t_3 t_5$ "works" to create the lower bound of $\tau en$ of, and on the other hand the part $t_7 t_5$ is the only chain of transitions participating to create the upper bound of $\tau en$ of $t_5$. We prefer to denote the parts of the loop $\sigma_t$. We denote by $iL(t_5) = t_1 t_3 t_5$ the part *which determines the* $\tau en_{t5} i$ and by $aL(t_5) = t_7 t_5$ the part *which determines the* $\tau en_{t5} a$ . Let $\tau en_{ti}$ be the initial value of the determinate TI of the transition $t_i$ prior to its first firing as the member of the stable loop $\sigma$, i.e.

$$\tau_{ti}^{(1)} = \tau en_{ti} + tf_{ti}(0)$$

We denote be $\tau_{ti}^{(1)}$ $t_i$ - generated TI after $j$-th firing of $t_i$ in the stable loop σ and

$$\tau_{ti}^{(j)} = \tau en_{ti}^{(j-1)} + tf_{ti}(0)$$

According to Lemma 1, Lemma 2, i.e. due to properties of TB nets and the stabile selffeded loops we have that

$$\tau_{ti}^{(j)} \prec \tau_{ti}^{(j+1)}$$

and also

$$\tau en_i^{(j)} \prec \tau en_i^{(j+1)} \quad \prec \in \{<_{sq}, \leq\}$$

Assume $\tau en_t = [\tau_{t'} i, \tau_{t''} a]$. To calculate $\tau en_t^{(j+1)}$ in the case of the stabile loop, we have to consider the contribution of σ to the lower and upper bound of $\tau en$. The contribution is denoted by

$$tf_{iL(t)} \text{ and } tf_{aL(t)}$$

to be the contribution to the lower and upper bound respectively. In our example for $t_5$-generated $\tau_4$ we get

$$\tau_4^{(n+1)} = \tau_{t_5}^{(n+1)} = \tau en_{t_5}^{(n+1)} + tf_{t_5}(0)$$
$$\tau en_{t_5}^{(n+1)} = [\tau_6^{(n)} i, \tau_7^{(n)} a]$$

To compute $\tau_6^{(n)} i$ and $\tau_7^{(n)} a$ we first compute $\tau_6^{(1)}$, $\tau_7^{(1)}$. Notice

$\tau_6$ is $t_3$ - generated i.e. $\tau_6 = \tau_{t3}$
$\tau_7$ is $t_7$ - generated i.e. $\tau_7 = \tau_{t7}$

We compute $\tau_t^{(1)}$ for all $t \in T(\sigma)$; in this particular case. We start with the calculation of initial values for $\sigma' = u\sigma\sigma$ ... provided $\sigma = t_7 t_1 t_3 t_5$ and $u = t_3 t_7 t_2 t_6 t_7 t_6$:

$$\tau_{t_3} = \tau_6 = tf_3(0) + \tau en_3 = 0.2 + dti_2(10,18) = 0.2 + 18 = 18.2$$
$$\tau_{t_7}^{'} = tf_7(0) + \tau en_7 = \vec{8} + 10 = [10,18]$$
$$\tau_{t_6} = tf_6(0) + \tau en_2 = 0.5 + \tau_{t_7} = 0.15 + [10,18] = [10.15,18.15]$$
$$\tau_{t_7}^{*} = tf_7(0) + \tau en_7 = \vec{8} + \tau_{t_6} = \vec{8} + [10.15,18.15] = [10.15,26.15]$$
$$\tau_{t_5} = tf_5(0) + \tau en_5 = 0.5 + [\max(\tau_6 i, \tau_2 i, \tau_7 i + 0.1), \tau_7 a]$$
$$= \overrightarrow{0.1} + [\max(18.2,18,10.25), 26.15] = [18.2, 26.25]$$

We are going now to construct $i$- and $a$- determinate loops for the TB net of the voice station (Fig. 7). We are choosing first $\sigma = t_7 t_1 t_3 t_5$. For the loop chosen we have $T(\sigma) = \{t_1, t_3, t_5, t_7\}$ and $P(\sigma) = \{p_1, p_2, p_3, p_4, p_6, p_7, p_8\}$.

We can see (Fig. 7) that in $P(\sigma)$ there will be both:

- *mono-generated* TIs ($\tau_1$, $\tau_2$, $\tau_3$, by $t_1$, while $\tau_7$ by $t_7$) and also
- *non mono-generated* TIs ($\tau_4$ either by $t_5$ or $t_2$, while $\tau_8$ either by $t_5$ or $t_6$)

In the following calculations based on the loop σ we choose for $\tau_4$ and $\tau_8$ to be $t_5$-generated. In the table below you can see the values of $tf_{ti}(0)$ for $i$ = 1,2,3,5,6,7.

| | i | a |
|---|---|---|
| $tf_{t1}(0) = 10$ | 10 | 10 |
| $tf_{t2}(0) = 10$ | 10 | 10 |
| $tf_{t3}(0) = \overline{0.2}$ | 0 | 0.2 |
| $tf_{t5}(0) = \overline{0.1}$ | 0 | 0.1 |
| $tf_{t6}(0) = 0.15$ | 0.15 | 0.15 |
| $tf_{t7}(0) = \vec{8}$ | 0 | 8 |

Table 1. Values of $tf_{ti}(0)$

Now we can calculate

$$tf_{iL(t_5)}(0) = (tf_{t_1}(0))_i + (tf_{t_3}(0))_i + (tf_{t_5}(0))_i = 10 + 0 + 0 = 10$$
$$tf_{aL(t_5)}(0) = (tf_{t_7}(0))_a + (tf_{t_5}(0))_a = 8 + 0.1 = 8.1$$

Now we write the formula to calculate the value of $\tau_4$

$$\tau_4 = [18.2 + n \cdot tf_{iL(t_5)}(0), 26.25 + n \cdot tf_{aL(t_5)}(0)]$$
$$\tau_4 = 18 + [0.2 + n \cdot 10, \quad 8.25 + n \cdot 8.1]$$

The expression for $\tau_4$ has a peculiar feature. Notice that after each iteration of $\sigma_t = t_1t_3t_7t_5$ the lower bound of TI $\tau_4$ will be increased by 10, while the upper bound by 8.1. So after some number of iteration $\tau_4$ become a dummy interval with lower bound greater then the upper bound!!!

Fig. 10 illustrates the method used to predict TIs of TB net for the voice station. Notice the tendency of TIs in places $p_4$, $p_7$ and $p_8$. They tend to become shorter, when we go through the loop chosen.

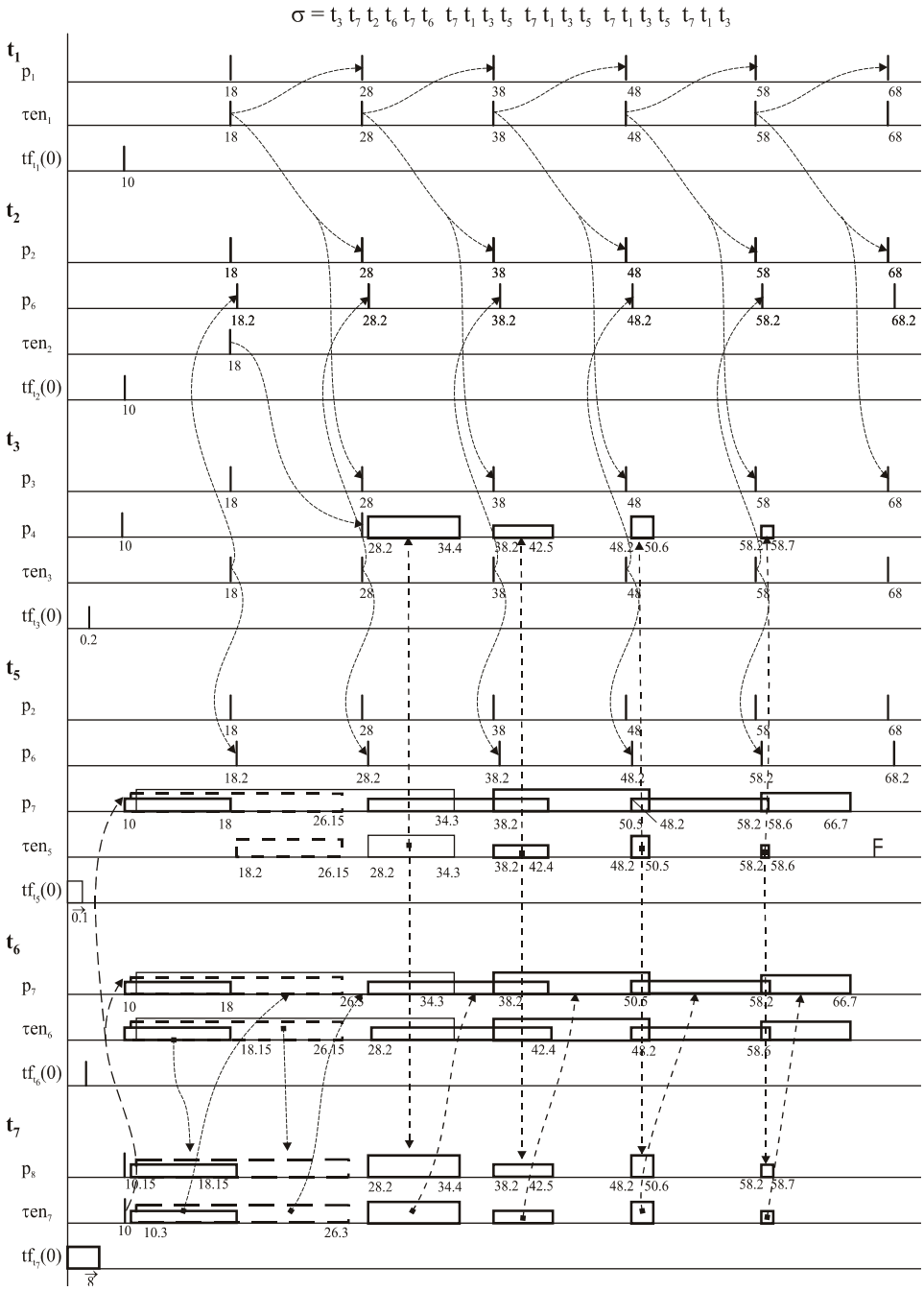



Fig. 10. Calculation of TIs

## 6. mFDT Environment

### 6.1 Motivation and formal methods involved

One of the assertions widely accepted in formal methods community states, that no single notation will ever address all aspects of a complex system (Bowen & Hinchey, 2006). This is also the case of Petri nets, which provides means to express non-determinism and concurrency, an easy-to-understand graphical notation and valuable analytical properties, but lacks other features, such as a verified development process and a formally sound and effective de/composition techniques. To cope with such an "incompleteness" of formal methods there have been many attempts to their integration. One of them was a proposal (Hudák & Grofčík, 2001) to develop a toolset called *multi Formal Description Technique Environment* (*mFDT Environment*, *mFDTE*), which will integrate Petri nets (PN) with two methods with complementary features – the *B-Method* and process algebras *ACP* and *APC*.

The ***B-Method*** (Abrial, 1996), with its B-Abstract Machine Notation (*B-AMN*) specification language, is a state-based model-oriented formal method. It offers a well-defined development process, which allows to specify a software system as a collection of so-called B-machines and to refine such an abstract specification to a concrete one. A consistency of the abstract specification and correctness of refinement are verified by means of proof obligations (*PObs*). There is an industrial tool, called Atelier B (Atelier B, 2009), which supports the whole development process and includes prover for PObs. The B-Method can be used for an additional analysis and implementation of PN models. On the other hand, we can confront invariants, listed in B-AMN specification, with invariants derived from corresponding PN.

***Process algebras*** view systems as processes, described in an algebraic way. In Process algebras we can deal with de/composition of systems very elegantly, because they support compositionality by definition. We picked out the *Algebra of Communicating Processes* (*ACP*) (Baeten & Weijland, 1990) and developed a new *Algebra of Processes Components* (*APC*) (Šimoňák, 2003; Šimoňák et al., 2008). APC is a modification of ACP, which allows a comfortable description of PN processes. Analytical apparatus of PN can be used for verification of process algebraic specification of a system and process algebras can be used for a de/composition of PN.

### 6.2 mFTDE structure and tools

The mFDTE will consist of tools for integrated formal methods and interfaces between languages of these methods (Fig. 11). Tools will allow designer to gain from advantages of individual methods and interfaces will provide correct and formally proved translation from a specification in one method to the equivalent specification in another one.

The tools are in an implementation and testing phase now and can be obtained by request from the authors. Current versions of the tools are described below and translation processes of existing interfaces in the following subsections.

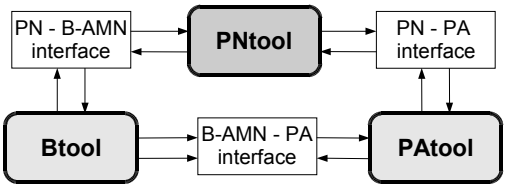



Fig. 11. mFDTE structure

The ***PNtool*** (Fig.12) is, quite naturally, a hearth of mFDTE and provides the richest functionality. The tool supports Generalized PN (*GPN*, also known as Place/Transition nets), TB nets, Evaluative PN (*EvPN*) and a limited subset of Coloured PN. EvPN (Hudák, 1980) is a Turing-powerful extension of GPN, which allows negative markings ($m(p)<0$) and place capacities defined with respect to individual arcs. In EvPN it is also possible that a change of a net marking, caused by a firing of some transition t, depends also on the marking of places, which are not adjacent to *t*.

PNtool provides a graphical editor and simulator for all supported PN types. The Petri Net Markup Language (PNML), an XML-based interchange file format for Petri-nets, is used to store GPN models. This allows using GPN created in another software tools for Petri nets, such as Petri Net Kernel, Renew, PEP and TINA. An extended version of PNML is used also for EvPN and we plan another extension for TB nets.

Computation of S- and T-invariants and the reachability analysis is supported for GPN. The current version of PNtool implements the first step of RP algorithm – creation of fsa $M_w$. For educational purposes the tool includes a step-by-step visualization of $M_w$ creation. The automaton created can be saved in the form of Petri net, using another modification of PNML. PNtool also contains a part of PN - B-AMN interface, allowing a translation of GPN and EvPN to computationally equivalent B-machine.

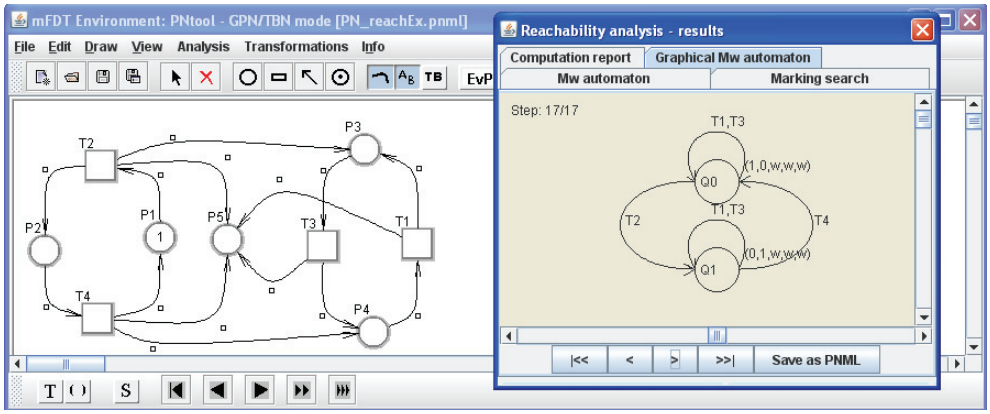



Fig. 12. The PNtool in GPN/TBN mode with reachability analysis results window open

The ***Btool*** focuses on a translation from B-AMN to JAVA. The translation process is inspired by that of jBTools (Voisinet, J.C. et al., 2002), but differs in various aspects, such as machines import mechanism and handling of output parameters.

Finally, the ***PAtool*** includes an editor for ACP and APC specifications and the PN – PA interface to translate the specifications from and to PN. To store the specifications a newly developed XML –based Process Algebra Markup Language is used.

### 6.3. Petri nets – B-AMN Interface

A theory of translations between B-AMN and PN, introduced in (Korečko, 2006), makes it possible to transform any GPN or EvPN into the computationally equivalent B-machine and almost any B-machine into the equivalent Coloured PN.

The *B-machine* is an abstract specification component of B-Method In general, a B-machine consists of a set of state variables (clause VARIABLES), an invariant to restrict the variables (clause INVARIANT), an initial operation to establish an initial state (INITIALISATION) and a set of operations to modify the variables (OPERATIONS). There are also other clauses intended for additional assertions and data components (parameters, sets and constants).

A basic idea of the translations is to link together similar behavioural concepts of both methods. Therefore places of PN are transformed to state variables of B-machine, initial marking to initialisation operation, transitions and adjacent arcs to operations and vice versa.

By translation of some GPN or EvPN $N$ we get a computationally equivalent B-machine $\pi(N)$ ($\pi$ is a mapping from PN to B-AMN). The two specifications, $N$ and $\pi(N)$, are in fact bisimilar.

```
MACHINE mchPiN
VARIABLES sv_1, sv_2, sv_3, sv_4, sv_5
INVARIANT sv_1∈ℕ ∧ sv_2∈ℕ ∧ sv_3∈ℕ ∧ sv_4∈ℕ ∧ sv_5∈ℕ
INITIALISATION sv_1:=1 || sv_2:=0 || sv_5:=0 || sv_3:=0 || sv_4:=0
OPERATIONS
  op_t1= SELECT sv_4>=1 THEN sv_5:=sv_5 + 1 || sv_3:=sv_3 + 1 || sv_4:=sv_4 - 1 END;
  op_t2= SELECT sv_1>=1 THEN sv_2:=sv_2 + 1 || sv_1:=sv_1 - 1 || sv_5:=sv_5 + 1 || sv_3:=sv_3 +
1 END;
  op_t3= SELECT sv_3>=1 THEN sv_5:=sv_5 + 1 || sv_3:=sv_3 - 1 || sv_4:=sv_4 + 1 END;
  op_t4= SELECT sv_2>=1 THEN sv_2:=sv_2 - 1 || sv_1:=sv_1 + 1 || sv_5:=sv_5 + 1 || sv_4:=sv_4 + 1
END
END
```

Fig. 13. B-machine obtained from the Petri net $N$ from Fig.1

A B-machine *mchPiN*, obtained from the net $N$ from Fig.1, can be seen in Fig.13. Values of machine variables are naturals ($\mathbb{N}$) and correspond to markings of $N$ ($sv_i$ to $m(p_i)$). Similarly, operations correspond to transitions of $N$. The operations consists of a guarded command "SELECT *P* THEN *S* END", which means "do *S*, if *P* holds". If *P* doesn't hold, then the command is not feasible. Operator "||" stands for parallel composition, so "$S_1$|| $S_2$" means "do $S_1$ and $S_2$ simultaneously". As it was said, a B-machine obtained by the translation can be used for an additional analysis of PN specification. For example, to check a deadlock freedom of $N$, we add a predicate saying "there must be at least one feasible operation in each state of $\pi(N)$" and prove PObs of $\pi(N)$. The extended invariant for *mchPiN* has the form (3).

$$sv_1\in\mathbb{N} \wedge \ldots \wedge sv_5\in\mathbb{N} \wedge (sv_4>=1 \vee sv_1>=1 \vee sv_3>=1 \vee sv_2>=1) \qquad (3)$$

To allow a refinement of B-machine obtained, we have to use a slightly modified always feasible form of operations with "IF *P* THEN *S* ELSE SKIP END" instead of "SELECT *P* THEN *S* END". A theory of PN to B-AMN translation, including an example of EvPN translation can be found in (Korečko, 2006; Korečko, 2009). The translation can be further extended to high-level Petri nets, e.g. by adapting an approach used in (Kalinichenko et al., 2005).
In an opposite direction a translation is more complicated. For example, we can get more that one PN transition for one operation because of a non-deterministic nature of B-machine operations. Here we use Coloured PN, that match the modelling power of B-AMN while retaining valuable analytical properties. A step-by-step demonstration of the translation from B-AMN to Coloured PN can be found in (Korečko et al., 2008), where it is also shown how a structural analysis of the Petri net obtained can be used to reveal some additional invariant properties, not specified in the original B-machine.

### 6.4. Petri nets – Process Algebra Interface

Transformations of PN-PA interface, introduced in (Šimoňák, 2003), consist of two parts, namely: linguistic semantics preserving transformation of process algebra ACP specification into the corresponding Petri net and the operational semantics preserving transformation of (Ordinary) Petri net into the process algebra APC.
The first of two transformations mentioned, is based on construction of *elementary nets*, corresponding to atomic actions of the ACP specification, including the empty process (ε) and the deadlock (δ). Additionally, *net operations* are introduced, corresponding to operators of the ACP (alternative composition, sequential composition, parallel composition and encapsulation), allowing composition of Petri nets in order to obtain the resulting net, corresponding to the original specification. A description of the transformation, including an example can be found in (Šimoňák, 2006).
The aim of the second transformation is to construct the APC specification from the source Petri net. The approach is based on creating special variables (named E-variables) for every place of given Petri net, expressing processes initiated in those places. Algebraic semantics is given as a parallel composition of all such variables, whose corresponding places hold token(s) within the initial marking. A description and a short example of the transformation can be found in (Šimoňák et al., 2008).

## 7. Conclusion

In this work some results concerned the reachability analysis of time critical systems based on Petri Nets have been presented. The issue is very important, as the nowadays experience with computer based systems shows. The importance of the issue is not only from the practical point of view, but also from the theoretical one. As we know, and also it was demonstrated, the reachability analysis in the case the state space is large, or even infinite, is an intractable problem. Things get even worse, when the time issue comes under consideration. The results presented lay a foundation for coping with the problem. They are based on the original RP algorithm, and the de/compositional method of reachability analysis developed by the first author. The corner stone here are the properties of the finite state automaton of the type $M_w$, that was revealed by a convex analysis approach to the fsa (Hudák, 1999).

In the state diagram of fsa $M_w$ ρ-simple loops play profound role in the reachability analysis of the ordinary PN as the results of this work demonstrate and it has been gathered enough arguments (Hudák & Teliopoulos, 1998b) that the role of ρ-simple loops remains in the issue of TRA of TB nets.

We distinguish two subclasses of loops in $M_{\pi w}$: *selffeeded* and *stabile* loops. The loop is selffeeded one if in a *t* - firing (*t* belongs to the loop) *t* "consumes" only tokens that was created solely by firings of loop's transitions. A loop can be called stabile if at any *t*-firing (*t* belongs to the loop) all tokens at precondition places are uniformly generated, i.e. at any *t*-firing at each repetition *t* consumes tokens from the same generators, i.e. transitions that generated tokens consumed by *t* (Hudák & Teliopoulos, 1998b) . There is a strong relation between the two types of loops (Hudák & Teliopoulos, 1998b). Each loop becomes stabile after some initialization, after that some TIP (we call it initial) is reached which starts stable part of computation.

The proposed algorithm for reachability problem has been partially implemented in the mFDT Enviroment and, thanks to the mFDTE interfaces, can be used also for specifications written in other formal specification languages.

After defining initial TIP for any loop on a path we can define the TIs of any chronos (tokens) in which it can exist in the future. The structure of those TIs reminds very much spectral image of time sequences (Hudák & Teliopoulos, 1998b). Great deal of work has been done already on the study of properties of different kinds of loops from the point of view of feeding transitions on the loop (Hudák & Teliopoulos, 1998a). There are some problems left, specifically from the point of view of different semantics (MWTS, STS) (Ghezzi et al., 1994).

Results of the theory presented show that once $M_{\pi w}$ has been constructed we can predict precisely future of any token and discover perhaps a moment when it disappears because of the emptiness or dummy feature of its TI. For any TI $\tau_t$ (*t* - generated TI with the name τ) we can construct a formula for the TI to be calculated. There are still some problems to be resolved, and we hope to deal with them in the future.

The RP algorithm works almost in the same way in the case of TB nets as it does in the case of ordinary Petri Nets. We hope that the questions raised above will be tackled upon as the subject of further research, and the results achieved will be published elsewhere.

## 8. References

Abrial, J.R. (1996). *The B-book: assigning programs to meanings*, Cambridge University Press, ISBN 0-521-49619-5, Cambridge, U.K.

Baeten, J.C.M. & Weijland W.P. (1990). *Process algebra*, Cambridge University Press, ISBN 0-521-40043-0, Cambridge, Great Britain

Baeten, J.C.M. & Bergstra, J.A. (1991). Real Time Process Algebra. *Formal Aspects of Computing*, Vol.3, No.2, (1991) pp.142-188, ISSN 0934-5043

Billington, J.; Wheeler, G. & Wilbur-Ham, M. (1988). Protean: A high-level Petri net tool for the specification and verification of communication protocols. *IEEE Trans.Software Eng*. Vol.14, No.3, (March 1988) pp. 301-316, ISSN 0098-5589

Bowen, J.P. & Hinchey, M.G. (2006). Ten commandments of formal methods... ten years later. *Computer*, Vol. 39, No. 1, (January 2006) pp. 40- 48, ISSN 0018-9162

Bruno, G. & Marchetto, G. (1986). Process-translatable Petri nets for the rapid prototyping of process control systems. *IEEE Trans.Software Eng.* Vol.12, No.2, (February 1986) pp. 346-357, ISSN 0098-5589

Genrich, H.J. (1986). Predicate/transition nets. In: *Advances in Petri Nets 1986*, Brauer, W.; Reisig, W. & Rozenberg, G. (Ed.), pp. 207-247, Springer Verlag, ISBN 0-387-17905-4, New York

Genrich ,H.J. & Lautenbach,K. (1981). System Modelling with High-Level Petri Nets. In: *Theoretical Computer Science 13*, pp. 109-136

Ghezzi, C.; Mandrioli, D.; Morasca, S. & Pezze,M. (1991). A unified high-level Petri net formalism for time-critical systems. *IEEE Trans.Software Eng.* Vol.17, No.2, (February 1991) pp. 160-172, ISSN 0098-5589

Ghezzi, C.; Morasca, S. & Pezze, M. (1994). Validating Timing Requirements for TB Net Specifications. *Journal of Systems and Software*, Vol.27, No.7, (November 1994) pp. 97-117, ISSN 0164-1212

Hudák, Š. (1980). *Extensions to Petri Nets*, Habilitation Thesis, Technical University of Košice, Slovakia

Hudák, Š. (1981). *The recursive decidability of the reachability problem for vector addition systems.* The University of Newcastle upon Tyne, Computing Laboratory, ASM/84, August 1981 (also in Proceedings of The Second European Workshop on the Theory and Applications of Petri Nets, Bad Honnef, Germany, September 1981).

Hudák, Š. (1994). De/compositional Reachability Analysis. *Journal of Electrical Engineering*, Vol.45, No.11, (1994) pp. 424-431, ISSN 0013-578X

Hudák, Š. (1996). Time Interval Semantics of TB nets, *Proceedings of the International Conference RSEE'96*, 12pp, Oradea, Romania, May 1996

Hudák, Š. (1999). *Reachability Analysis of Systems Based on Petri Nets*, elfa s.r.o., ISBN 80-88964-07-5, Košice, Slovakia

Hudák, Š. & Grofčík, J. (2001). An Environment for Design and Analysis of Time-Critical Systems, *Proceedings of EMES'2001*, pp. 66-75, Oradea, Romania, May 2001.

Hudák, Š. & Teliopoulos, K. (1997). TB Nets: properties of Time Interval Profiles, *Proceedings of the International Conference RSEE'97*, 8pp., Oradea, Romania, May 1997

Hudák, Š. & Teliopoulos, K. (1998a). Loop Spectral Analysis of Time Rechability Problem, *Proceedings of RSEE'98*, 11pp, Oradea, Romania, May 1998

Hudák, Š. & Teliopoulos, K. (1998b). TB Nets and TRA of Time-critical Systems, *Proceedings of the Scientific Conference Artificial Intelligence in Industry*, pp. 156-165, High Tatras, Slovakia, April 1998

Jensen, K. & Kristensen, L.M. (2009). *Coloured Petri Nets. Modelling and Validation of Concurrent Systems*, Springer Verlag, ISBN 978-3-642-00283-0

Kalinichenko, L.A.; Stupnikov, S.A. & Zemtsov N.A. (2005). Extensible Canonical Process Model Synthesis Applying Formal Interpretation. *Proceedings of ADBIS'05*, LCNS vol.3631 pp. 183-198, ISBN: 978-3-540-28585-4, Talin, Estonia, September 2005, ISBN 3-540-28585-7, Springer Verlag, Berlin-Heidelberg

Korečko, Š. (2006) *Integration of Petri Nets and B-Method for the mFDT Environment*. PhD thesis. DCI FEEI TU Košice, Slovakia, 2006 (in Slovak)

Korečko, Š.; Hudák, Š. & Šimoňák, S. (2008). Analysis of B-machine based on Petri Nets, P*roceedings of CSE 2008*, pp. 24-33, ISBN 978-80-8086-092-9, Stará Lesná, Slovakia, September 2008, elfa s.r.o, Košice

available from: hornad.fei.tuke.sk/~korecko/pblctns/CSE2008_SKor.pdf
Korečko, Š. (2009). *From Petri nets to B-Method*, Technical report DCI 1/2009, DCI FEEI TU Košice, 2009, available from: hornad.fei.tuke.sk/~korecko/pblctns/trEvPN_B.pdf
Murata, T. (1989). Petri Nets:Properties, Analysis and Applications, *Proceedings of the IEEE*, Vol. 77, No. 4., (April 1989) pp. 541-580, ISSN 0018-9219
Olderog, E.R. (1991). *Nets, Terms and Formulas*, Cambridge University Press, ISBN 0-521-40044-9, Cambridge, U.K.
Ostroff, J.S. (1989).*Temporal Logic for Real-Time Systems*, Research Studies Press Ltd., ISBN 0-08380-086-6, U.K.
Peterson, J.L (1981). *Petri Net Theory and the Modelling of Systems*, Prentice Hall PTR, ISBN 0-136-61983-5, Upper Saddle River, NJ, USA
Reisig, W. (1985). *Petri nets: An Introduction*, Springer Verlag, ISBN 0-387-13723-8, Heidelberg
Šimoňák, S. (2003). *Formal methods integration based on Petri nets and process algebra transformations*. PhD Thesis, DCI FEEI TU Košice, 2003 (in Slovak)
Šimoňák, S. (2006). Formal Methods Transformation Optimizations within the ACP2PETRI Tool. *Acta Electrotechnica Et Informatica*, Vol.6, No.1, (2006) pp. 75-80, ISSN 1335-8243, available from: www.aei.tuke.sk
Šimoňák, S.; Hudák, Š. & Korečko, Š. (2008). APC Semantics for Petri Nets. *Informatica*, Vol. 32, No.3, (2008) pp. 253-260, ISSN 0350-5596, available from: www.informatica.si
Voisinet, J.C.; Tatibouet, B. & Hammad, A. (2002). jBTools: An experimental platform for the formal B method. P*roceedings of PPPJ'02*, pp. 137—140, ISBN 0-901-51987-1 Dublin, Ireland, June 2002, National University Of Ireland, Maynooth
Atelier B website (2009). www.atelierb.eu

# 14

# Supervisory Control and High-level Petri nets

Chiheb Ameur ABID, Sajeh ZAIRI and Belhassen ZOUARI
*LIP2 Laboratory - University of Tunis*
*Tunisia*

## 1. Introduction

The Supervisory Control Theory (SCT) (Ramadge & Wonham, 1989) was developed to provide a formal methodology for the automatic synthesis of controllers for Discrete Event Systems (DES). In this theory, a system, called a plant, is assumed to have uncontrollable behaviours which may violate some desired specifications. Hence, these behaviours have to be controlled by means of a feedback controller, called a controller (or a supervisor), so that the system fulfils the specifications. Primarily, the SCT was studied in the context of automaton based models. More recently, a special interest is given to the Petri net models for the studying of the control problem (Ghaffari et al., 2003; Giua & DiCesare, 1994; Sreenivas & Sreenivas, 1997), since they represent a good trade-off between modelling power and analysis capabilities. For details about the supervisory control problem methods based on Petri nets, one can refer to (Holloway et al., 1997; Su et al., 2005). In addition, high level nets, especially Coloured Petri nets (CP-nets) (Jensen & Rozenberg, 1991), provide a great improvement over the ordinary Petri nets. Notably, the high expressiveness of CP-nets allows to obtain compact models even for large systems, while keeping the same formal analysis capabilities. However, not many works have addressed the supervisory control problem by considering a CP-net as a plant model. In this context, we can cite the method developed in (Makungu et al., 1999). This method addresses the forbidden state problem for a class of CP-nets where the process to be controlled is separated from the control logic.
In this chapter, we review our previous works (Abid & Zouari, 2008; Zouari & Ghedira, 2004; Zouari & Zairi, 2005) for the supervisory control problem of DES modelled by CP-nets. The control specifications herein considered are expressed in terms of forbidden states, i.e. states which have to be avoided by the controlled model. In a first approach, we propose to derive a controller for a plant CP-net model by using the theory of regions. According to the control specifications, the desired behaviours are extracted from the rechability graph associated with the plant model. Then, the theory of regions is used in order to design the controller. Thanks to the expressiveness of CP-nets, as a main advantage, the obtained controller is reduced to one single place. Secondly, we propose to optimise the first approach in order to deal efficiently with symmetric systems. Indeed, the reachability graph of a symmetric system can be represented by an optimised version, called symbolic reachability graph (Chiola et al., 1991; 1997), which is quite smaller. Thereby, the use of symbolic graphs allows to alleviate one important drawback of the latter approach which is the well-known problem of the state space explosion. Moreover and consequently, the use of a smaller graph allows to reduce the complexity of the synthesis process. Finally, we propose an approach which considers as plant model a CP-net that is assumed to be structured on a set of generic processes sharing a set

of resources. In addition to the avoiding of the use of the theory of regions, this approach generates a controller as an active process, modelled by a CP-net, and having the advantage to be implemented directly on existing tools such as CPN-Tools (Jensen et al., 2007).
The remainder of the paper is organised as follow. The second section introduces the notation and definitions used in CP-nets. Section 3 provides the basic concepts of the theory of regions. Section 4 deals with the synthesis of a CP-net controller for the forbidden state problem by applying the theory of regions. Section 5 optimises the latter approach for symmetric systems by applying the theory of regions on the basis of symbolic reachability graphs. Section 6 gives how to design a generic CP-net controller for certain systems without using the theory of regions. Finally, section 7 summarizes the main conclusions and perspectives of this chapter.

## 2. Well-Formed Coloured Petri nets (WF-nets)

High level nets (Jensen & Rozenberg, 1991) represent a natural extension of ordinary Petri net formalism. They enhance both readability and expressivity of Petri nets. As a main advantage, high level nets allow the generation of compact models even for large systems. This extension is mainly done by the introduction of 'colour' structures to identify tokens. Coloured (or in general High-level) Petri nets are particularly well-adapted for the modelling of parametric systems which behaviours depend on the basic structure of the model rather than on the cardinalities of the colour sets. The CP-net model used in this chapter is the Well-Formed coloured Petri nets (WF-nets) model (Jensen & Rozenberg, 1991). WF-nets are equivalent in expressiveness to CP-nets, but are syntactically restricted by enforcing a particular structuring on colour classes and functions.

In this section, we briefly present the different notions related to CP-nets, according to the syntax defined in the WF-net model.
WF-nets have the same modelling power as CP-nets, although syntactically different. Before presenting a formal definition of the model, let us present the related basic notions.
A *multiset* is a set in which given elements may appear several times. Given a set $A$. $Bag(A)$ denotes the set of finite multisets on $A$. A multiset "$a$" can be represented as a sum: $a = \sum_{x \in A} a(x).x$ in which $a(x)$ gives the number of occurrences of the element $x$ in the multiset "$a$".
*Object classes* are finite non-empty sets of objects or basic colours. A class may be viewed as a set made up of elements of the same type. We can distinguish particular type of classes, called ordered classes, for which an order relation is defined on its elements.
*Colour domains* may be defined as a Cartesian product of object classes and is associated with either a transition or a place. When associated with a transition, it defines the set of all its firing instances (coloured firings). When associated with a place, it defines the set of all its possible markings.
*Colour functions* are a pondered sum of tuples of basic colour functions. Colour functions are associated with the labels of WF-net arcs. These functions allow to specify the number of coloured tokens to be consumed and to be produced when firing a given transition. There are three basic colour functions:

- the *identity* function, used for the choice of any object in a class, specified by a variable $X$,
- the successor function, used to specify the circular successor of an object in an ordered class, noted $\oplus X$,

- the diffusion function, used to specify all the objects of a class $C_i$, noted $All_{C_i}$.

A *guard* is a Boolean function defined on a colour domain and which role is to restrict it to a subdomain. When a guard is associated with the colour domain of a transition, it limits its possible firings. But a guard can also be associated with a colour function labelling an arc in order to indicate whether this arc is valid with respect to the guard value.
Let $[g]$ be a guard and $f$ be a colour function. The guarded function $[g].f$ is defined by:
$\forall c \in C(t), [g]f(c) = (\text{if } g(c) \text{ then } f(c) \text{ else } 0)$.
We can note that a guard can lead to a cancelation between a place and a transition.

**Definition 1.** *A coloured Petri net is a 6-tuple*
$N =< P, T, C_l, C, W^-.W^+, \Phi, M_0 >$ *where:*
*P is a finite set of places,*
*T is a set of transitions verifying* $P \cap T = \emptyset, P \cup T \neq \emptyset$,
$C_l = \{C_1, C_2, ..., C_k\}$ *is a set of object classes such that* $\forall i, j \in \{1, ..., k\}, i \neq j, C_i \cap C_j = \emptyset$,
*C is the colour function, defined from* $P \cup T$ *into a set of colour domains. An element c of* $C(s)$ *is a tuple* $< c_1, ..., c_k >$ *and is called a* colour *of s,*
$W^-, W^+$ *are the input and output functions (also called the incidence functions) defined on* $P \times T$*, such that* $W^-(p,t)$ *and* $W^+(p,t)$ *are guarded colour functions representing linear applications mapping* $Bag(C(t))$ *onto* $Bag(C(p))$*, for all* $(p,t) \in P \times T$,
$\Phi$ *is a function which associates a guard with any transition. By default* $\Phi$ *is* true *for any transition t,*
$M_0$ *the initial marking is a function defined on P, such that* $M_0(p) \in Bag(C(p))$*, for all* $p \in P$.
$W = W^+ - W^-$ *indicates the incidence matrix, and* $W(p,.)$ *is a line vector of such a matrix.*

For reasons of clarity, we assume in this paper that the object classes are not ordered.
The dynamic behaviour of a coloured Petri net is determined by the following firing rule:
A transition $t$ is enabled for a colour $c$ and a marking $M$, denoted by $M[t,c\rangle$, iff $\forall p \in P, M(p) \geq W^-(p,t)(c)$.
The marking $M'$ obtained after the firing of $(t,c)$ is computed as:

$$\forall p \in P, M'(p) = M(p) - W^-(p,t)(c) + W^+(p,t)(c)$$

The notation $M[t,c\rangle M'$ is used to indicate this reachability relation. Using the firing rule, it is possible to construct a reachability graph $R(N)$, whose nodes are the markings reachable from the initial marking, and whose arcs represent the reachability relation. Such an arc is labeled by the transition name and the associated colour involved in the reachability relation between two given nodes.

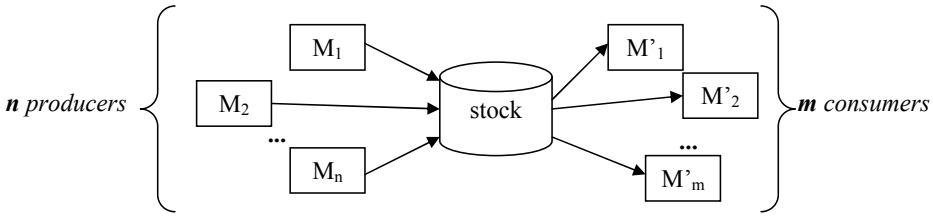



Fig. 1. Problem of producer-consumer

Throughout this chapter, we consider the well-known producer-consumer problem. As it is illustrated in Fig. 1, there are two kinds of machines, namely producers and consumers, sharing a stock. We have $n$ producers and $m$ consumers. A producer can produce an object and transfers it in the stock, while a consumer operates by using an object which have been already produced and transferred in the stock by a producer. The WF-net modelling this problem is illustrated in Fig. 2. For sake of simplicity, we have reduced the behaviour of a consumer to one state and one action. The consumption of an object, deposited in the stock, is traduced by the execution of transition $t3$. When a producer produces an object, it transfers it in the stock by executing transition $t2$. When place $p3$ contains no tokens, it indicates that the stock is full (producers can not transfer a new object in the stock). When place $p4$ contains no tokens, then the stock is empty. $C_1$, $C_2$ and $C_3$ denote the object classes of this net. $C1$ represents the producers, $C2$ allows to indicate the state of the stock and $C_3$ represents the consumers. The set of places of this net is $P = \{p1, p2, p3, p4\}$. The colour domains of places are: $C(p1) = C(p2) = C_1$, $C(p3) = C(p4) = C_2$ and $C(p5) = C_3$. The set of transitions is $T = \{t1, t2, t3\}$. The colour domains of transitions are: $C(t1) = C_1$,$C(t2) = C_1 \times C_2$ and $C(t3) = C_2 \times C_3$.

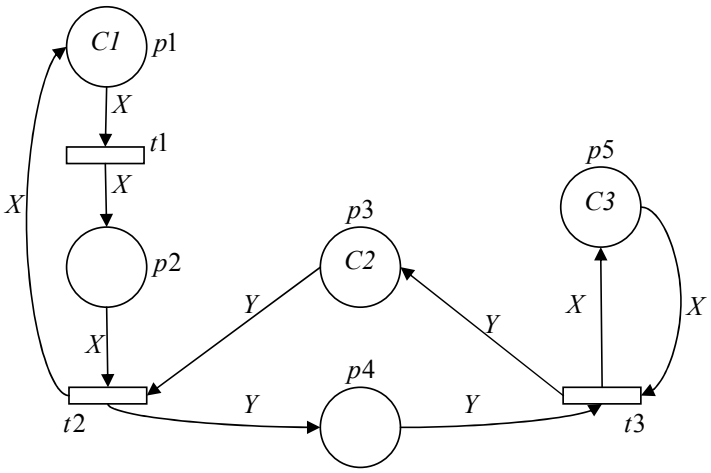



Fig. 2. Well-formed net of producer-consumer problem

## 3. Theory of regions

The aim of the theory of regions (Badouel et al., 1995) is to decide wether a given automaton is isomorphic to the reachability graph of a net, then constructing it. Ghaffari et al. (Ghaffari et al., 2003) are the first who proposed an adaptation of this theory for the synthesis controller problem using Petri nets. Their proposed method allows to add control places to an initial Petri net in order to avoid reaching undesired states. Considering a plant model modelled by a Petri net, this method starts by constructing its associated reachability graph. After that, one has to provide markings which must be avoided by the system. These markings correspond to *forbidden states*. This step enables to identify the *dangerous markings* as the predeces-

sors of markings that allow reaching forbidden states by *uncontrollable events*. An uncontrollable event corresponds to a transition beyond any control procedure. Forbidden and *blocking* markings are removed in order to obtain a strongly connected modified reachability graph that respects liveness property and implements the maximally permissive legal behaviour. A marking is said blocking if it does not allow to reach a *final state*. Such a state represents a proper termination of some task and corresponds to a stable state of the system. From the obtained graph, the parameters of a pure control place, to be connected to a plant model, are computed by resolving a linear system of equations. According to the theory of regions, three classes of equations are defined. The work of Ghaffari et al. is based on the following theorem which defines how to derive a controller for an ordinary Petri:

**Theorem 1.** *Let $N =< P, T, W, M_0 >$ be a bounded Petri net such that P is a set of places, T is a set of transitions, and $M_0$ is its initial marking. Let R be the reachability graph of N. Let $R_c$ be the desired legal behaviour of N ($R_c$ is a subgraph of R). The supervisory control problem can be optimally solved by adding a set of control places $P_c$ to N iff there exists a solution $(M_0(p_c), W(p_c, .)), \forall p_c \in P_c$ satisfying the following equations:*

1. *The* reachability equation *for every marking in $R_c$:*

$$M(p_c) = M_0(p_c) + W(p_c, .)\overrightarrow{\Gamma}_M \geq 0 \tag{1}$$

   *where $\Gamma_M$ is a non oriented path of G from $M_0$ to M and $\overrightarrow{\Gamma}_M$ is its associated vector, called the vector counting of $\Gamma_M$. $\overrightarrow{\Gamma}_M$ is indexed by transitions of T. Each line $\overrightarrow{\Gamma}_M[t]$ represents the algebraic sum of occurrences number of t in $\Gamma$.*

2. *The* cycle equation *of $R_c$*

$$W(p_c, .)\overrightarrow{\gamma} = 0, \forall \gamma \in S_c \tag{2}$$

   *where $S_c$ is the set of cycles of G, and $\overrightarrow{\gamma}$ is the vector counting defined similarly as $\overrightarrow{\Gamma}_M$.*

3. *For each pair $(M, t)$ such that t does not fire from M, it exists at least one control place $p_c$ which satisfies the equation of* state separation inequation*:*

$$M_0(p_c) + W(p_c, .)\overrightarrow{\Gamma}_M + W(p_c, t) < 0 \tag{3}$$

Equation of type (1), called reachability conditions, indicates that every reachable marking within the legal behaviour must remain reachable under control. Similarly, the cycle equations (2) indicate that the cycles must remain reachable under control. Finally, an equation of type (3), called an event separation condition, specifies for a pair $(M, t)$ that the control must prevent the transition $t$ from firing in marking $M$.

## 4. Synthesis of controllers for CP-nets

In this section, we present a controller synthesis approach for a DES modelled by a CP-net, where the control specifications are expressed in terms of forbidden markings. According to the provided control specifications, we determine the *admissible behaviours* from the reachability graph of the plant model, which are represented by an appropriate graph, called the *admissibility graph*. An admissible behaviour represents a behaviour of the controlled system under both safety specification and non-blocking requirement. Thanks to the expressiveness of CP-nets, the controller to be determined is reduced to one single CP-net place. Its parameters are obtained by applying the theory of regions on the basis of the computed admissibility graph.

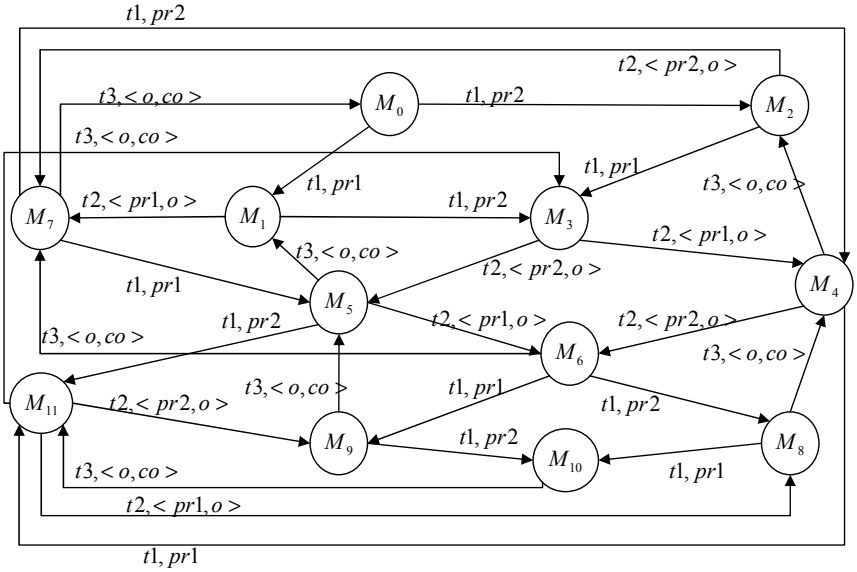



Fig. 3. Reachability graph of the producer-consumer problem

### 4.1 Computation of the admissibility graph

Let us assume that we have the reachability graph of a plant CP-net model. The control constraints considered herein are specified through a finite set of undesired markings. These markings and blocking ones are qualified as *forbidden markings*. A forbidden marking must not be reached by the controlled model. Thus, the key idea for the determination of the admissibility graph is to remove forbidden markings from the initial reachability graph of the plant model, and also to identify markings which lead inevitably to forbidden ones.

The SCT classifies the transitions into two categories. First category consists of the *controllable transitions* which may be disabled when it is necessary. In contrast, the transitions belonging to the second category, called *uncontrollable transitions*, are beyond any control procedure. Hence, we assume that the transition set $T$ of the plant model is partitioned into two disjoint subsets: the set $T_c$ of controllable transitions, and the set $T_u$ of uncontrollable ones.

The role of a controller is to restrict the behaviour of the plant model by disabling some controllable transitions in order to avoid reaching forbidden states. The disabling of a controllable transition is performed in a *dangerous marking* from which the firing of the transition leads to a forbidden marking. So that, we have to identify the set $\Omega$ of *state-transitions* to be disabled. Every element of $\Omega$ is a couple $(M,(t,c))$ where $M$ is a dangerous marking, and $t$ is a controllable transition such that the firing of $t$ with colour $c$ from $M$ yields to a forbidden marking.

As we have previously mentioned, the admissibility graph is computed from the reachability graph of the plant model by removing forbidden nodes. In addition, nodes becoming unreachable from the initial marking and the non coreachable markings must be removed from the admissibility graph. The identification of dangerous and forbidden nodes is performed according to the following rules:

- a marking is qualified as dangerous if it has at least one output arc where its destination is a forbidden marking,
- a marking is qualified as forbidden when it has no output arcs and it is not a final marking, or it is a dangerous marking and it has at least one output arc labelled by an uncontrollable transition,
- every forbidden marking must be removed with its input and output arcs.

The computation of the admissibility graph $R_c$ and the set $\Omega$ is given by Algorithm 1. It is worth noting that it is an enhanced version of the algorithm proposed in (Zouari & Ghedira, 2004). The algorithm considers the reachability graph $R$ of the plant model, the set $FM$ of specified forbidden markings, the set $MS$ of final markings and the set $T_u$ of uncontrollable transitions. In each iteration of the main loop, we identify forbidden markings. These markings and their input/output arcs are then removed from the graph. After that, we qualify as forbidden the markings which are not reachable from the initial marking. Further, non coreachable nodes are qualified as forbidden. The loop terminates when all forbidden markings are processed.

**input** : $R$ a reachability graph
$FM$ is the set of initially specified forbidden markings
$MS$ is the set of final markings
$T_u$ is the set of uncontrollable transitions
**output**: $R_c$ the admissibility graph ; $\Omega$ the set of state-transitions

$DM \leftarrow \emptyset; TE \leftarrow \emptyset; \Omega \leftarrow \emptyset; R_c \leftarrow R$
**repeat**
  Take a non coloured element $f$ from $FM$
  Colour $f$ in $FM$
  **for** *every input arc* $(x,(t,c),f)$ *of* $f$ **do**
    **if** $t \in T_u$ **then**
      $FM \leftarrow FM \cup \{x\}$
    **else**
      $DM \leftarrow DM \cup \{x\}$ ; $TE \leftarrow TE \cup \{(x,(t,c))\}$
  Remove $f$, the input and output arcs of $f$ from $R_c$
  **for** *every node M of* $R_c$ **do**
    **if** *M is not reachable from* $M_0$ *or M is not coreachable or (M has no output arcs and* $M \notin MS$*)* **then** $FM \leftarrow FM \cup \{M\}$
  **if** $M_0 \in FM$ **then** exit //there is no solution
**until** *all elements of FM are coloured* ;
$DM = DM \setminus FM$
**for** *every element y of DM* **do**
  **for** *any element* $(x,(t,c))$ *of TE* **do**
    **if** $y == x$ **then** $\Omega \leftarrow \Omega \cup \{(x,(t,c))\}$

**Algorithm 1**: Computing the admissibility graph

Let us apply this algorithm to our problem of producer-consumer such that $C1 = \{pr1, pr2\}$, $C2 = \{o, o\}$ and $C3 = \{co\}$. The reachability graph of this problem is given by Fig. 3. Let $M = (p1, p2, p3, p4, p5)$ be the structure of the marking vector. Assuming that $M8 = (pr1, pr2, 0, o, co)$ and $M9 = (pr2, pr1, 0, o, co)$ are the specified forbidden markings. Applying Algorithm 1, we obtain as results the admissibility graph described in Fig. 4 and the set

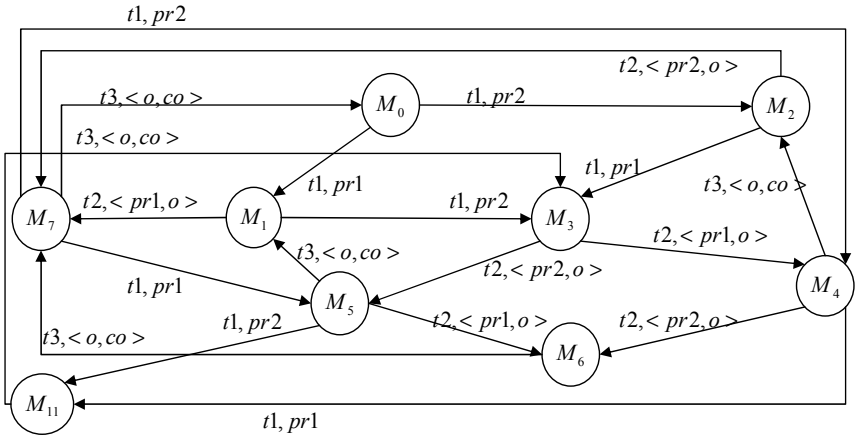



Fig. 4. Admissibility graph of the producer-consumer problem

of state-transitions $\Omega = \{(M6,(t1,pr1)),(M6,(t1,pr2)),(M11,(t2,< pr1,o >)),(M11,(t2,< pr2,o >))\}$.

### 4.2 Synthesis of the controller

The controller synthesis consists in solving numerous systems similar to those of type (1), (2) and (3) formulated from the admissibility graph. The solutions of obtained systems allow to determine the necessary parameters of the controller which is expressed in term of CP-nets. Thanks to the expressiveness power of CP-nets, the controller is reduced to one single place. Therefore, the necessary parameters allowing to build the controller (a CP-net place) $p_c$ and achieving its connection to the plant model are its colour domain $C_{pc}$, its initial marking $M_0(p_c)$ and its incidence vector $W(p_c,.)$.
First, we propose to reformulate the equations/inequations of type (1), (2) and (3) in order to deal with CP-nets instead of ordinary Petri nets. Finally, we give the algorithm allowing the synthesis of a controller for a plant CP-net model.

Let us consider the admissibility graph $R_c$ of a plant CP-net model, and the set $\Omega$ of state-transitions, and $p_c$ be the controller CP-net place to determine. We denote its colour domain by $C_{pc}$. Each object of $C_{pc}$ is related to one or several elements of $\Omega$. As it is stated by the theory of regions, the controller place $p_c$ must satisfy the reachability condition. This condition guarantees that every marking in the admissible behaviours remains reachable under control. Let $M$ be a marking of the admissibility graph $R_c$. The reachability condition related to $M$ is:

$$\forall v \in C_{pc}, M(p_c)(v) = M_0(p_c)(v) + W(p_c,.)(v)\,\overrightarrow{\Gamma}_M \geq 0 \tag{4}$$

where $\overrightarrow{\Gamma}_M$ is the vector counting of any non oriented path $\Gamma_M$ in $R_c$ from $M_0$ to $M$.
For instance, we consider the marking $M_7$ of the admissible graph in Fig. 4. For every $v \in C_{pc}$, the related reachability condition to $M_7$ is: $M_7(p_c)(v) =$

$M_0(p_c)(v) + W(p_c,t1)(pr1)(v) + W(p_c,t2)(< pr1, o >)(v)$.

Cycle equations ensure that the cycles of $R_c$ must remain reachable under control. Hence, the place $p_c$ has to satisfy the following equation:

$$\forall v \in C_{pc}, \forall \gamma \in S_c, W(p_c,.)(v)\overrightarrow{\gamma} = 0 \tag{5}$$

where $S_c$ denotes the set of cycles of $R_c$ and $\overrightarrow{\gamma}$ is the vector counting of the cycle $\gamma$.
In the admissibility graph of Fig. 4, the cycle equation related to the oriented cycle $(M_0, M_1, M_7)$ is expressed by the equation $\forall v \in C_{pc}, W(p_c,t1)(pr1)(v) + W(p_c,t2)(< pr1, o >)(v) + W(p_c,t3)(< o, co >)(v) = 0$.

Finally, a event separation equation associated with an element $(M,(t,c))$ of $\Omega$ allows the controller $p_c$ to prevent the firing of $t$ in $M$ with colour $c$:

$$\forall v \in C_{pc}, M_0(p_c)(v) + W(p_c,.)(v)\overrightarrow{\Gamma}_M + W(p_c,t)(c)(v) < 0 \tag{6}$$

As an example, the disabling of the event separation event $(M6, t1, pr1)$ is ensured through the inequation $\forall v \in C_{cp}, M_0(p_c)(v) + W(p_c,t1)(pr1)(v) + W(p_c,t1)(pr2)(v) + W(p_c,t2)(< pr2, o >)(v) + W(p_c,t2)(< pr1, o >)(v) + W(p_c,t1)(pr1)(v) < 0$.

Our proposed Algorithm 2 builds, in an incremental manner, the controller components. In each iteration of the algorithm, it solves one system where a new object of $C_{pc}$ is introduced as an unknown factor, and a new element of $\Omega$ is considered in (6).
Then, we have to *fold the partial solution* $W(p_c,t_i)(c)(v_j)$ in order to determine $W(p_c,.)(v_j)$. Indeed, the colour domain $C(t_i)$ is partitioned into $k_i$ sets $E_i^s$ ($s = 1..k_i$) such that: $c', c'' \in E_i^s$, iff $W(p_c,t_i)(c')(v_j) = W(p_c,t_i)(c'')(v_j) = \alpha_i^s$. Hence, for every transition $t_i$, the colour functions associated with the same set $E_i^s$ are grouped. Thus, the folding is achieved as follows:

$$W(p_c,t_i)(v_j) = \sum_{s=1}^{k_i} [\bigvee_{c' \in E_i^s} [\bigwedge_{X \in var(t_i)} (X = X(c'))].\alpha_i^s] \tag{7}$$

**input** : $R_c$ is the admissibility graph
$\Omega$ the set of event separation instances
**output**: $M_0(p_c)$ and $W(p_c,.)$

$j \leftarrow 0$
Compute the basis cycles of $R_c$
Generate the reachability conditions (4)
Compute the independent cycle equations (5)
**repeat**
  $j \leftarrow j+1$
  Let $v_j$ be new object Let $(M,(t,c)) \in \Omega$
  $\Omega = \Omega \setminus \{(M,(t,c))\}$
  Generate the event separation condition (6) for $(M,(t,c))$
  Solve the system made up of (4), (5) and (6) after replacing $v$ by $v_j$.
  **if** *there is no solution* **then**
    the algorithm terminates as the legal behaviour can not be enforced
  **else**
    Remove from $\Omega$ the elements having the same solution
    **for** *every transition* $t_i \in T$ **do** Fold the solution
**until** $\Omega = \emptyset$ ;

**Algorithm 2**: Synthesis of the controller

Applying Algorithm 2 on the basis of admissibility graph of Fig. 4 and the set $\Omega = \{(M6,(t1,pr1)),(M6,(t1,pr2)),(M11,(t2,< pr1,o >)),(M11,(t2,< pr2,o >))\}$ of state-transitions, we obtain the controller place $p_c$ having as colour domain the set $C_{pc} = \{v1\}$. The controlled model is illustrated by Fig. 5. As an example showing the operation of folding, we consider the partial solutions $W(p_c,t1)(v1)(pr1)$ and $W(p_c,t1)(v1)(pr1)$. Indeed, we have $W(p_c,t1)(v1)(pr1) = W(p_c,t1)(v1)(pr2) = 1$. Consequently, the folding of these two partial solutions according to (7) gives $W(p_c,t1)(v1) = 1[X = pr1 \vee X = pr2]$.

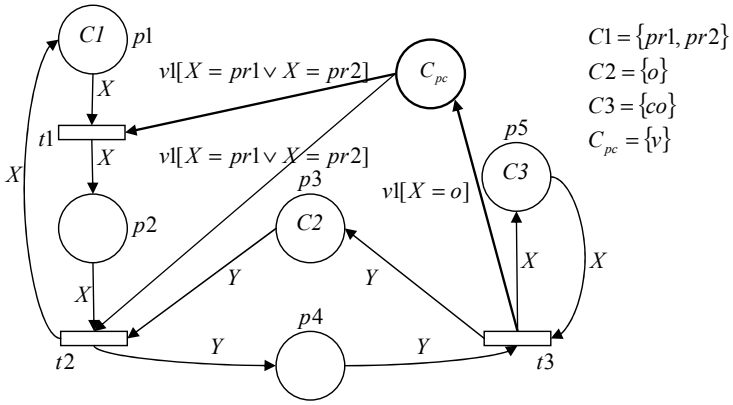



Fig. 5. Controlled model

## 5. Optimised controller synthesis for symmetric systems

In practice, reachability graphs obtained from CP-nets are often huge, thereby it becomes sometimes impossible to perform the synthesis process. As an attractive solution to alleviate this issue is the use of *Symbolic reachability graphs* (SRGs) instead of ordinary reachability graphs. Indeed, an SRG allows the construction of a reduced representation of the ordinary state space without unfolding of colour sets. Experiments (Daws & Tripakis, 1998) have proven that the size of the symbolic reachability graph is quite small in practice. In addition, the building of an SRG is performed automatically from the structure of a WF-net by exploiting its *behaviour symmetries*.
Following the same steps of the approach described in the previous section, we propose to optimise the controller synthesis for symmetric DES. The optimisation is achieved mainly by applying the theory of regions on the basis of symbolic reachability graphs instead of ordinary reachability graphs. Given a CP-net as a plant model, we build its SRG. Then, a treatment is required in order to produce a unique representation for the arcs of the SRG. From the obtained SRG, we determine the graph modelling the desired behaviours according to the control specifications. Finally, the theory of regions is applied on the basis of the latter graph in order to derive the controller, which is represented by a single place expressed in terms of CP-nets.

### 5.1 Symbolic reachability graphs

The symbolic reachability graph of WF-net is based on the idea of *symmetry* of objects of the basic colour classes. Intuitively, a behaviour symmetry is the fact to do *not distinguish* the identities of colours that potentially have the same evolving.
For instance, if we consider the CP-net modelling the dining philosophers problem (Chiola et al., 1997), it is not necessary to distinguish the identities of philosophers. Indeed, for *any* philosopher, the associated structural behaviour may be expressed in terms of synchronisation with its right and left neighbours. In this case, philosophers have symmetrical behaviours. In many other classical problems, behaviour symmetries may be obtained from colours representing processes (clients, servers,...) or resources.
A symbolic marking (SM), a node of a SRG, is a marking, the colours of which are gathered into equivalence classes, forgetting the identity of colours but keeping the cardinality of each represented equivalence class. The SMs are constructed using symmetries that are computed without building the ordinary reachability graph.
In well-formed nets, due the restricted operators defined on object classes, it has been proved that symmetrical colours in a given marking cause the same behaviour. The colours which have structurally similar behaviour, i.e. that can be exchanged at any point in the evolution of the system with no impact on the sequences of firable transitions, are grouped into *static subclasses*, which are not modified during the construction.
Let us consider that any $C_i$ class of objects is partitioned into $n_i$ static subclasses:

$$\forall i \in \{1, ..., n\}, C_i = \bigcup_{q=1}^{n_i} D_{i,q}$$

For instance, in our considered problem of producer-consumer, all producers behave symmetrically, thus $C_1$ corresponds to one static subclass denoted by $D_{1,1}$. Similarly, $C_2$ corresponds to one static class denoted by $D_{2,1}$, and $C_3$ corresponds to a static subclass denoted by $D_{3,1}$.

In contrast, a *dynamic subclass* is a subset of a static subclass. It groups colours having the same distribution throughout the places of the WF-net. A dynamic subclass is characterised by its cardinality and by the static subclass to which it belongs. Although the number and cardinality of these dynamic subclasses evolve during the SRG construction, dynamic subclasses always constitute a partition of static subclasses (the producers that are working and those that are waiting for instance). Thus dynamic subclasses concisely represent the permutations that are permitted on an SM without modifying future sequences of firable transitions.
Now, we give the formal definition of an SM. Intuitively, an SM is expressed by a product of dynamic subclasses.

**Definition 2.** *Let $I = \{1, .., n\}$ be the set of class indexes. A symbolic marking $\mathcal{M}$ is a 4-tuple $R =< m, card, d, marq >$ satisfying:*

- $m : I \longrightarrow N$, *such that $m(i)$ (denoted also $m_i$) is the number of dynamic subclasses of $C_i$ in $\mathcal{M}$. The set of dynamic subclasses of $C_i$ in $\mathcal{M}$ is $\widehat{C}_i = \{Z_i^j | 0 < j \leq m_i\}$,*
- $card : (\bigcup_{i \in I} \widehat{C}_i) \longrightarrow \mathbb{N}$,
- $d : (\bigcup_{i \in I} \widehat{C}_i) \longrightarrow \mathbb{N}$ *such that:*
  1. *$d(Z_i^j)$ is the index $q$ of a static subclass $D_{i,q}$,*
  2. *$\forall i \in I, \forall j, k$ s.t. $0 < i \leq n \wedge 0 < j < k \leq m_i, d(Z_i^j) \leq d(Z_i^k)$*
- *$\forall p \in P, marq(p) : \otimes_{i \in I}(\widehat{C}_i)^{e_i} \longrightarrow \mathbb{N}$ where $e_i$ represents the number of occurrences of $C_i$ in $C(p)$, and $\otimes$ denotes the Cartesian product.*

*Moreover, R must satisfy: $\forall M \in \mathcal{M}, \forall i \in I, \exists \psi_i : C_i \longrightarrow \widehat{C}_i$ such that:*

1. $|\psi_i^{-1}(Z_i^j)| = card(Z_i^j)$
2. $\forall i \in I, \exists D_{i,q}$ *such that* $\psi_i^{-1}(Z_i^j) \subseteq D_{i,q}$ *and* $q = d(Z_i^j)$
3. $\forall p \in P, \forall c \in C(p), M(p, \otimes_{i \in I} \otimes_{j=1}^{e_i} c_i^j) = M(p)(\otimes_{i \in I} \otimes_{j=1}^{e_i} \psi_i(c_i^j))$

In order to illustrate an example of an SM, we consider the SRG of the producer-consumer given in Fig. 6. Let us consider the initial SM $\mathcal{M}_0 = (Z_1^1, 0, Z_2^1, 0, Z_3^1)$ of this SRG. This SM is expressed by the dynamic subclass $Z_1^1$ in place $p1$, the dynamic subclass $Z_2^1$ in place $p3$ and $Z_3^1$ in place $p_5$. Since the cardinality of $Z_1^1$ is $|Z_1^1| = 2$, and $d(Z_1^1) = 1$ ($Z_1^1$ is a subset of $D_{1,1}$), then $Z_1^1$ represents all the elements of $D_{1,1}$, namely the two tokens $pr1$ and $pr2$. Following the same reasoning for $Z_2^1$, we conclude that it represents all the elements of the second static subclass $D_{2,1}$. Also, $Z_3^1$ represents the elements of $D_{3,1}$.
In order to build directly a new SM from a current one, the classical notion of a transition instance is replaced by the notion of *symbolic instance*. It corresponds to a splitting of the dynamic subclasses of the current SM in order to isolate quantities of colours that can be used for the *symbolic firing*. Indeed, in a symbolic firing, instance dynamic subclasses are assigned to the transition parameters instead of objects. When an instance dynamic subclass is assigned to a parameter, it means that any object in the subclass can be assigned to the parameter.
Let $I = \{1, ..., n\}$ be the set of class indexes. Let $t$ be a transition, the colour domain of which is $C(t) = \bigotimes_{i=1}^{n} \bigotimes_{j=1}^{e_i} C_i$. Let $\mathcal{M}$ be a symbolic marking and $\mathcal{R}$ a symbolic representation of $\mathcal{M}$.
We say that $(\bigotimes_{i=1}^{n} \bigotimes_{j=1}^{e_i} Z_i^{\lambda_i(j), \mu_i(j)})$ is a *symbolic instance* for $t$ wrt. $\mathcal{R}$, if and only if:
$\lambda = \{\lambda_i : \{1, ..., e_i\} \to \mathbb{N}^*\}$ and $\mu = \{\mu_i : \{1, ..., e_i\} \to \mathbb{N}^*\}$ such that $\forall i \in I, \forall x \in \{1, ..., e_i\}$,

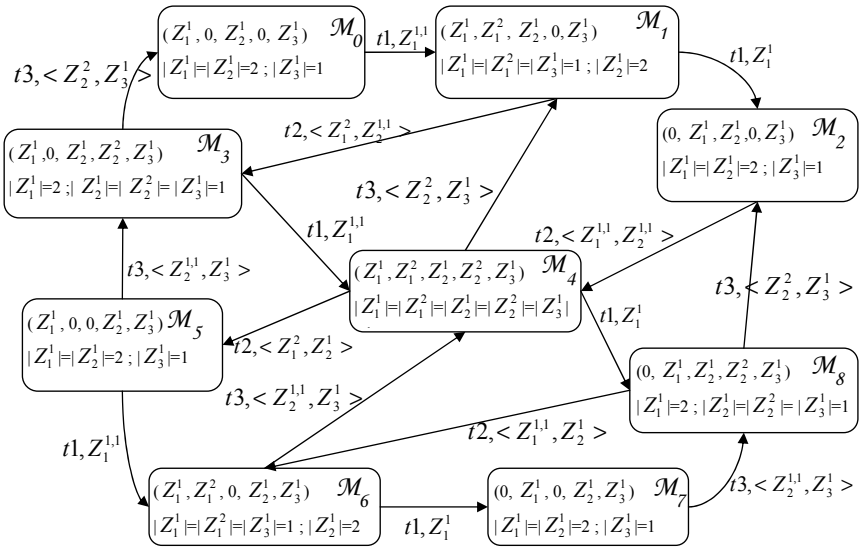



Fig. 6. Symbolic reachability graph of the producer-consumer problem

- $\lambda_i(x) \leq \mathcal{R}.m(i)$ ,
- $\mu(i) \leq \mathcal{R}.card(Z_i^{\lambda_i(x)})$
- $\forall k \in \mathbb{N}^*$ s.t. $k < \mu_i(x), \exists x' < x$ s.t. $\lambda_i(x') = \lambda_i(x) \wedge \mu_i(x') = k$

If $e_i = 0$, we do not define $\lambda_i$ and $\mu_i$.

A symbolic instance for a transition is a product of dynamic subclasses. A dynamic subclass may occur several times: if some $\mu_i$ values are equal, with respect to the same dynamic subclasses, then the same object is referred. In practice, one can note that $\lambda_i$ and $\mu_i$ are specified only if necessary.
As an example for a symbolic firing, we consider the SRG of fig 6. In SM $\mathcal{M}_0$, two producers are in place $p1$, namely $pr1$ and $pr2$. Since $X$ may be bound to any producer, then two coloured firings are possibles: the firing of transition $t1$ with colour $pr1$ or with colour $pr2$. All elements within a dynamic subclass $Z_i^j$ are fully equivalent, then there is only one way to bind a variable to any $Z_i^j$, whatever its cardinality. Hence, a single symbolic binding is possible from the SM $\mathcal{M}_0$, $X$ is bound to (a value in) $Z_1^1$. These possible firings are represented by one symbolic firing inducing an arc labelled with $(t1, Z_1^{1,1})$ such that $Z_1^{1,1}$ represents any object of $Z_1^1$. The new induced SM by the firing of $t1$ in $\mathcal{M}_0$ is $\mathcal{M}_1 = p1(Z_1^1) + p2(Z_1^2) + p3(Z_2^1) + p5(Z_3^1)$ such that $|Z_1^1| = |Z_1^2| = |Z_3^1| = 1$ and $|Z_2^1| = 2$. $Z_1^1$ represents an object of $D_{1,1}$, while $Z_1^2$ represents another object of the same static subclass. $Z_2^1$ represents the two objects of $D_{2,1}$.

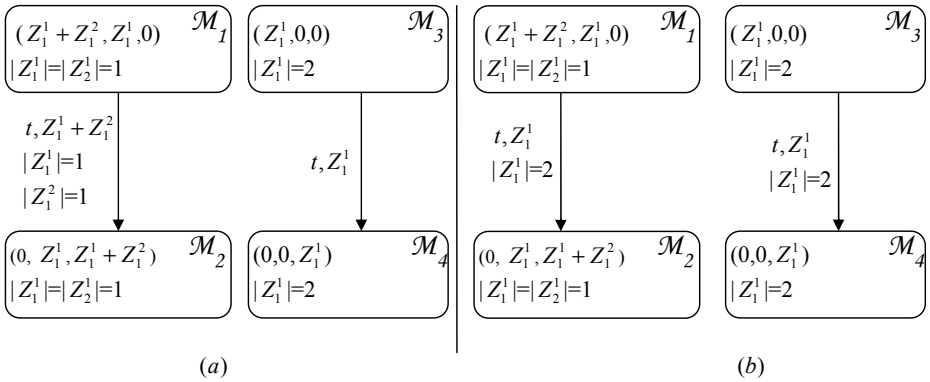



Fig. 7. Canonisation of SRG arcs

### 5.2 Canonisation of arcs

The parameters of the controller are determined by resolving linear system equations formulated from the admissibility graph. Most terms of the formulated equations are determined from the arcs of the admissibility graph. When dealing with SRGs, the main issue is that the same symbolic firing instances can be encoded differently in the graph.

A symbolic firing of a transition $t$ in an SM $\mathcal{M}$ indicates the removed objects from the input places of $t$ in the corresponding arc of the SRG. The removed objects are represented by a product of dynamic subclasses. These dynamic subclasses correspond to partial instances of dynamic subclasses of $\mathcal{M}$. Therefore, the same objects may not have the same representation in arcs. As a consequence, the comparison of two symbolic firings related to the same transition becomes more complex, since it requires the computation of represented objects by the dynamic subclasses labelling the arcs, and then one has to check whether the sets of objects are the same, or not. In order to simplify this operation of comparison, we propose to define a canonical representation of dynamic subclasses labelling arcs induced by symbolic firings. By this way, we guarantee an unique representation for the labels of arcs. This unique representation allows to compare two arcs, induced by the symbolic firing of the same transition, by an equality test between the dynamic subclasses labelling the arcs. Such comparison is performed according to the following three criterions:

- they have the same labels,
- they belong to the same static subclass,
- they have the same cardinalities.

Similarly to the canonisation of SMs, we exploit two properties on dynamic subclasses, called minimality and ordering in order to canonise the representations of arcs.

Given an arc labelled with $(t, \widehat{c})$. $\widehat{c}$ is a tuple indexed by the input places of $t$, where each component is a product of dynamic subclasses specifying the objects moved from the input places of $t$. The same algorithms used for the canonisation of SMs can be applied for arcs. Here, we will not present in detail these algorithms, since they can be found in (Chiola et al., 1991). We just note that this computation is organized in two steps:

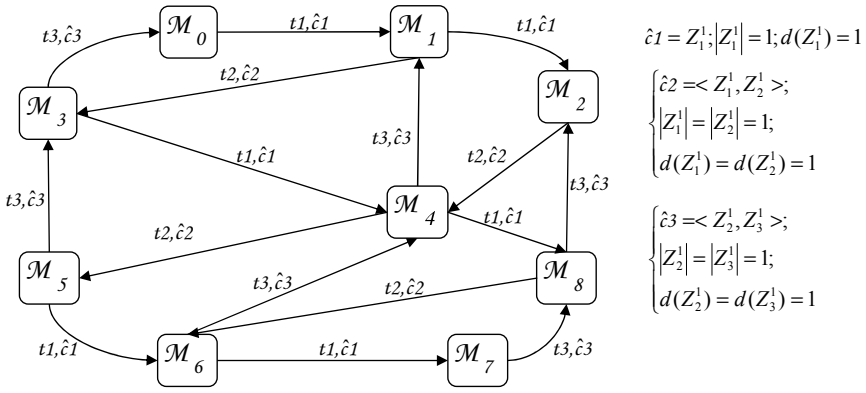


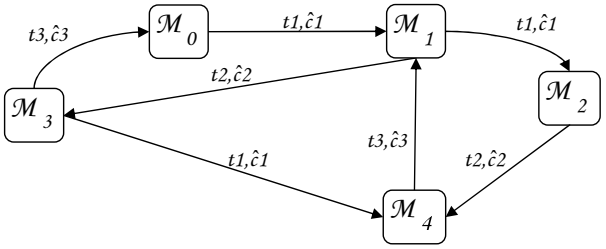

Fig. 8. SRG of consumer-producer problem with canonised arcs

- the computation of a minimal representation. In this step, we compute the smallest number of the dynamic subclasses. Thus, two dynamic subclasses having the same distribution in $\hat{c}$ are grouped into one dynamic subclass.
- the search of an ordered representation. It consists of readjusting the dynamic subclasses indexes according to an ordering criterion. The ordering criterion has to be defined in a such way that it will set a unique indexing scheme for dynamic subclasses, i.e. it has to be a total order. Here, we choose the lexicographic order on the input places associated with the considered transition.

Figure 7*(a)* gives a subgraph of an SRG illustrating the issue related to representation of arcs. Here, the WF-net of the SRG consists of three places $p1$, $p2$ and $p3$. Its object class contains two objects. Although the two symbolic firings of $t$ in SMs $\mathcal{M}_1$ and $\mathcal{M}_3$ are identical, they are represented differently. Indeed, a symbolic firing of transition $t$ is performed by removing two objects from its input place $p1$ and transferring them into place $p3$. In marking $\mathcal{M}_1$ the two objects in place $p1$ are represented by the sum $Z_1^1 + Z_1^2$. Consequently, the firing of $t$ in $\mathcal{M}_1$ induces an arc labelled with $(t, Z_1^1 + Z_1^2)$. In marking $\mathcal{M}_3$ the two objects in place $p1$ are represented by one dynamic subclass $Z_1^1$ such that $|Z_1^1| = 2$. Firing of $t$ in $\mathcal{M}_3$ induces an arc labelled with $(t, Z_1^1)$. Performing the canonisation of arcs, we obtain the same label for the two arcs as it is illustrated by Fig. 7*(b)*. Now let us consider the SRG of Fig. 6. Canonising the arcs of this graph gives the graph shown in Fig. 8. The latter graph will be used in order to determine the controlled behaviour respecting the control specification.

### 5.3 Synthesis of controller based on SRGs

Given an SRG with canonised arcs of a plant model, the determination of a controller is ensured through three steps. First, one has to provide the control specifications. According to these specifications, we build, in a second step, a symbolic graph implementing the desired behaviour from the SRG. Finally, in third step, we apply the theory of regions on the basis of the latter graph in order to determine the controller which is represented by one CP-net place.




$$\hat{c}1 = Z_1^1; \left|Z_1^1\right| = 1; d(Z_1^1) = 1$$

$$\begin{cases} \hat{c}2 =< Z_1^1, Z_2^1 >; \\ \left|Z_1^1\right| = \left|Z_2^1\right| = 1; \\ d(Z_1^1) = d(Z_2^1) = 1 \end{cases}$$

$$\begin{cases} \hat{c}3 =< Z_2^1, Z_3^1 >; \\ \left|Z_2^1\right| = \left|Z_3^1\right| = 1; \\ d(Z_2^1) = d(Z_3^1) = 1 \end{cases}$$

Fig. 9. Admissibility symbolic graph

**Step 1 - Control specifications**
In a forbidden state problem, the control specifications are defined by providing a set of undesirable (forbidden) states. The provided states will be used to extract the desired behaviours from the SRG of the plant model. Therefore, the undesired states have to be expressed in terms of SMs corresponding to nodes of the considered SRG. Such specifications fit well with the nature of symmetric systems made up of several components behaving similarly.
We consider the SRG of Fig. 6. As control specifications, we assume that one aims to restrict the stock capacity to one object. Such specifications can be symbolically expressed by providing the undesirable SMs $\mathcal{M}_5$, $\mathcal{M}_6$ and $\mathcal{M}_7$.

**Step 2 - Computing the desired behaviours**
Here, the desired behaviours are implemented by a subgraph of the SRG, called the *admissibility symbolic graph*. Algorithm 1 can be easily adapted to SRGs in order to compute the admissibility symbolic graph from an SRG and the set $\Omega_{SM}$ of SM-transitions. In a SRG, the set $\Omega_{SM}$ is used instead of the set $\Omega$ of state-transitions. Each element of $\Omega_{SM}$ is couple $(\mathcal{M}, (t, \widehat{c}))$ where $\mathcal{M}$ is a *dangereous SM* in which the symbolic firing of transition $(t, \widehat{c})$ must be forbidden, i.e. $t$ must be prevented from firing in $M$ with every colour $c \in \widehat{c}$. The main change to be made for Algorithm 1 is to deal with SMs (resp. symbolic instances) instead of ordinary markings (resp. coloured firings).
For example, let us consider the SRG of Fig. 6 and the control specifications corresponding to the restriction of the stock capacity to one object. Computing the admissibility symbolic graph, we obtain the graph illustrated by Fig. 9 and the set of SM-transitions $\Omega_{SM} = \{(\mathcal{M}_4, (t1, \widehat{c}1)), (\mathcal{M}_8, (t2, \widehat{c}2))\}$.

**Step 3 - Construction of the controller**
In this final step, we construct the controller by applying the theory of regions on the basis of the admissibility symbolic graph. Using CP-nets allows to represent the obtained controller by one CP-net place $p_c$ which will be connected to the plant model. For this, we have to determined its colour domain $C_{pc}$, its initial marking $M_0(p_c)$ and its incidence vector $W_c(p_c, .)$. Using a symbolic graph requires the reformulation of the three classes of equations/inequations (4), (5) and (6) according to the structure of the symbolic graphs. Indeed, in an ordinary reachability graph, these equations are deduced from the basic relation between two reachable markings.

Let $v \in C_{pc}$. Let $M$,$M'$ be two reachable markings. Let $t$ be a transition enabled in $M$ for $c \in C(t)$, such that $M[t,c\rangle M'$, then:
$\forall v \in C(p_c)$,

$$M'_c(p_c)(v) = M_c(p_c)(v) + W_c(p_c,t)(c)(v) \tag{8}$$

where $M'_c(p_c)(v)$ is the number of $v$ in place $p_c$ at the marking $M'$, $M_c(p_c)(v)$ is the number of $v$ in place $p_c$ in $M$ and $W_c(p_c,t)(c)(v)$ is the incidence value associated with place $p_c$ and transition $t$ for the colour $(c)(v)$.
Further, we have $W_c(p_c,t)(c)(v)$=$W_c(p_c,t)(c')(v), \forall c' \in C(t); \widehat{c}' = \widehat{c}$.
Thus, there exists two symbolic markings $\mathcal{M}$ and $\mathcal{M}'$ such that $M \in \mathcal{M}$, $M' \in \mathcal{M}'$ and $\mathcal{M}[[t,\widehat{c}\rangle\rangle\mathcal{M}'$. Thus, the equation (8) can be written as follows:

$$M'_c(p_c)(v) = M_c(p_c)(v) + W_c(p_c,t)(\widehat{c})(v) \tag{9}$$

Therefore, the three classes of equations depicted by the theory of regions can be reformulated for an SRG as follows:

- The reachability conditions (4) indicate that every reachable SM must remain reachable under control. $\forall \mathcal{M} \in R_c$,

$$M(p_c)(v) = M_0(p_c)(v) + W(p_c,.)(v)\,\overrightarrow{\Gamma}_{\mathcal{M}} \geq 0 \tag{10}$$

  where $M$ is an ordinary marking represented by $\mathcal{M}$, $M(p_c)(v)$ is the occurrence of a given object $v$ in the marking of $p_c$, $W(p_c,.)(v)$ is the incidence vector relatively to the object $v$, $\Gamma_{\mathcal{M}}$ is any non oriented path in $R_c$ from $\mathcal{M}_0$ to $\mathcal{M}$, and $\overrightarrow{\Gamma}_{\mathcal{M}}$ is its associated vector. $\overrightarrow{\Gamma}_{\mathcal{M}}$ is indexed by transitions of $T$ and is called the *vector counting* of $\Gamma_{\mathcal{M}}$. Each line of $\overrightarrow{\Gamma}_{\mathcal{M}}$ represents the sum of occurrence number of $(t_i,\widehat{c})$ such that $c \in C(t_i)$ and $(t_i,\widehat{c})$ labels an arc in the considered path. Formally:

$$\overrightarrow{\Gamma}_{\mathcal{M}} = \begin{pmatrix} \Gamma^1_{\mathcal{M}} \\ : \\ \Gamma^m_{\mathcal{M}} \end{pmatrix}$$

  where $\Gamma^i_{\mathcal{M}} = \sum_{(t_i,\widehat{c})\in\Gamma} \alpha_i(\widehat{c}).\widehat{c}\ \forall i \in \{1,...,m\}$ where $\alpha_i(\widehat{c})$ is the occurrence number of $(t_i,\widehat{c})$ in the path $\Gamma_{\mathcal{M}}$ and which is determined from $R_c$.

- the cycle equations (5) indicate that the cycles of the admissibility graph $R_c$ must remain reachable under control. In other words, the place $p_c$ should satisfy (for every object v of $C_{pc}$):

$$W(p_c,.)\,\overrightarrow{\gamma} = 0, \forall \gamma \in S_c \tag{11}$$

  where $S_c$ is the set of cycles of the admissibility graph $R_c$. $\overrightarrow{\gamma}$ is a vector counting of a cycle $\gamma$ defined similarly as $\Gamma_{\mathcal{M}}$. It is worth to note that according to well known results of graph theory, the cycle equations can be reduced to independent cycle equations of basis cycles (Schrijver, 1986).

- an event separation condition (6) indicates that the control must prevent from firing a state transition of $\Omega$. In other words, the place $p_c$ and an occurrence object v of $C_{pc}$, must solve at least one event separation instance $(\mathcal{M},(t,\widehat{c}))$ from $\Omega$:

$$M_0(p_c)(v) + W(p_c,.)(v)\,\overrightarrow{\Gamma}_{\mathcal{M}} + W(p_c,t)(v)(\widehat{c}) < 0 \tag{12}$$

In order to derive a controller, Algorithm 2 can be applied such that we replace respectively the admissibility graph and set of state-transitions by the admissibility symbolic grah and the SM-transitions. In addition, here the folding is performed differently, since each partial solution, obtained for a new object $v_j$ in an iteration, consists of $M_{pc}(v_j)$ and $\forall t_i \in T, \forall \widehat{c} \in \widehat{C}(t_i); W(p_c, t_i)(\widehat{c})(v_j)$ . The aim is to determine for every transition $t_i$ the incidence vector $W(p_c, t_i)(v_j)$. Hence, we propose to achieve the folding of a partial solution through the following two steps for every transition $t_i$:

- We determine for every colour $c$ represented by $\widehat{c} \in \widehat{C}(t_i)$ the colour function $W(p_c, t_i)(c)(v_j)$ which is equal to $W(p_c, t_i)(\widehat{c})(v_j)$,
- We partition the colour domain $C(t_i)$, of a transition $t_i$, into $k_i$ sets $E_i^s$, $(s = 1..k_i)$ such that $c, c' \in E_i^s$, iff $W(p_c, t_i)(c)(vj) = W(p_c, t_i)(c')(v_j) = \alpha_i^s$. Then, the incidence function for the transition $t_i$ is

$$W(p_c, t_i)(v_j) \leftarrow \sum_{s=1}^{k_i} \left[ \bigvee_{c' \in E_i^s} \left[ \bigwedge_{X \in var(t_i)} (X = X(c')) \right] \right] \alpha_i^s$$

  where $var(t_i)$ is the set of variables appearing on arcs that have $t_i$ as source or destination.

Let us consider the admissibility graph shown in Fig. 9 and the set $\Omega_{SM} = \{(\mathcal{M}_4, (t1, \widehat{c}1)), (\mathcal{M}_8, (t2, \widehat{c}2))\}$. The obtained controller consists of one place $p_c$ as it is shown in Fig. 10. One can easily check that the symbolic graph of the controlled model is exactly the admissibility symbolic graph of the plant model.

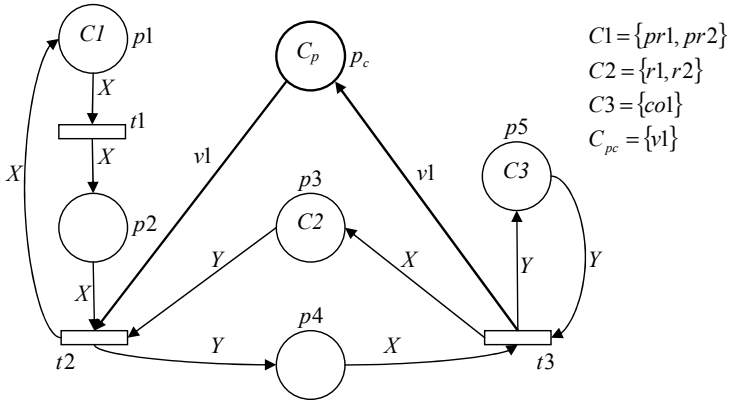



Fig. 10. Controlled model

## 6. Active controller

Considering as a plant model a CP-net that is assumed to be structured on a set of generic processes sharing a set of resources, i.e. a resource allocation system, we present the "Active Controller" approach allowing to simplify the design of a generic CP-net controller for a such

system. In previous sections, we have used the theory of regions in order to derive a controller for a DES modelled by a CP-net. The use of the theory of regions requires the resolution of numerous systems allowing the determination of the necessary parameters of the controller. The ¨Active Controller¨ approach was introduced to avoid solving such systems. The key idea of this approach is that the controller must be able to handle enough information to detect reaching a dangerous state, and from which, it removes appropriate authorisations in order to disable the firing of some transitions.
The major difference, from previous approaches, lies in the synthesis of the controller. In the ¨Active Controller¨ approach, the controller is characterised by a CP-net subnet (with a fixed structure (4 places, 2 transitions)) representing its behaviour. The variable part (basic colour sets, initial marking and some arc expressions connecting the controller CP-net to the original CP-net) of this controller CP-net depends on the specificity of the studied system and it is generated from the control specification. More precisely, it is defined based on the set $\Omega$ of state-transitions. So, the same model can be used in several applications. Thus, the controller synthesis may be considered as parametrable.
The active controller acts as priority process having two states. It is either in a ¨Monitor¨ state in which it observes the evolution of the original network, or in an ¨Alert¨ state from which it inhibits the firing of some transitions. The controller permanently maintain, in a dedicated place, the current marking of the original system. The controller enters the ¨Alarm¨ state, if it detects, based on an appropriate marking of an additional place, that the original system has reached a dangerous state. When entering the ¨Alarm¨ state, the controller removes specific marking from an appropriate additional place to disable the firing of the forbidden transitions associated with this global state. The controller lets the original system to evolve towards an admissible state. Once the overall current state is changed, the controller leaves the ¨Alarm¨ state.
The functioning of the controller is based on the higher priority associated with its transitions. Such a priority enables it to preempt the transitions of the original system.
The ¨Active Controller¨ approach is based on two steps:

- the admissibility computation: This computation is done as explained in section 4.1.
- the construction of the controller: this step allows the generation of the controller CP-net and its connection to the plant CP-net model.

Fig. 11 describes the inputs and the outputs of each step. The following section introduces the controller construction method. The determination of the admissible behaviour has already been presented in section 4.1.

### 6.1 Controller construction method

At the beginning of this step, we assume that we have already generated the set $\Omega$ of state-transitions by Algorithm 1. Based on this information, we generate the controller, modelled by a CP-net, and we define its connection to the plant CP-net model. The synthesis of a such controller may be automatised. It is worth noting that the generated CP-net is autonomous and does not require external devices to ensure the control.
At this level, we handle two kinds of specifications:

- the control specification. More precisely:
  - the set of dangerous markings (denoted DM),

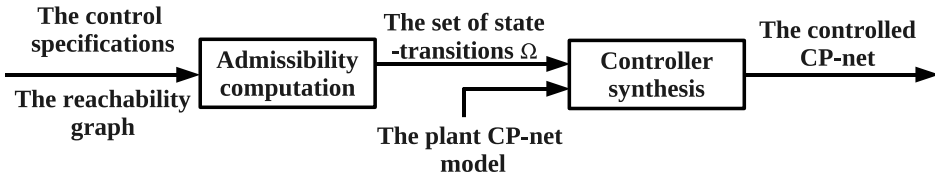



Fig. 11. ``Active Controller'' steps

– the set of forbidden state-transitions associated with each dangerous marking d $\in$ DM (denoted FT(d)). FT(d) may be viewed as an application from DM to the set of subsets of T.

- the uncontrolled system specification represented by a CP-net $N$ such that: $N =< P, T, C_l, C, W^-, W^+, \Phi, M_0 >$.

The output of this step is a new CP-net such that its functioning automatically satisfies the control specifications. As previously explained, the key idea of the ``Active Controller'' is that the controller must handle enough information to detect the reaching of a dangerous state. Further, in a dangerous marking, we remove appropriate authorisations by disabling the firing of some coloured transitions, called *forbidden coloured transitions*. This method relies on the following points:

- The information related to dangerous states and their associated forbidden transitions are defined by the initial marking of a specific place. This marking is computed on the basis of the set $\Omega$.
- The current state (marking) of the plant model is handled in a special added place. This information is modelled by a composed token (tuple of colours) in accordance with the CP-nets semantics.
- An authorisation is associated with each coloured forbidden transition to enable its firing. All authorisations are managed in a third added place.
- The generated controller CP-net has two supplementary transitions. The first one is fired when a dangerous state is detected, while the second is fired when the dangerous state is quitted. These two controller transitions must be immediately fired when enabled to remove or replace the appropriate authorisations. Thus, they must have the highest priority over all the other transitions.
- The necessary additional CP-net components (colour functions, synchronisation arcs, markings, etc.) must be also defined to ensure the desired management of the controller.

In the following, we formally detail the generation of the controlled CP-net model.
The model representing the system under control is a CP-net $N^*$ obtained from $N$ so that $N^* =< P^*, T^*, {C_l}^*, C^*, {W^*}^-, {W^*}^+, \Phi^*, {M^*}_0 >$ where:
$P^* = P \cup \{$CM, DaM, AT, AS$\}$, with:

CM representing the Current Marking,
DaM representing the Dangerous Markings,
AT representing the Authorisations for forbidden Transitions, and
AS representing the Alert State of the controller.

$T^* = T \cup \{\text{A-In, A-Out}\}$, with:

A-In representing entering the alert state,
A-Out representing quitting the alert state,
A-In and A-Out have the highest priority.

$C_l{}^* = C_l \cup \{C_{num}, C_{FT}\}$, with:

$C_{num} = \{0, 1, 2, , MaxInt\}$ is a class representing a set of finite positive integers. Its elements will model the occurrence of some given tokens. We assume MaxInt large enough to be greater than the bound of the maximum occurrences of any token in a reachable marking. As, we deal with bounded CP-nets, this property holds,

$C_{FT}$ is a class representing all coloured forbidden transitions. The different element of this class will be defined based on the application FT.

Each of the added controller place is characterised by a specific colour and by an initial marking. Those parameters are generated as follows:

Place CM has a complex colour domain that is a Cartesian product of $C_{num}$ performed on the basis of the number of process classes, the number of places per process and the resource class. The role of CM is to handle information about the current state of the controlled system.
The token marking of CM is a long tuple made up of counters where each one holds the information about the occurrence of tokens in a given place (according to the lexical order) among process places and the occurrence of tokens in the resource place. The colour domain of CM will be defined as follows:

$$C^*(\text{CM}) = \bigotimes_{i=1}^{|C_R|} C_{num} \bigotimes_{j=1}^{|C_P|} \bigotimes_{k=1}^{|NbP(P_j)|} C_{num}$$

Where:
* $C_R$ represents the resource class,
* $C_P$ represents the different types of processes,
* $NbP(P_j)$ defines the number of places associated with the process type $P_j$. CM is always mono-marked and its initial marking $M_0^*$(CM) is performed on the basis of the initial marking of the CP-net associated with the studied plant. $M_0^*$(CM) may be algorithmically determined.

The colour domain of the place DaM is:

$$C^*(\text{DaM}) = C^*(\text{CM}) \times C_{FT}$$

The initial marking of DaM is not updated, since this place is only read accessed. The number of tokens in DaM is equal to $\sum_{d \in DM} |FT(d)|$

The colour domain and the initial marking of the place AT are: $C^*$(AT)= $C_{FT}$ , and $M_0^*$(AT) = $C_{FT}$. Indeed, initially, all forbidden transitions are authorised.

The colour domain of the place AS is: $C^*$(AS)= $C^*$(CM). Initially this place is empty $M_0^*$(AS) = $\varnothing$.

Finally, the colour functions of the arcs connecting the controller places to a subset of $T$ and to the transitions A-In and A-Out must be defined.

As the role of CM is to hold the current marking of N, it is connected to every transition of T using an input arc (reading marking) and an output arc (updating marking).

$\forall t \in T, W^{*-}(\text{CM,t}) = < X_{1,1}, ..., X_{k,xk}, Y_1, ..., Y_u > = < X >$
where $X_{i,j}$ is a variable defined on $C_{num}$ computing the number of tokens in the place i of the process type j and $Y_u$ is a variable defined on $C_{num}$ reading the occurrence of colour u in the resource places;

$\forall t \in T, W^{*+}(\text{CM,t}) = < X'_{1,1}, ..., X'_{k,xk}, Y'_1, ..., Y'_u > = < X' >$
where $X'_{i,j}$ and $Y'_u$ are variables defined on $C_{num}$ and determined as follows :

$X'_{i,j} = X_{i,j} - \chi$, with $\chi = W^+(p_{ij}, t) - W^-(p_{ij}, t)$,

and

$Y'_u = Y'_u - \xi$, where $\xi$ is computed as follows:

$$\left.\begin{array}{l} W^-(r,t) = \sum_i \alpha_i . r_i \\ W^+(r,t) = \sum_i \alpha'_i . r_i \end{array}\right\} \Rightarrow \xi = \alpha'_u - \alpha_u$$

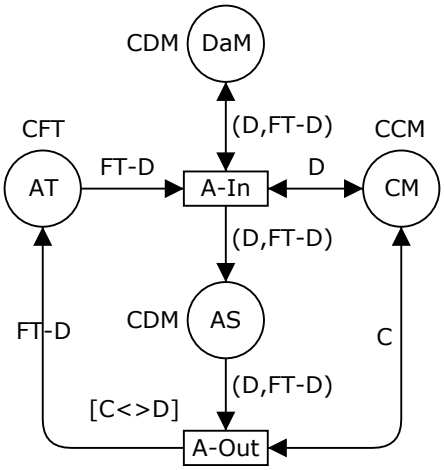



Fig. 12. The ¨Active Controller´´ subnet

The place AT is connected to every forbidden transition by one input/output arc in order to check the presence of the associated firing authorisation:

$\forall t \in C_{FT}$, $W^{*+}$(AT, t)= $W^{*-}$(AT, t)= $< Xt >$ ,

where $Xt$ is defined on $C_{FT}$ and represent the identity of the coloured forbidden transition.

The colour functions of the arcs connecting the controller places to the transition A-In and A-Out are defined as follows:

$W^{*-}$(DaM, A-In)=$W^{*+}$(DaM, A-In)=$< D, FT\text{-}D >$;
$W^{*-}$(CM, A-In)=$W^{*+}$(CM, A-In)=$< D >$;
$W^{*-}$(AT, A-In)=$< FT\text{-}D >$; $W^{*+}$(AS, A-In)=$< D, FT\text{-}D >$;
$W^{*+}$(AS, A-Out)=$< D, FT\text{-}D >$; $W^{*-}$(CM, A-Out) =$< C >$;
$W^{*+}$(AT, A-Out)=$< FT\text{-}D >$;

$D$ and $C \in$ C(CM) (i.e. they are a tuple of variables) , $FT\text{-}D \in C_{FT}$
Transition A-Out is associated with the predicate: $[C \neq D]$

Fig. 12 represents the CP-net modelling the controller behaviour. It is worth to note that the controller CP-net is connected to the plant CP-net model through the places AT and CM as it was previously defined.
In the next section, we apply the generation of the controller to our problem of producer-consumer.

### 6.2 Example

We consider the previously introduced producer-consumer problem modelled by the WF-net of Fig. 2. We assume that $C1 = \{2pr\}$, $C2 = \{o1, o2\}$ and $C3 = \{co\}$. The reachability graph of this problem is given by Fig. 13.

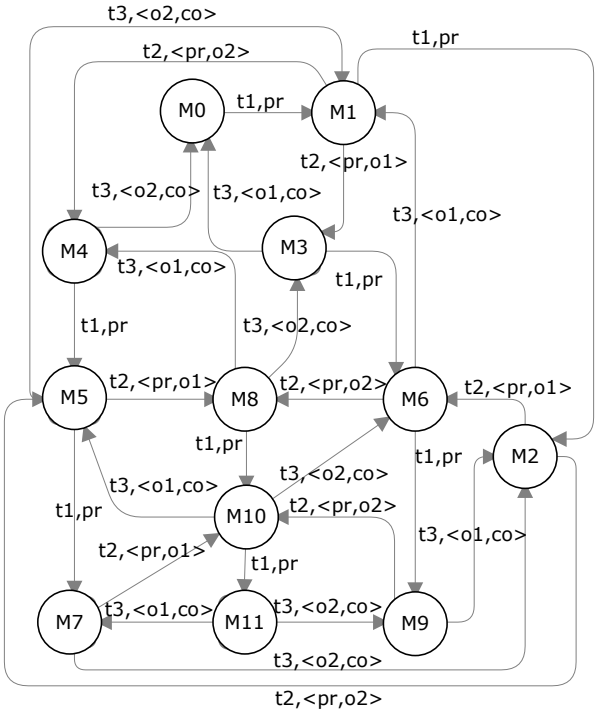



Fig. 13. The reachability graph

Assuming that the stock must contain at most one object, such a specification induces that $M8$, $M10$ and $M11$ are the forbidden markings. Applying Algorithm 1, we obtain the admissibility graph described in Fig. 14 and the set of state-transitions $\Omega = \{(M5, (t2, < pr, o1 >)), (M6, (t2, < pr, o2 >)),$
$(M7, (t2, < pr, o1 >)), (M9, (t2, < pr, o2 >))\}$
Let us illustrate the different variable elements of the Active Controller associated with the considered producer-consumer system.

$C^*$(CM)= $C_{num} \times C_{num} \times C_{num} \times C_{num} \times C_{num}$
The first and the second elements respectively define the number of producers in places $p1$ and $p2$. The third element computes the number of consumers in place $p5$. The fourth (respectively the fifth element) handles the number of objects of type $o1$ (respectively $o2$) in place $p4$.

$C_{FT}$={$t2PrO_1$, $t2PrO_2$}
$t2PrO_1$ (respectively $t2PrO_2$) represents the authorisation to fire transition $t2$ with colour $< pr, o_1 >$ (respectively $< pr, o_2 >$).

$M_0^*$(CM)=<2,0,1,0,0>
$M_0^*$(DaM)=<<1,1,1,0,1>,$t2PrO_1 >$ + <<1,1,1,1,0>,$t2PrO_2 >$ + <<0,2,1,0,1>,$t2PrO_1 >$

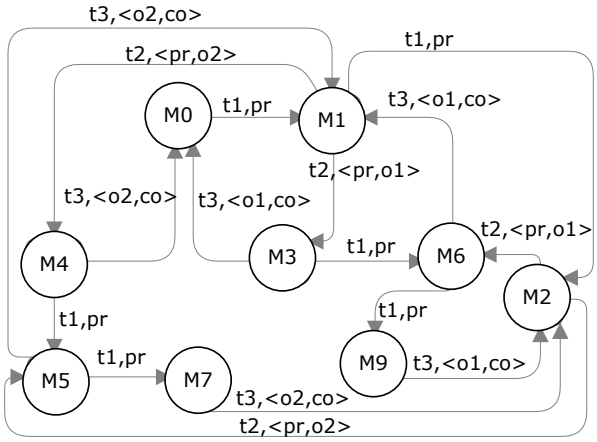



Fig. 14. The admissibility graph

$+ <<0,2,1,1,0>, t2PrO_2 >$
$M_0^*$(AT)= $t2PrO_1 + t2PrO_2$
$\forall t \in T$, $W^{*-}$(CM,t)=$< X_{1,1}, X_{1,2}, X_{2,1}, Y_1, Y_2 >$

$W^{*+}$(CM,t1{$pr$})=$< X_{1,1} - 1, X_{1,2} + 1, X_{2,1}, Y_1, Y_2 >$
$W^{*+}$(CM,t2{$< pr, o1 >$})=$< X_{1,1} + 1, X_{1,2} - 1, X_{2,1}, Y_1 + 1, Y_2 >$
$W^{*+}$(CM,t2{$< pr, o2 >$})=$< X_{1,1} + 1, X_{1,2} - 1, X_{2,1}, Y_1, Y_2 + 1 >$
$W^{*+}$(CM,t3{$< o1, co >$})=$< X_{1,1}, X_{1,2}, X_{2,1}, Y_1 - 1, Y_2 >$
$W^{*+}$(CM,t3{$< o2, co >$})=$< X_{1,1}, X_{1,2}, X_{2,1}, Y_1, Y_2 - 1 >$

## 7. Conclusion

In this chapter, we have dealt with the control of DES, modelled by CP-nets, for the problem of forbidden states. The use of CP-nets allows compact models even for large and complex systems. The first approach, based on the theory of regions, can be addressed to any kind of DES modelled by CP-nets. Considering a CP-net as plant model, in a first step of this approach, the graph implementing the desired behaviours is determined from the reachability graph of the considered DES according to the control specifications. Then, the theory of regions is applied in order to design the controller. Thanks to the expressiveness of CP-nets, the obtained controller is represented by one single place. In a second approach, we propose to cope with the combinatorial explosion of state space for symmetric systems. Indeed, the state space of a symmetric system can be represented by a condensed version, the symbolic reachability graph, which is quite smaller. Following similar steps as the first approach, the second approach allows to deal efficiently with symmetric systems. Indeed, the theory of region is applied on the basis of a symbolic reachability graph instead of the ordinary one. Finally, third approach avoids the use of the theory of regions which requires the resolution of numerous linear systems in order to determine the controller. Indeed, the generated controller is an

active process, modelled by a generic CP-net, that permanently observes the plant model to detect the reaching of dangerous states, and then it removes appropriate authorisations.

## 8. References


Abid, C. & Zouari, B. (2008). Synthesis of controllers using symbolic reachability graphs, *Proceedings of 9th International Workshop of Discrete Event Systems (WODES'08)*, Goteborg, pp. 314–321.

Badouel, E., Bernardinello, L. & Darondeau, P. (1995). Polynomial algorithms for the synthesis of bounded nets, *Proceedings of the 6th International Joint Conference CAAP/FASE on Theory and Practice of Software Development*, Vol. 915, Lecture Notes In Computer Science, Aarhus, pp. 364–378.

Chiola, G., Dutheillet, C., Franceschinis, G. & Haddad, S. (1991). On well-formed coloured nets and their symbolic reachability graph, *in* K. Jensen & G. Rozenberg (eds), *High-Level Petri Nets – Theory and Application*, Springer, pp. 373–396.

Chiola, G., Dutheillet, C., Franceschinis, G. & Haddad, S. (1997). A symbolic reachability graph for coloured Petri nets, *Theoretical Computer Science* **176**(1-2): 39–65.

Daws, C. & Tripakis, S. (1998). Model checking of real-time reachability properties using abstractions, *TACAS*, pp. 313–329.

Ghaffari, A., Rezg, N. & Xie, X. (2003). Design of live and maximally permissive petri net controller using the theory of regions, *Proceedings of IEEE Transactions on Robotics and Automation*, Vol. 19, Aarhus, pp. 137–142.

Giua, A. & DiCesare, F. (1994). Petri net structural analysis for supervisory control, *IEEE Transactions on Robotics and Automation* **10**(2): 185–195.

Holloway, L. E., Krogh, B. H. & Giua, A. (1997). A survey of petri net methods for controlled discrete eventsystems, *Discrete Event Dynamic Systems* **7**(2): 151–190.

Jensen, K., Kristensen, L. M. & Wells, L. (2007). Coloured petri nets and cpn tools for modelling and validation of concurrent systems, *Int. J. Softw. Tools Technol. Transf.* **9**(3): 213–254.

Jensen, K. & Rozenberg, G. (1991). *High-Level Petri Nets: Theory and Application*, Springer Verlag.

Makungu, M., Barbeau, M. & St-Denis, R. (1999). Synthesis of controllers of process modeled as coloured petri nets, *Journal Discrete Event Dynamic Systems Theory Applications Kluwer Academic Publishers* **Vol. 9**(No. 2): 147–169.

Ramadge, P. & Wonham, W. (1989). The control of discrete event systems, *Proceedings of IEEE*, Special Issue on Discrete Event Dynamic Systems, pp. 81–98.

Schrijver, A. (1986). *Theory of Linear and Integer Programming*, John Wiley and Sons, NY.

Sreenivas, S. & Sreenivas, R. S. (1997). On the existence of supervisory policies that enforce liveness in discrete event dynamic systems modeled by controlled petri nets, *IEEE Transactions on Automatic Control* **42**: 94–5.

Su, H. Y., Wu, W. M. & Chu, J. (2005). Liveness problem of petri nets. supervisory control theory for discrete event systems, *ACTA AUTOMATICA SINICA* **31**(1): 143–150.

Zouari, B. & Ghedira, K. (2004). Synthesis of controllers using coloured petri nets and theory of regions, *Proceedings of IFAC Workshop on Discrete Event Systems (WODES'04)*, Reims, pp. 231–236.

Zouari, B. & Zairi, S. (2005). Synthesis of active controller for resources allocation systems, *Proceedings of Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools (CPN'05)*, pp. 79–98.

15

# Using Petri Net for Modeling and Analysis of a Encryption Scheme for Wireless Sensor Networks

Hugo Rodríguez, Rubén Carvajal, Beatriz Ontiveros, Ismael Soto
*Universidad de Santiago de Chile*
*Chile*

Rolando Carrasco
*Newcastle University*
*UK*

## 1. Introduction

Nowadays the wireless sensor networks (WSN) are increasingly being required for applications where the data reliability needs to be duly guaranteed. Due to this, many research works have been directed in this sense. However, WSN security is not an entirely solved issue.
The aim on this chapter is to use Petri Nets (PN) to model and analyze the validity of an encryption scheme applied to WSN. Usually the modeling of communication systems can become quite complex, most of the time is necessary to consider a large number of variables and mathematical models to describe them. Nevertheless, structural properties of a system and dynamic characteristics of its behavior can not be properly derived from those models. On the other hand, PN can model dynamic characteristics such as synchronization and concurrency.
PN were developed by Carl A. Petri in 1962 as a mathematical tool for the study of communication with automata. Its further development was facilitated by the fact that PN can be used to model properties such as process synchronization, concurrent operations, conflicts or resource sharing and deadlock freedom among others (Reisig, 1986), (Reisig & Rozenberg, 1998). In literature we found works where PN have been used in order to model and analyze several related aspects of a WSN. In (Chen et al., 2008) the authors use PN to determine the minimal necessary number of devices in a coal mine, in order to establish the location of miners in case of accident. There are many works related to power constrains, most recently work (Liu, Ren, Lin & Jiang, 2008) deals with the power saving problem, where the authors have proposed four sleeps scheduling schemes for different sorts of and analyze each of them by Stochastic Petri Net (SPN). By using the steady state probability matrix of the SPN models is obtained the average power consumptions and events delays. Also, there are many references devoted to communication protocols and recently in reference (Haines et al., 2007), PN are used as a formal verification technique of a MAC protocol. Where a case study is presented applying this technique to IEEE 802.11 centralised control mechanisms to support delay sensitive streams and fading data traffic. In reference (Pengand et al., 2007) author have used the PN to model the Session Initiation Protocol (SIP) and accomplished its verification.

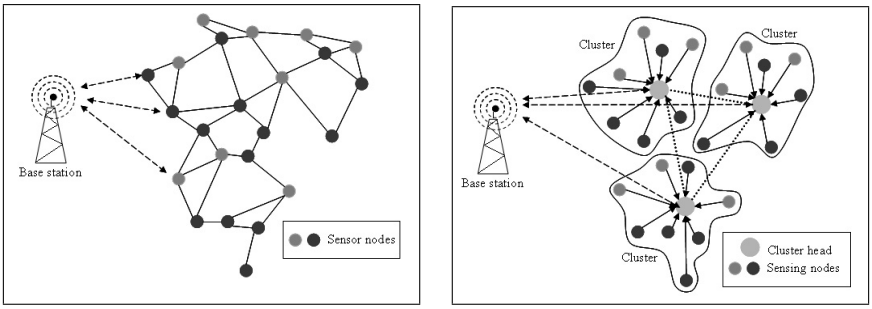
Another interesting work is presented in (Liu, Ye, Zhang & Li, 2008) where the 802.11i 4-way handshake protocol is analyzed utilizing High-level Petri Nets, confirming that the protocol is vulnerable to Denial-of-Service attack during handshake, then the authors propose an improved key management scheme.
The goal of the chapter is to present the most important part of the PN theory and to describe the possibilities of practical applications on WSN.
This paper is organized as follows: In section 2 the WSN are introduced, theirs principal characteristics and basic architectures, a brief introduction about to EC and LDPC codes and the basic concepts of PN are explained as well. In Section 3, we focus on the description of a communication model considered in order to a WSN. In Section 4, deal with the modeling phase by means of PN for our system. A PN model for the communication system and cryptography protocol are developed. Finally, we dedicated both a section to the conclusions and another to the references.-

## 2. Basic Concepts

In this section we developed briefly the topics that we are needed to better understand this work. For a deeper understanding look up the cited bibliography in each case.

### 2.1 The WSNs

The WSNs are defined as compound networks of a large number of tiny devices called sensor nodes, which have limited processing power, storage, bandwidth, and energy. In addition, a WSN might be often deployment on a large scale throughout a geographic region in hostile environments.
According to how sensors are grouped and how the information of sensors is routed through the network, there are two basic architectures to WSN, flat and hierarchical. In a flat architecture all nodes have almost the same communication capabilities and resource constraints and the information is routed by each sensor. In a hierarchical architecture, the sensor nodes are grouped in clusters where one of the member nodes is the "cluster head". This node is responsible for management and routing tasks. Fig. 1(a) shows a flat WSN model and a hierarchical WSN configuration is showed in Fig. 1(b).




(a) Flat WSN Model (b) Hierarchical WSN Model

Fig. 1. Architectures of Wireless Sensor Network

Further regarding the properties of the sensing interfaces of their nodes the WSNs can be classified in homogeneous (HoWSN) or heterogeneous (HeWSN). The HoWSN are character-

ized by the fact that all network nodes have the same properties unlike the heterogeneous (HeWSN) where nodes of different nature like temperature and movement sensors coexist.
Fig. 1(b) shows a HeWSN. Including the base station there are three types of nodes: base nodes, header nodes and sensors nodes. The base station is the interface between an application and a sensor network. The application network implements all the programs which manage the sensor network. The header nodes are devices with capability to route information that comes from other nodes. The sensor nodes are those that perceive changes produced in the environment such as temperature, humidity, light, movement among others.
Fig. 1(b) also shows the clustered sensor network. The sensor network is divided into several clusters. The headlines represent connection between the header nodes and the base station. Then header nodes manage the sensor nodes and route messages from others nodes; the sensor nodes are scattered in the monitoring field and it implements data process such as reading, encrypting, encoding and transmission.
As shown in the Fig. 2, a sensor node is composed of four basic components: sensing unit, processing unit, transceiver unit and a power unit.

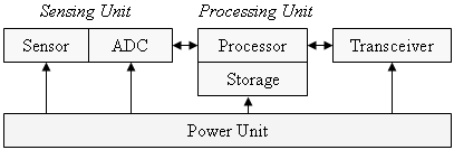



Fig. 2. Components of a Sensor

The sensing units are usually composed of two sub-units: Sensors and analogy-to-digital converters (ADCs). The analogy signals perceived by the sensor which are based in the observed phenomenon are converted to digital signals by the ADC, and then they are nourished to the unit of processing. The process unit, that is generally associated to a little storage device, manages / handles the procedures which make the sensor node collaborates with the others nodes in order to carry out the assigned sensor task.

### 2.2 Elliptic Curves

In our scheme we are using an elliptic curve defined over a finite field of characteristic two. In this case we only considered a representation of field elements and exist efficient way to effect arithmetic operations. The option we choose is to use optimal normal base (ONB). In which the add operation can accomplish by means of the operation XOR, without cost; squaring can be performed simply by a cycle shift of the coordinates of an element, hence, in hardware, it is almost cost free. A normal basis multiplication is not so simple but always more efficient than in other base. Reference over multiplication in ONB can be found in (Reyhani, 2003; 2006).
According to (Hankerson et al., 2004), an elliptic curve over a finite field $F_{2^m}$ is defined as the set of points that satisfies the following equation:

$$E : y^2 + xy = x^3 + ax^2 + b \tag{1}$$

where $a, b \in F_{2^m}$ and $b \neq 0$ in $F_{2^m}$. The set of solutions $(x, y)$ joined with a point at infinity, and special addition operation define an abelian group, are called the elliptic curve group.
An Elliptic Curve Cryptosystem (ECC) bases its security in the Elliptic Discrete Logarithm Problem (EDLP), that is, in the Discrete Logarithm Problem (DLP) defined on the group of

rational points of an elliptic curve.
Given an elliptic curve $E$ over a finite field $F$, a point $G \in E(F)$ and another known point $Q$ which is multiple of that point $G$. the problem is to find the integer $n$ such that $nG = Q$. This problem is computationally difficult to solve.

### 2.3 LDPC codes

The Low-density parity check (LDPC) codes were introduced by Robert Gallager in 1962 (Gallager, 1962), but with computational capabilities available then were dismissed. In the mid 90 were rediscovered by MacKay (MacKay, 1990) reaching great popularity by his performance closely to Shannon limit.
LDPC codes are a class of linear block codes characterized to have its sparse parity-check matrix which contains only a few 1's in comparison to the amount of 0's. In other words, the degree of all nodes is low. These codes can be represented by a Tanner bipartite graph consisting of two sets of nodes $\{c_i\}$ and $\{f_i\}$, respectively called variable nodes and check nodes. See example in Fig. 3.

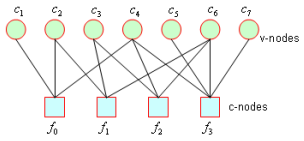



Fig. 3. Tanner Bipartite Graph

There are two kinds of LDPC codes: regular and irregular. For regular LDPC codes, all nodes of the same type have the same degree. For irregular LDPC codes, the degree of each set of nodes is chosen according to some distributions. A code vector $c$ is obtained multiplying a message vector $m$ by a generate matrix $G$

$$c = m * G \tag{2}$$

The corresponding check matrix $H$ has the property to be constructed with independent lineally rows vectors has which form a subspace of the subspace generated by the rows vectors of $G$. This signifies than each vector code satisfies the condition:

$$H * c = 0 \tag{3}$$

property that enables the decoding.
In (Castiñeira & Guy, 2006) we found a chapter dedicated to the binary LDPC codes. Description, construction and decoding algorithms can be studied here. Other reference about to construction of codes is (Carrasco & Johnston, 2008).
The decoding of LDPC codes is based on sum product algorithm (SPA), also called belief propagation algorithm (BPA), or message passing algorithm (MPA) which iteratively updates the posterior probabilities of bit nodes.
If we defined $q_{i,j}(b), b \in \{0,1\}$, as the probability computed based both on the received signal $y_i$ and the message $r_{j',i}(b)$ passed from the neighbors check nodes $c_i$ excluding the check node $j$. Also, we defined $r_{i,j}(b)$ as the probability computed based on the message $q_{j',i}(b)$ passed from the neighbors variable nodes $v_j$ excluding the variable node $i$ .
The MPA in the probability domain then proceeds as follows.

1. Estimate the noise power $\sigma^2$ . Then for $i = 1, 2, \ldots, n$, initialize $P_i(b) = prob(c_i = b/y_i)$.
2. Set $q_{ij}(b) = P_j(b)$ if the variable node $i$ and the check node $j$ are connected.
3. Update $r_{ij}(b)$ using

$$\begin{cases} r_{ji}(0) = \frac{1}{2} + \frac{1}{2}\prod_{i' \in v_{j\backslash i}} (1 - 2q_{i'j}(1)), \\ r_{ji}(1) = 1 - r_{ji}(0) \end{cases}$$

4. Update $q_{ij}(b)$ using

$$q_{ij}(b) = K_{ij}P(b) \prod_{j' \in c_{i\backslash j}} (r_{j'i}(b)); \;\; b = 0, 1$$

   where $K_{ij}$ is chosen to meet $q_{ij}(0) + q_{ij}(1) = 1$.
5. Compute the posterior probability of the code bit $c_i$ using

$$Q_i(b) = K_i P_i(b) \prod_{j \in c_i} (r_{ij}(b)); \;\; b = 0, 1$$

   where $K_i$ is chosen to ensure that $Q_i(0) + Q_i(1) = 1$.
6. For $i = 1, 2, ..., n$ set up:

$$\begin{cases} \widehat{c} = 1 \; if \; Q_i^{(1)} > Q_i^{(0)}, \\ \widehat{c} = 0 \; if \; Q_i^{(1)} < Q_i^{(0)} \end{cases}$$

If $\widehat{c}H^T = 0$ or the number of iterations equals the maximum limit, stop; else, go to Step 2. Here, $H$ is the parity check matrix of the LDPC codes.

### 2.4 Petri nets

A PN is identified as a particular kind of bipartite directed graph populated by three types of objects. They are places, transitions, and directed arcs connecting places and transitions. Formally, a PN can be defined by:

$$\mathcal{N} = \langle \mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{W} \rangle$$

where:

a. $\mathcal{T}$ is the set of places, $\mathcal{T}$ is the set of transitions, $\mathcal{P} \cap \mathcal{T} = \emptyset$

b. $\mathcal{F} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ is flow relationship

c. $\mathcal{W} : \mathcal{F} \to \mathcal{N}$ assigns a weight or multiplicity to each arc.

Usually, actions are associated to transitions and conditions are associated to places. A transition is enabled if each input place of contains at least a number of tokens equal to the weight of the arc connecting to. When an enabled transition fires, it removes tokens from its input places and deposits them on its output places. PN models are suitable for representing systems that exhibit concurrence, conflict, and synchronization.
Some important PN properties include a boundness (no capacity overflow), liveness (freedom from deadlock), conservativeness (conservation of non consumable resources), and reversibility (cyclic behavior). The concept of liveness is closely related to the complete absence of

deadlocks. A PN is said to be live if, no matter what marking has been reached from the initial marking, it is possible to ultimately fire any transition of the net by progressing through further firing sequences. This means that a live PN guarantees deadlock-free operation regardless of the firing sequence. Validation methods of these properties include reachability analysis, invariant analysis, reduction method, siphons/traps-based approach, and simulation (Reisig, 1986), (Reisig & Rozenberg, 1998).

## 3. System Description

Fig. 4 shows a block communication model we propose to communicate a header node and the base station in a hierarchical WSN. This model should use an elliptic curve encryption scheme and LDPC code in the process of messages transmission. On the left side we have the header node, transmitter of the message, and on the right side we have the base node, receiver of the message.

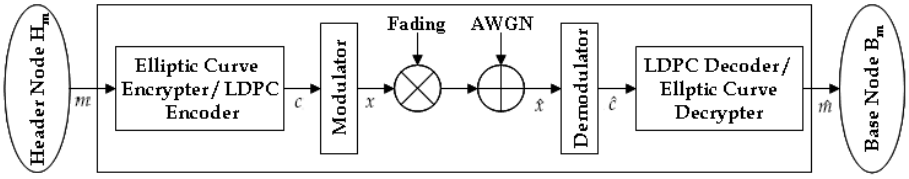



Fig. 4. Communication Model

In this model $m$ represents the transmitted binary message sequence. This sequence gets into the system to an Elliptic Curve Encrypter/LDPC Encoder block, where a transformation function converts them to an output code $c$, called usually a codeword. This codeword is the input sequence to a modulator block which converts them in a modulated signal $x$, which goes through the communication channel. Here the signal is disturbed by the noise coming from transmission medium and suffers the fading effects own of the wireless channels. From this process derives a corrupted signal $\hat{x}$, estimated from the signal $x$. Then the signal $\hat{x}$ enters to a demodulator block to be converted over again to a binary sequence. This sequence is called $\hat{c}$ and is an estimate of the codeword $c$. The LDPC Decoder / Elliptic Curves Decrypter block receives the binary sequence and transforms them by means of an inverse function in a message sequence $\hat{m}$ which is an estimate of the original message $m$. The base station receives $\hat{m}$ as a valid message and the process finishes.

## 4. Modeling and Analysis

In this section we develop some derived PN models since the communication system we have presented in the section 3. Firstly we analyze the system from the perspective of the communication among process. Then, we model a cryptography scheme which would be based on an elliptic curve, which at the same time is defined on a finite field.

### 4.1 Communication System

Fig. 5, shows the PN diagram of the communication system consistent with the model we presented on Fig. 4.

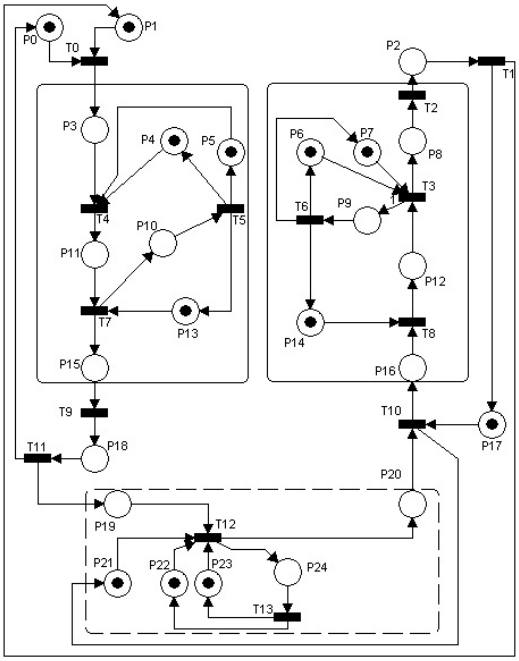



Fig. 5. Petri Diagram of the Communication System

We have made some assumptions regarding to the system in order to simplify the model.

1. Both transmitter and receiver are idle before begin the communication;
2. The channel is always available, there is not fight in access to the channel;
3. The public and private keys associated with the EC, are considered resources always available during a communication instance;
4. Both LDPC encoder and decoder are considered resources always available for the system.

The meaning of each *place* and *transition* is described in the Table 1 and 2 respectively. The initial marked which defines the system behavior denotes the following: The *places* $P0$ and $P1$ marked with one token tells us there is a new message and the transmitter is idle in order to transmit it. The places $P4$, $P5$ denotes resource availability to achieve the encryption process on the sender side. One mark in the *place* $P4$ will tell us that we count on a suitable EC and the another one in the *place* $P5$ will tell us as well that we dispose a valid public key to work on. In the same way the marks on the *places* $P6$ and $P7$ mean resource availability in order to achieve the decryption process on receiver side. In this case, one mark in the *place* $P6$ tells us that we dispose of the EC and anther one in the *place* $P7$ tells us that we count on a valid key to work on, here, the private key of the receiver.
On the other hand we have the coding resources, in the graphic represented by the *places* $P13$ and $P14$. A token allocated on the *place* $P13$ denotes LDPC encoder availability whereas the another one allocated on the *place* $P14$ shows the LDPC decoder is available. The *places* $P22$ and $P23$ represent noisy and fading processes into the channel. The tokens on these *places*

denote presence of those processes. The last *place* marked from the initial marked of the net is $P21$ which means that the channel is idle and free to transmit a signal.

| Place | Description | Place | Description |
|---|---|---|---|
| P0 | New message | P13 | LDPC encoder (sender side) |
| P1 | Transmitter idle | P14 | LDPC decoder (receiver side) |
| P2 | Received message | P15 | Encoded message (codeword) |
| P3 | Incoming message | P16 | Demodulated message |
| P4 | EC on the sender side (resource) | P17 | Receiver idle |
| P5 | Valid key (receiver public key) | P18 | Modulated message |
| P6 | EC on the receiver side (resource) | P19 | Channel input |
| P7 | Valid key (receiver private key) | P20 | Channel output |
| P8 | Decrypted message | P21 | Channel Idle |
| P9 | LDPC decoder and EC Restoring (receiver) | P22 | Fading process |
| P10 | LDPC decoder and EC Restoring (sender) | P23 | Noise process |
| P11 | Encrypted message | P24 | Channel parameters restoring |
| P12 | Decoded message | | |

Table 1. Places Description of PN in Fig. 5

Basically the PN shows different status that a message has to go through the communication process. The *transitions* represent functions that transform the message into different phases of process. Thus we have as follows:

1. Initially, the *transition* $T0$ is triggered, since the initial marked on the places $P1$ and $P2$ enable to do it. Then the message passes to *incoming message* status (place $P3$ is loaded with a token);
2. The *transitions* $T4$ triggers, because of the tokens in its incoming places $P3, P4, P5$ enables it. Then these ones are removed, new tokens are generated and allocated on the single output place $P11$ passing to the *encrypted message* status.
3. In the same way the *transition* $T7$ triggers removing the tokens from the places $P11$ and $P23$ and loading another two into places $P10$ and $P15$. The token in $P10$ makes conditional the resources restoring and the one in $P15$ turns the message into an *encode message* status. On the other one the *transition* $T5$ triggers tokens which restore both the curve and encode parameters.
4. A token on $P15$ enable the *transition* $T9$ which accomplishes his trigging passing to *modulated message* status, (place $P18$ is marked with a token).
5. Enabled now, the *transition* $T11$ trigger removing the token from $P18$ and allocating one into the place $P0$ and another one into the place $P19$ (*message in the channel input*).

| Transition | Description | Transition | Description |
|---|---|---|---|
| T0 | Transmit message | T7 | Encode message |
| T1 | Wait for message | T8 | Decode message |
| T2 | Deliver message | T9 | Modulate message |
| T3 | Decrypt message | T10 | Demodulate message |
| T4 | Encrypt message | T11 | Send message to channel |
| T5 | Restore parameters | T12 | Perturb message |
| T6 | Restore parameters | T13 | Reset channel parameters |

Table 2. Transitions Description of PN in Fig. 5

All these processes above encrypting-encoding-modulating belong to sender side, and in inverse proportion to this we have the receiver side; demodulating-decoding-decrypting. Nevertheless, we must as well to describe the channel process. Simply to say that if both places $P9$ and $P21$ are loaded each with a token, besides the places $P22$ and $P23$ then the transition $T12$ trigger. In other words, if there is a message in the channel input and this is idle, on the other hand there are both the noisy and fading processes, then the disturbance process is enabled which generate distortion in the message.
We perform two types of analysis over the PN model: a structural and a graph based analysis. The aim of the first is to show that the amount of states is finite (boundedness) and that all the represented activities can be done (liveness). The aim of graph based analysis is to determine the absence of dead states. The analysis was performed with the Petri net analyzer tool INA Roch & Starke. (1999).

Table 3 shows the structural analysis of the model of Fig. 5 where we can see there are eleven P/Invariants including all places, so we can conclude that the net is bounded. There are one T/Invariant (Table 7) covering all transitions, so the model is live. Finally the resulting reachability graph has 80 states large enough to draw it here.

| | **Place Invariants** |
|---|---|
| 1 | *P4 + P5* |
| 2 | *P6 + P7* |
| 3 | *P6 + P9* |
| 4 | *P4 + P10 + P11* |
| 5 | *P10 + P13* |
| 6 | *P6 + P12 + P14* |
| 7 | *P2 + P8 + P22 + P16 + P17* |
| 8 | *P0 + P3 + P4 + P10 + P15 + P8* |
| 9 | *P20 + P21* |
| 10 | *P22 + P23* |
| 11 | *P22 + P24* |
| | **Transition Invariants** |
| 1 | *T1 + T2 + T3 + T4 + T5 + T6 + T7 + T8 + T9 + T10 + T11 + T12 + T13* |

Table 3. Place and Transition Invariants of the Net in Fig. 7.

### 4.2 Cryptographic Protocol

We have used a sequence diagram (Fig. 6) to describe part of a secure communication protocol. It is based in the public key encryption scheme proposed by Menezes-Vanstone in (Menezes

et al., 1993) but also we have considered (Ontiveros et al., 2006). Specifically is considered the communication between a sensor and header node.

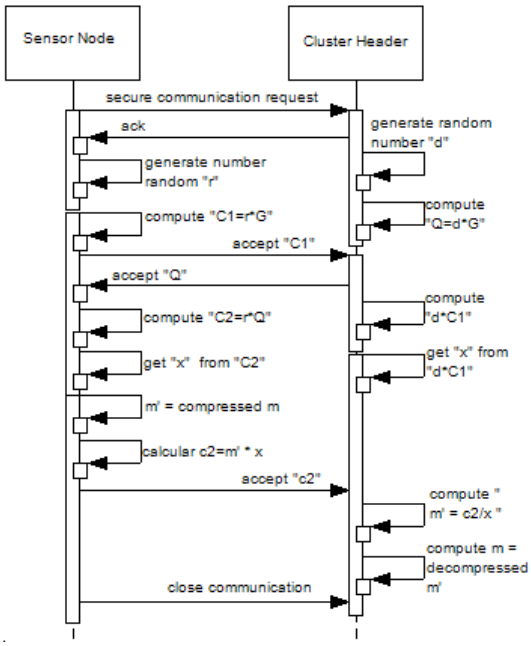



Fig. 6. Sequence Diagram of the Encryption and Decryption Process

It is important to consider the moments which operations are accomplished, with the aim of minimizing communication times. The sensor and header node should have same elliptic curve cryptoprocessor embedded in their architecture and both should know also a rational point $G$ which is a curve points generator.

The sequence diagram of Fig. 6 describes the behaviour of the protocol, as follows: when the sensor node wishes to send a message $m$ to their header node, sends a *security communication request*. The header node reply with a *acknowledgment of receipt (ack)* leaving the communication established. Both nodes performs concurrent tasks, the sensor node generates a random number $r$ and computes $C1 = r * G$, meanwhile the header node generates a random number $d$ and computes $Q = d * G$. After, when the sensor node has completed its task it sends the point computed $C1$ to the header node that receive it and sends the point $Q$ to the sensor node. Again, both nodes accomplish concurrent task with the received components as showed in the Fig. 6.

The sensor node computes $C2 = r * Q$ and it get the coordinate $x$ from $C2$ to compute it with the message $m$ which compress and send it to the header node. Meanwhile, the header node B computes the point $d * C1$ received and it gets its coordinate $x$ which it will use next to the decryption process. The next step consist in that collector node decompress and decrypts the message $m$ computing $m = C2/x$. Finally, in order to reduce the complexity, we assume that the sensor node close the communication.

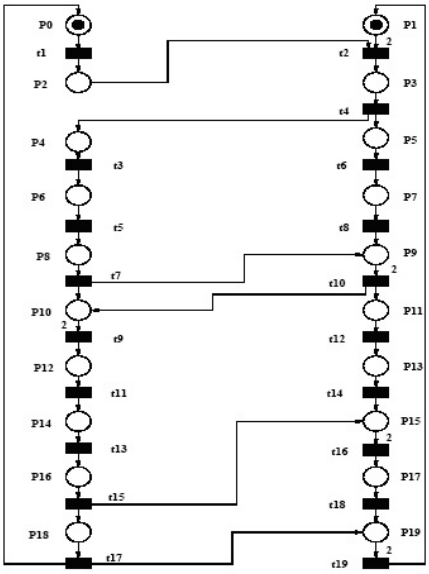



Fig. 7. Petri Diagram of the Cryptography Scheme

The Fig. 7 represents the PN model of the communication protocol discussed previously. Fig. 4 shows the meaning of each place and Table 5 shows the meaning of each transition.
We perform two types of analysis over the PN model: a structural and a graph based analysis. The aim of the first is to show that the amount of states is finite (boundedness) and that all the represented activities can be done (liveness). The aim of graph based analysis is to determine the absence of dead states. The analysis was perfomed with the Petri net analyzer tool INA Roch & Starke. (1999).

| Place | Description | Place | Description |
|---|---|---|---|
| P0 | Idle | P10 | Waiting $Q$ |
| P1 | Waiting for communication request | P11 | $Q$ sent |
| P2 | Ready to transmits | P12 | $C2$ computed |
| P3 | Security communication request received | P13 | $d * C1$ computed |
| P4 | Communication request reply received | P14 | $x$ obtained |
| P5 | Communication request sent | P15 | Waiting $c2$ |
| P6 | Random Number $r$ generated | P16 | $c2$ computed |
| P7 | Random Number $d$ generated | P17 | $c2$ decompressed |
| P8 | $C1$ computed | P18 | $c2$ sent |
| P9 | $Q$ computed and waiting $C1$ | P19 | Communication closed |

Table 4. Meaning of Places in Fig. 7

Table 3 shows the structural analysis of the model of Fig. 7 where we can see there are eleven P/Invariants, including all places, so we can conclude that the net is bounded. There are

one T/Invariant (Table 6 covering all transitions, so the model is live. Finally the resulting reachability graph represented by the set shown in Table 7 has 42 states.

| Place | Description | Place | Description |
|---|---|---|---|
| t1 | Start communication request | t11 | Get coordinate $x$ from $C2$ |
| t2 | Processing communication request | t12 | Compute $d * C1$ |
| t3 | Generate random number $r$ | t13 | Compute $c2m * x$ |
| t4 | Sending ack | t14 | Get $x$ from $d * C1$ |
| t5 | Compute $C1 = r * G$ | t15 | Compress and Send $c2$ |
| t6 | generate random number $d$ | t16 | Decompress $c2$ |
| t7 | Sending $C1$ | t17 | Close communication |
| t8 | Compute $Q = d * G$ | t18 | Compute $m = c2/x$ |
| t9 | Compute $C2 = r * Q$ | t19 | Return to wait communication request |
| t10 | Sending $Q$ | | |

Table 5. Transitions Description of PN Fig. 7

| | **Place Invariants** |
|---|---|
| 1 | *P0 + P2 + P3 + P4 + P5 + P6 + P7 + P8 + P9 + P10 + P12 + P14 + P16 + P18* |
| 2 | *P0 + P1 + P2 + P3 + P4 + P5 + P6 + P7 + P8 + P9 + P10 + P11 + P12 + P13 + P14 + P15 + P16 + P17 + P19* |
| | **Transition Invariants** |
| 1 | *t1 + t2 + t3 + t4 + t5 + t6 + t7 + t8 + t9 + t10 + t11 + t12 + t13 + t14 + t15 + t16 + t17 + t18 + t19* |

Table 6. Place and Transition Invariants of the Net in Fig. 7

| | | | | | |
|---|---|---|---|---|---|
| M0 | = | *PO + P1* | M21 | = | *P18 + P19* |
| M1 | = | *P1 + P2* | M22 | = | *P0 + 2P15 + P19* |
| M2 | = | *P3* | M23 | = | *P2 + 2P15 + P19* |
| M3 | = | *P4 + P5* | M24 | = | *P13 + P15 + P18* |
| M4 | = | *P5 + P6* | M25 | = | *P0 + P13 + P15 + P19* |
| M5 | = | *P5 + P8* | M26 | = | *P2 + P13 + P15 + P19* |
| M6 | = | *P7 + P8* | M27 | = | *P14 + P15* |
| M7 | = | *P7 + P9 + P10* | M28 | = | *P11 + P16* |
| M8 | = | *2P9 + P10* | M29 | = | *P11 + P15 + P18* |
| M9 | = | *2P10 + P11* | M30 | = | *P0 + P11 + P15 + P19* |
| M10 | = | *P11 + P12* | M31 | = | *P2 + P11 + P15 + P19* |
| M11 | = | *P11 + P14* | M32 | = | *P12 + P13* |
| M12 | = | *P13 + P14* | M33 | = | *P12 + P15* |
| M13 | = | *P13 + P16* | M34 | = | *2P10 + P13* |
| M14 | = | *P15 + P16* | M35 | = | *2P10 + P15* |
| M15 | = | *2P15 + P18* | M36 | = | *P8 + P9* |
| M16 | = | *P17 + P18* | M37 | = | *P5 + P9 + P10* |
| M17 | = | *P0 + P17 + P19* | M38 | = | *P6 + P7* |
| M18 | = | *P2 + P17 + P19* | M39 | = | *P6 + P9* |
| M19 | = | *P2 + 2P19* | M40 | = | *P4 + P7* |
| M20 | = | *P0 + 2P19* | M41 | = | *P4 + P9* |

Table 7. Reachability Set of PN in Fig. 7

## 5. Conclusion

We have used PN to model and formally analyze a communication system which holds a elliptic curve encryption scheme and it could works on a WSN. The results obtained shows formally that our approach are correct. However, we plan increase the complexity of models. We should consider critical factors related to constrained capacities of sensors nodes. We have analyze a more elaborate encryption scheme keep in mind adding a intermediate entity for the keys management.

## 6. References

Carrasco, R. A. & Johnston, M. (2008). *Non Binary Error Control for Wireless Communication and Data Storage*, John Wiley & Sons Ltd, United Kingdom.

Castiñeira, J. & Guy, P. (2006). *Essentials of Error Control Coding*, John Wiley & Sons Ltd, England.

Chen, S., Ge, Q., Shao, Q. & Zhu, Q. (2008). Modelling and performance analysis de wireless sensor network systems using petri nets, *International Technical Conference on Circuits/Systems, Computers and Communications* .

Gallager, R. G. (1962). Low-density parity-check codes, *IRE Transactions on Information Theory* (No. 1): 21–28.

Haines, R., Clemo, G. & Munro, A. (2007). Petri-nets for formal verification of mac protocols, *IET Software* **Vol. 1**: 39–47.

Hankerson, D., Menezes, A. & Vangtone, S. (2004). *Guide to Elliptic Curve Cryptography*, Springer Professional Computing.

Liu, B., Ren, F., Lin, C. & Jiang, X. (2008). Performance analysis of sleep scheduling schemes in sensor networks using stochastic petri net, *International Conference on Communications (ICC'08)* (No.): 4278 – 4283.

Liu, J., Ye, X., Zhang, J. & Li, J. (2008). Security verification of 802.11i 4-way handshake protocol, *International Conference on Communications (ICC'08)* **Vol. 1**: 1642–1647.

MacKay, D. J. C. (1990). Good error-correcting codes based on very sparse marices, *IEEE Transactions on Information Theory* (No. 2, pp. 399).

Menezes, A. J., Okamoto, T. & Vanstone, S. (1993). Reducing elliptic curve logarithms to logarithms in a finite field, *IEEE Transactions on Information Theory* (No. 5).

Ontiveros, B., I.Soto & Carrasco, R. (2006). Construction of an elliptic curve over finite fields to combine with convolutional code for cryptography, *Transactions on Computers Circuits, Devices and Systems* pp. 1642–1647.

Pengand, Y., Zhanting, Y. & Jizeng, W. (2007). Petri net model of session initiation protocol and its verification, *International Conference on Communications (ICC)* **Vol. 1**: 1861–1864.

Reisig, W. (1986). *Petri Nets. An introduction*, Springer-Verlag, Berlin Heidelberg.

Reisig, W. & Rozenberg, G. (1998). *Lectures on Petri Nets I: Basic Models*, Springer-Verlag, Berlin Heidelberg.

Reyhani, A. (2003). Efficient multiplication beyond optimal normal bases, *IEEE Transactions on Computers* **Vol. 52**(No. 4): 1642–1647.

Reyhani, A. (2006). Efficient algorithms and architectures for field multiplication using gaussian normal bases, *Transactions on Computers* **Vol. 55**(No. 1): 1642–1647.

Roch, S. & Starke., P. H. (1999). *INA: Integrate Net Analizer*, Humboldt-Universität zu Berlin, Berlin.

16

# Diagnosability in Switched Linear Systems

Lizette Rubio- Gómez[1], David Gómez-Gutiérrez[1], Antonio Ramírez-Treviño[1], Javier Ruiz-León[1] and Guillermo Ramírez-Prado[2]
[1]*Centro de Investigaciones y de Estudios Avanzados del IPN Unidad Guadalajara, México.*
[2]*Universidad Autónoma de Aguascalientes, México.*

## 1. Introduction

This work is concerned with fault detection of Switched Linear Systems (*SLS*) and the design of asymptotic diagnosers for this class of systems. In the approach herein addressed a system can evolve from a normal behavior to a faulty behavior selected from an a priori known set of possible faulty behaviors. The faulty or normal behaviors are represented by a family of Linear Systems (*LS*) and an Interpreted Petri Net (*IPN*) is used to represent the way in which the normal and faulty behaviors are visited. Moreover, there could exist several *LS* representing the system normal behavior in some operation points. Notice that several real systems can be represented with this model, for instance power systems can be represented by a family of *LS*, one for each operation point and one for each fault.
In *LS* there are works dealing with continuous fault diagnosis. In (Massoumnia et al., 1989) a residue is used to show when a fault $f_i$ can be detected in the system. Fault $f_i$ can be seen on the output of a residual model for fault $f_i$. However the number of faults that may be detected is restricted to the range of the output matrix. In (Gertler et al., 2002) the design of residual diagnosers is proposed.
The diagnosability in Discrete Event Systems (*DES*) has been addressed using Finite Automata (*FA*) (Sampath et al 1995), where a *DES* is diagnosable if there is no $F_i$ indeterminate cycles. Some extensions have been made to previous work, as (Hashtrudi-Zad et al., 2003), where the diagnoser and the system are allowed to start in different initial conditions. Diagnosability in *DES* has been also addressed using Petri Nets (*PN*) (Hadjicostis et al., 1999), (Hadjicostis et al., 2000) where a fault is detected if a conservative marking law is not fulfilled. In (Ramirez et al., 2007) a fault is detected when a siphon is unmarked, leading to a deadlock in the whole *PN*.
In (Fourlas et al., 2005) and (Fourlas 2009) the problem of fault diagnosability in hybrid system was addressed. That approach detects and isolates faults using the event sequences and associating to each faulty event a guard that can be taken from the continuous variables or discrete labels. The way in which continuous variables are chosen, however, is not mentioned.
This work is focused in fault diagnosis of systems where the set of potential faults can be a priori known. However the occurrence of them in real time needs to be detected and

isolated. A normal assumption followed in this work is that the faults are fired from certain system states, i.e. they cannot occur everywhere. For instance, a motor can be broken if it is working.

In the present work the diagnosability is addressed combining the residue generation, distinguishability and indeterminate cycles. Fault rises are represented by the firing of some *PN* transitions, herein named faulty transitions. Thus the *SLS* jump from one *LS* representing a normal behavior to another *LS* representing a faulty behavior by the firing of faulty transitions. Thus the fault detection and isolation could be carried out by detecting or computing the firing of such faulty transitions. In order to do such task, the knowledge of both, continuous and discrete parts of the input-output *SLS* information, are used.

In particular, the firing of a faulty transition can be detected if it is event detectable. This concept was derived for *IPN* (Ramirez et al., 2007), now this concept is extended to *SLS* in the three following ways.

If the faults are detectable in the sense introduced in (Massoumnia et al., 1989) then a set of residue generators, one generator per fault, can be built. Thus, when the output of generator representing fault $f_i$ is different from zero, then that transition $t_{f_i}$ (representing fault $f_i$) was fired for sure, i.e. the fault and the firing of a faulty transition was detected and isolated.

If the *LS* $\Sigma_a$ evolving before the firing of a faulty transition $t_{f_i}$ is distinguishable from the *LS* $\Sigma_b$ evolving after the firing of $t_{f_i}$ then the firing of $t_{f_i}$ can be detected and isolated by computing which *LS* systems is evolving, either $\Sigma_a$ or $\Sigma_b$.

If the firing of a faulty transition cannot be detected because it is not event detectable, the faults are not detectable as in (Massoumnia et al., 1989) and distinguishability does not hold, then the not firing of a faulty transition can be detected if the firing of transitions representing the *SLS* normal behavior is detected.

This paper is organized as follows. In Section 2 the background of *LS*, and *IPN* are presented, as well as the *SLS* definition and diagnosability in *SLS*. In Section 3 the characterization of diagnosability using the concepts of distinguishability and event detectable is presented. The diagnoser design is presented in Section 4, and an illustrative example is reported in Section 5. The last section presents the conclusions and future work.

## 2. Preliminaries

Through this work, *SLS* are represented by the tuple $\langle \mathcal{F}, (Q, M_0) \rangle$, where $\mathcal{F}$ is a family of continuous *LS* and $(Q, M_0)$ is an *IPN*. Next two subsections are devoted to briefly present these dynamical systems. An interested reader can consult (Chen, 1970; Wonham, 1979) for *LS*, and (Desel and Esparza, 1995; Rivera-Rangel, 2005) for Petri nets and interpreted Petri nets. Afterwards, the formal definition of *SLS* is presented.

### 2.1 Linear systems

Definition 2.1 A Linear System *(LS)* is described by for all t≥0

$$\Sigma \begin{cases} \dot{x}(\tau) = Ax(\tau) + Bu(\tau), & x(\tau_0) = x_0 \\ y(\tau) = Cx(\tau) \end{cases} \tag{2.1}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^p$ is the system input vector, $y \in \mathbb{R}^q$ is the system output vector, and $A$, $B$, and $C$ are, respectively, $n \times n$, $n \times p$ and $q \times n$ constant matrices. The state space of the dynamical equation (2.1) is $\mathcal{X}$.

Through this work, equation (2.1) will be referred as the *LS* $\Sigma(A, B, C)$ or simply system $\Sigma$.

**Definition 2.2** A *LS*, $\Sigma(A, B, C)$, is said to be observable at $\tau_0$ if there exists a finite $\tau_1 > \tau_0$ such that for any state $x_0$ at time $\tau_0$, the knowledge of the input $u[\tau_0, \tau_1]$ and the output $y[\tau_0, \tau_1]$ over the time interval $[\tau_0, \tau_1]$ suffices to determine the state $x_0$. Otherwise, the dynamic equation $\Sigma(A, B, C)$ is said to be unobservable at $\tau_0$.

**Theorem 2.3** Let $\Sigma(A, B, C)$ be a *LS*. Then $\Sigma(A, B, C)$ is observable if and only if the unobservable subspace $\mathcal{N} \subseteq \mathcal{X}$ of $\Sigma(A, B, C)$ is the trivial subspace 0, i.e. for all t≥0

$$\mathcal{N} = \bigcap_{i=1}^{n} \ker\left(CA^{i-1}\right) = 0 \tag{2.2}$$

or equivalently, if and only if it does not exist a nontrivial subspace $\mathcal{V} \subseteq \mathcal{X}$, such that $\mathcal{V} \subseteq \ker C$ and $\mathcal{V}$ is $A$-invariant (i.e. $A\mathcal{V} \subseteq \mathcal{V}$).

*Proof.* The proof is presented in (Wonham, 1979). ■

### 2.2 Petri nets and Interpreted Petri nets

**Definition 2.4** A Petri net system or Petri net $N = (G, M_0)$ is a bipartite digraph where $G = (P, T, F)$ and:

- $P = \{p_1, p_2, \ldots\}$ is a finite set of vertices called places,
- $T = \{t_1, t_2, \ldots\}$ is a finite set of vertices called transitions,
- $F$ is a relation on $P \cup T$ such that $F \cap (P \times P) = F \cap (T \times T) = \emptyset$.
- $M_0$ is the initial token distribution or initial marking, where a marking $M: P \to \mathbb{Z}^+$ is the number of tokens associated to each place, this is usually expressed as a vector of dimension equal to $|P|$.

The incidence matrix of $G$ is $C = [c_{ij}]$ such that

$$c_{ij} = \begin{cases} 0 & \text{if } \left(p_i, t_j\right) \notin F \text{ and} \left(t_j, p_i\right) \notin F \text{ or} \\ & \left(p_i, t_j\right) \in F \text{ and } \left(t_j, p_i\right) \in F \\ -1 & \text{if } \left(p_i, t_j\right) \in F \text{ and} \left(t_j, p_i\right) \notin F \\ 1 & \text{if } \left(p_i, t_j\right) \notin F \text{ and } \left(t_j, p_i\right) \in F. \end{cases}$$

In a *PN* system, a marking $M$ enables a transition $t_j$ if it marks every place $p_i$ such that $\left(p_i, t_j\right) \in F$; if $t_j$ is enabled at $M_k$ then the transition $t_j$ can be fired reaching a new marking $M_{k+1}$ (written as $M_k \xrightarrow{t_j} M_{k+1}$).

In a *PN*, $\sigma = t_1 t_2 \ldots t_k$ is a firing transition sequence leading from $M$ to $M_k$ (written as $M \xrightarrow{\sigma} M_k$) if $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} \cdots \xrightarrow{t_k} M_k$. A marking $M_k$ is said to be reachable from $M$ if $M \xrightarrow{\sigma} M_k$ for some firing transition sequence $\sigma$. The reachability set of a *PN* is the set of all possible markings reached from $M$, this set is denoted by $R(G, M)$.

A Petri net $(G, M_0)$ is live if, for every $M \in R(G, M_0)$ and every $t \in T$ there exists a marking $M' \in R(G, M)$ which enables $t$; it is cyclic if $M_0$ is reachable from every $M \in R(G, M_0)$ and it is binary if for every marking $M_k \in R(G, M_0)$ and every place $p_i \in P$ $M_k(p_i) \leq 1$.

As mentioned before, *IPN* are used to capture the discrete nature of *SLS*, this extension allows to associate input and output signals to *PN* models. These nets are defined as follows.

**Definition 2.5** An interpreted Petri net *(IPN)* is the pair $(Q, M_0)$ with $Q = (G, \xi, \Gamma, \lambda, \varphi)$ where:

- $(G, M_0)$ is a *PN* system.
- $\xi = \{\alpha_1, \alpha_2, \ldots\}$ is a finite set of input symbols.

• $\Gamma = \{\gamma_1, \gamma_2, \dots\}$ is a finite set of output symbols.

• $\lambda: T \to \xi \cup \{\varepsilon\}$ is a labeling function of transitions with the following constraint: $\forall t_j, t_k \in T,\ j \neq k,\ \lambda(t_j) \neq \varepsilon, \lambda(t_k) \neq \varepsilon$ if there exists $p_i$ such that $(p_i, t_j) \in F$ and $(p_i, t_k) \in F$ then $\lambda(t_j) \neq \lambda(t_k)$, where $\lambda(t) = \varepsilon$ represents a silent transition and $\varepsilon$ is an internal system event.

There exists an output relation $\varphi \subset P \times \Gamma$ associating to each place a set of output symbols, where $(p_i, \gamma_j) \in \varphi$ if the output symbol $\gamma_j$ is generated as an output when the place $p_i$ is marked. The relation $\varphi$ can be represented in a matricial form, where $\varphi_{ij} = 1$ if $(p_i, \gamma_j) \in \varphi$ and $\varphi_{ij} = 0$ otherwise.

If $\lambda(t_j) = \alpha_i \neq \varepsilon$ and the symbol $\alpha_i$ is present as an input when $t_j$ is enabled, then $t_j$ must fire. If $\lambda(t_j) = \varepsilon$ and $t_j$ is enabled then $t_j$ can be fired. The transition $t_i$ is said to be manipulated if $\lambda(t_i) \neq \varepsilon$. Otherwise it is non manipulated.

The reachability set of $(Q, M_0)$ is denoted as $R(Q, M_0)$ and is defined in a similar way as $R(G, M_0)$. Notice that $R(Q, M_0) \subseteq R(G, M_0)$ since the labeling function $\lambda$ may force two transitions equally labeled to fired simultaneously in the *IPN*.

**Definition 2.6** Let $M_i \overset{t_{i+1}}{\to} M_{i+1} \overset{t_{i+2}}{\to} \cdots \overset{t_l}{\to} M_l$ be a marking sequence. The input-output sequence $\omega = (\alpha_0, y_0) \cdots (\alpha_k, y_k)$ generated by $M_i \cdots M_l$ is defined inductively as follows:

• $\alpha_0 = \varepsilon$ and $y_0 = \varphi M_i$.

• If $\alpha_n = \lambda(t_j)$ and $y_n = \varphi M_j$ then $\alpha_{n+1} = \lambda(t_k)$ and $y_{n+1} = \varphi M_k$ if $\varphi M_k \neq \varphi M_j$ and there is no $M_{\hat{k}}$ occurring after $M_j$ and before $M_k$ such that $\varphi M_{\hat{k}} \neq \varphi M_j$ and $\varphi M_{\hat{k}} \neq \varphi M_k$.

Note that, if $\varphi M_i = \varphi M_{i+1}$ then the input-output sequence generated by $M_i M_{i+1}$ is the same as the one generated by $M_i$ or $M_{i+1}$, say $\omega = (\varepsilon, \varphi M_i)$.

The marking sequences set corresponding to $\omega$ is defined as for all t≥0

$$S_\omega = \{M_i \cdots M_k | M_i \cdots M_k \text{ generates the input} - \text{output sequence } \omega\} \quad (2.3)$$

### 2.3 Switched linear system

**Definition 2.7** A Switched Linear System *(SLS)* is the tupla $SLS = \{F, N\}$, where $F$ is a family of Linear Systems (*LS*) $\{\Sigma_1, \Sigma_2 \dots, \Sigma_s\}$ and $N$ is an Interpreted Petri Net *(IPN)*, where the following considerations are fulfilled:

- The interpreted Petri net $N$ is live, binary and cyclic.
- $\Phi: P \to F \cup \{\mu\}$ is a function associating to each place a $LS$ or $\mu$, $\Phi(\mathrm{p_i}) = \mu$ indicates that $p_i$ has not associated $LS$, where the following constraint is fulfilled: $\forall t_j \in T\ \sum_{p_i \in {}^\bullet t_j} \dim \Phi(p_i) = \sum_{p_k \in t_j^\bullet} \dim \Phi(p_k)$
- $\forall t_j$ such that $\{p_1, \dots, p_q\} \in {}^\bullet t_j$ and $\{p_q, \dots, p_g\} \in t_j^\bullet$ is defined $\delta_{t_j}: \mathbb{R}^n \to \mathbb{R}^n$ with $\delta_{t_j}\left(\begin{bmatrix} \overline{x_1}(\tau_k^-) \\ \vdots \\ \overline{x_q}(\tau_k^-) \end{bmatrix}\right) = \begin{bmatrix} \overline{x_q}(\tau_k) \\ \vdots \\ \overline{x_g}(\tau_k) \end{bmatrix}$, where $\overline{x_\iota}$ has the same dimension than $\Phi(p_i)$, $p_i \in {}^\bullet t_j$ and $\overline{x_\iota}$ has the same dimension then $\Phi(p_i)$, $p_i \in t_j^\bullet$. The final state at time $\tau_k$ when $t_j$ is fired, $\overline{x_\iota}(\tau_k)$ is the initial condition of $\Phi(p_i)$.

- If the output symbol $\gamma_j$, where $(p_i, \gamma_j) \in \varphi$ appears more than one time, then it must be associated to places of the same P-component.
- The dwell time in a state $M_k$ is finite and different from zero.

**Definition 2.8** Let $\{F, N\}$ be a *SLS*. The *SLS* is named a Complete *SLS* if the system faults are included in the model in the following way. A faulty transition $t_{f_i}$ and a faulty place $p_{f_i}$ are added to the *IPN* in order to represent each fault $f_i$ that the *SLS* model must capture. The place $p_j$ of the *SLS* where the system can evolve from a normal behavior into a faulty behavior, where fault $f_i$ is present, must be connected through an arc from $p_j$ to $t_{f_i}$ and also an arc from $t_{f_i}$ to $p_{f_i}$ must be added. The set $F$ is recomputed as $F' = F \cup \{\Phi(p_{f_1}), \ldots, \Phi(p_{f_k})\}$.

**Remark 2.9** Through this work, $F, P$ *and* $T$ are decomposed as $F = F_N \cup F_F$, $P = P_N \cup P_F$ and $T = T_N \cup T_F$ in a *Complete SLS* $\{F, N\}$, where $F_N, P_N,$ *and* $T_N$ are elements of the *SLS* and $F_F, P_F,$ *and* $T_F$ are the faulty elements added to the *Complete SLS* to include the faulty behavior in the *SLS*.

**Definition 2.10** Let $\{F, N\}$ be a *Complete SLS* and $T_F = \{t_{f_1}, \ldots, t_{f_k}\}$ the set of faulty transitions. A place $p_k \in {}^{\blacksquare}T_F$ is named a risk place and a transition $t_i \in T - T_F$ where $t_i \in p_k{}^{\blacksquare}$ is named a post-risk transition.

## 2.4 Example

**Example 2.11** Consider the *SLS* where the discrete part is represented by the *IPN* depicted below

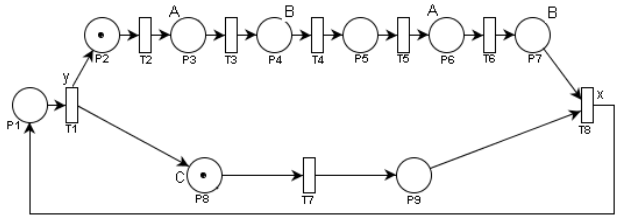



Fig. 1. Normal behavior of the *SLS*.

The *LS* associated to each place are presented in the next table:

| $P$ | $\Phi(P_N) = \Sigma(A, B, C)$ | | |
|---|---|---|---|
| $p_1$ | $A_1 = \begin{bmatrix} -2 & 0 & 1 & 0 \\ -2 & 3 & 1 & 0 \\ 0 & -3 & 2 & -1 \\ -3 & 2 & 1 & 1 \end{bmatrix}$ | $B_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$ | $C_1 = \begin{bmatrix} -1 & 0 & 2 & 1 \\ 1 & 0 & -1 & 0 \end{bmatrix}$ |
| $p_2$ | $A_2 = \begin{bmatrix} -2 & -1 \\ 3 & -5 \end{bmatrix}$ | $B_2 = \begin{bmatrix} -3 \\ 1 \end{bmatrix}$ | $C_2 = [1 \quad 3]$ |
| $p_3$ | $A_3 = \begin{bmatrix} -7 & -3 \\ 2 & -2 \end{bmatrix}$ | $B_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ | $C_3 = [7 \quad 15]$ |
| $p_4$ | $A_4 = \begin{bmatrix} -3 & 0 \\ 3 & -4 \end{bmatrix}$ | $B_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ | $C_4 = [0 \quad 1]$ |
| $p_5$ | $A_5 = \begin{bmatrix} -3 & 3 \\ 0 & -4 \end{bmatrix}$ | $B_5 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ | $C_5 = [2 \quad -1]$ |
| $p_6$ | $A_6 = \begin{bmatrix} -5 & -4 \\ 2 & 1 \end{bmatrix}$ | $B_6 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ | $C_6 = [15 \quad 12]$ |
| $p_7$ | $A_7 = \begin{bmatrix} -5 & 2 \\ -1 & -2 \end{bmatrix}$ | $B_7 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ | $C_7 = [1 \quad 0]$ |
| $p_8$ | $A_8 = \begin{bmatrix} -1 & 3 \\ 0 & -2 \end{bmatrix}$ | $B_8 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ | $C_8 = [2 \quad -9]$ |
| $p_9$ | $A_9 = \begin{bmatrix} -6 & 1 \\ -3 & -2 \end{bmatrix}$ | $B_9 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ | $C_9 = [1 \quad 1]$ |

Table 1. Linear Systems associated to *IPN* places.

Notice that the *LS* evolving when the place $p_1$ is marked is of dimension four with two inputs and two outputs, when the transition $t_1$ is fired the *LS* associated to the places $p_2$ and $p_8$ evolves independently from each other, each one of dimension two with a single input and a single output, this situation may be interpreted as a decoupled of two machines cooperating with each other, in a similar way the firing of transition $t_8$ represents that two machines are cooperating in such a way that their dynamics couple together.
Then the faulty behavior $F_F = \{p_{10}, p_{11}\}$ is added according to the proposed model. The resulting *Complete SLS* model with both behaviors is represented in next figure.

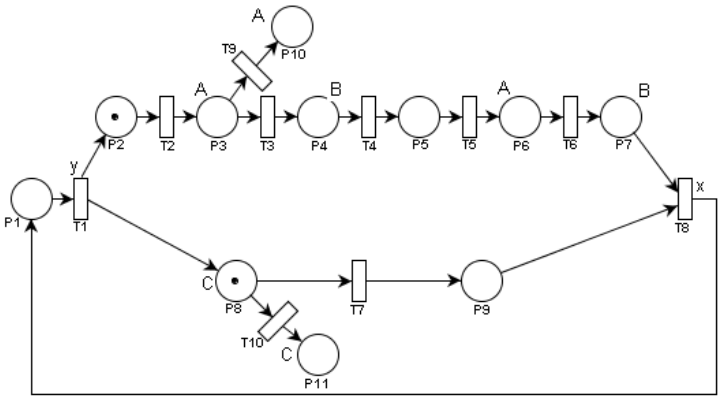



Fig. 2. Normal and faulty behavior of the *SLS*

The post-risk transitions are $t_3$ and $t_7$, while $T_F = \{t_9, t_{10}\}$. The faults $\Phi(p_{10})$ and $\Phi(p_{11})$ have the following information associated:

| $P$ | $\Phi(P_F) = \Sigma(A, B, C)$ | | |
|---|---|---|---|
| $p_{10}$ | $A_{10} = \begin{bmatrix} -2 & 0 \\ 2 & -5 \end{bmatrix}$ | $B_{10} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ | $C_{10} = [1 \quad 0]$ |
| $p_{11}$ | $A_{11} = \begin{bmatrix} -3 & -1 \\ 3 & -7 \end{bmatrix}$ | $B_{11} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ | $C_{11} = [-3 \quad -1]$ |

Table 2. Linear Systems associated to *IPN* places.

## 3. Characterization of diagnosable *Complete SLS*

The characterization of diagnosability in *SLS* is based on the observation of the firing transition sequence containing a faulty transition. However, since the marking of some places cannot be observed from the output information, then an output information sequence could be generated by several possible firing transition sequences, some of them containing the faulty transition.

The diagnosability characterization is then reduced to compute the actual fired transition sequence in a finite number of fired transitions when a faulty transition is fired. Notice that the firing of a faulty transition is equivalent to that post risk transitions cannot be fired. Thus the idea behind this work is to detect if post risk transitions can be fired (the *SLS* is normaly evolving) or not (the *SLS* is in a fault behavior).

In order to ensure that the detection of the firing of post risk transitions is computed in a finite number of transition fires, the concept of relative distance between transitions is presented. Although the computation of this distance seems to be an *NP* complete problem, the finiteness of this distance can be computed efficently for the *IPN* used in the *SLS*.

### 3.1 Relative Distance Concepts

**Definition 3.1** The relative distance between any pair of transitions $t_i, t_j \in T$, $D(t_i, t_j)$, in the *IPN*, is the maximum number that $t_i$ can be fired when a token is held in the place ${}^\bullet t_j$. The maximum relative distance between any pair of transitions $t_i, t_j \in T$ is $D_H(t_i, t_j) = \max(D(t_i, t_j), D(t_j, t_i))$.

Notice that in live, cyclic and binary *IPN* the finiteness of this distance can be computed easily. For instance, the *IPN* of the *SLS* in Example 2.12 is covered by P-components, and these P-components become siphons when faulty transitions are added. Thus the firing of a faulty transition unmarks a P-component, thus all transitions of the P-component cannot fire anymore, thus the maximum relative distance between any pair of transitions of the P-component is finite. The next proposition states that when all T-Components share transitions with a P-component, if the P-component becomes unmarked, then the *IPN* is no longer live.

**Proposition 3.2** Let $\{F, N\}$ be a *SLS* and $\{F', N'\}$ be a Complete *SLS*, where all T-Components of the *IPN* in the SLS share transitions with a P-component containing a risk place $(p_i)$. If the faulty transition connected to the risk place is fired, then after a finite number of transition firing the *IPN* is no longer live (or blocked).

**Proof**. Since the *IPN* of the *SLS* is live and binary, then it is covered by P-Components (Dessel et al., 1995). When the faulty transition is added, the places of the P-Component

become a siphon, thus the firing of the faulty transition unmarks the siphon. Since siphons cannot be marked again, then all transition in the P-Component cannot be fired. Moreover, since all T-Component share transitions with the P-Component, and every T-Component needs to fire all its transitions to be live, then eventually the transitions of the T-Component cannot be fired, since they need the firing of the transitions in the P-component. ■

**Corollary 3.3** Let $\{F, N\}$ be a *SLS* and $\{F', N'\}$ be a Complete *SLS*, where all T-Components of the *IPN* in the *SLS* share transitions with any P-component containing a risk place $(p_i)$. Then $t_i, t_j \in T$ is $D_H(t_i, t_j) < \infty$, where $t_i$ is the post risk transition $t_i \in {}^{\blacksquare}p_i$.

Since the firing of any sequence contains transition $t_i \in {}^{\blacksquare}p_i$ then the idea is to add a "marker" to transition $t_i \in {}^{\blacksquare}p_i$ in such a way that the firing (or not firing) of this transition can be observed from the *SLS* output. The firing of such transition can be detected using the *LS* input output information based on distinguishability property or the *IPN* input output information, based en event detectability property. Next subsection formalizes these ideas.

### 3.2 Distinguishability Concepts

**Definition 3.4** The linear systems $\Sigma_i(A_i, B_i, C_i)$, $\Sigma_j(A_j, B_j, C_j)$ are said to be distinguishable from each other if the knowledge of the input $u[t_0, t_1]$ and the output $y[t_0, t_1]$ over the finite time interval $[t_0, t_1]$ suffices to determine which *LS* is evolving.

**Notation 3.5** Let $\Sigma_i(A_i, B_i, C_i)$, and $\Sigma_j(A_j, B_j, C_j)$ be two *SISO* linear systems, then the linear system $\bar{\Sigma}\{\bar{A}, \bar{B}, \bar{C}\}$ denotes the extended LS form with the matrices for all t≥0

$$\begin{aligned} \bar{A} &= \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \\ \bar{B} &= \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \\ \bar{C} &= [C_1 \quad -C_2]. \end{aligned} \tag{3.1}$$

**Lemma 3.6** Let $\Sigma_1(A_1, B_1, C_1)$, $\Sigma_2(A_2, B_2, C_2)$ be two *SISO LS* where $A_1 \in \mathbb{R}^{n\times n}$ and $A_2 \in \mathbb{R}^{m\times m}$. Then the linear systems $\Sigma_1(A_1, B_1, C_1)$ and $\Sigma_2(A_2, B_2, C_2)$ are distinguishable from each other if and only if the only solution to the equation for all t≥0

$$\bar{C}e^{\bar{A}t}\left[x_0 + \int_0^t e^{-\bar{A}\tau}\bar{B}u(\tau)d\tau\right] = 0 \tag{3.2}$$

is $x_0 = 0$ and $u(t) = 0$.

**Proof.** If the linear systems $\Sigma_1(A_1, B_1, C_1)$ and $\Sigma_2(A_2, B_2, C_2)$ are indistinguishable from each other then there exists an input $u(t)$ such that the same output $y(t)$ is produced by both systems when $u(t)$ is applied, i.e. for two different initial conditions $x_0^1$, $x_0^2$ it holds that for all t≥0:

$$y(t) = C_1 e^{A_1 t}\left[x_0^1 + \int_0^t e^{-A_1\tau}B_1 u(\tau)d\tau\right] \tag{3.3}$$

and

$$y(t) = C_2 e^{A_2 t}\left[x_0^2 + \int_0^t e^{-A_2\tau}B_2 u(\tau)d\tau\right] \tag{3.4}$$

then combining equations (3.3) and (3.4):

$$C_1 e^{A_1 t}\left[x_0^1 + \int_0^t e^{-A_1\tau}B_1 u(\tau)d\tau\right] = C_2 e^{A_2 t}\left[x_0^2 + \int_0^t e^{-A_2\tau}B_2 u(\tau)d\tau\right] \tag{3.5}$$

this equation can be written as

$$C_1 e^{A_1 t}x_0^1 - C_2 e^{A_2 t}x_0^2 = \int_0^t \left[-C_1 e^{-A_1(t-\tau)}B_1 + C_2 e^{-A_2(t-\tau)}B_2\right]u(\tau)d\tau. \tag{3.6}$$

Now, since (3.6) is equivalent to:

$$\begin{aligned}&[C_1 \quad -C_2]\begin{bmatrix}e^{A_1 t} & 0\\ 0 & e^{A_2 t}\end{bmatrix}\begin{bmatrix}x_0^1\\ x_0^2\end{bmatrix} =\\ &-\int_0^t [C_1 \quad -C_2]\begin{bmatrix}e^{A_1(t-\tau)} & 0\\ 0 & e^{A_2(t-\tau)}\end{bmatrix}\begin{bmatrix}B_1\\ B_2\end{bmatrix}u(\tau)d\tau\end{aligned} \tag{3.7}$$

Equation (3.7) can be written in terms of the matrices (3.1), with $x_0 = [x_0^1 \quad x_0^2]^T$, then

$$\bar{C}e^{\bar{A}t}\left[x_0 + \int_0^t e^{-\bar{A}\tau}\bar{B}u(\tau)d\tau\right] = 0. \tag{3.8}$$

Since $\Sigma_1(A_1, B_1, C_1)$ is indistinguishable from $\Sigma_2(A_2, B_2, C_2)$, thus there exist solutions $x_0 \neq 0$ and $u(t) \neq 0$ to Equation (3.8). The converse is also true, then $\Sigma_1(A_1, B_1, C_1)$ and $\Sigma_2(A_2, B_2, C_2)$ are indistinguishable from each other if and only if the only solution to Equation (3.8) is $x_0 = 0$ and $u(t) = 0$. ■

**Theorem 3.7** Let $\Sigma_i(A_i, B_i, C_i)$, $\Sigma_j(A_j, B_j, C_j)$ be two *SISO* linear systems, where the matrices $A_i \in \mathbb{R}^{n\times n}$ and $A_j \in \mathbb{R}^{m\times m}$. Then the linear systems $\Sigma_1(A_1, B_1, C_1)$, $\Sigma_2(A_2, B_2, C_2)$ are distinguishable from each other if and only if the extended *LS* $\Sigma_{i,j}(A_{i,j}, B_{i,j}, C_{i,j})$ has no system zeros.

The proof follows from Lemma 3.6 ■

### 3.3 Distinguisher Design

The distinguisher proposed in this work is presented in Figure 3. It is capable of compute which *LS* is evolving from a set of possible evolving and distinguishable from each other set of *LS*. This figure shows that the diagnoser is composed of an observer, a set of simulation models of all possible evolving *LS* and a decision block.

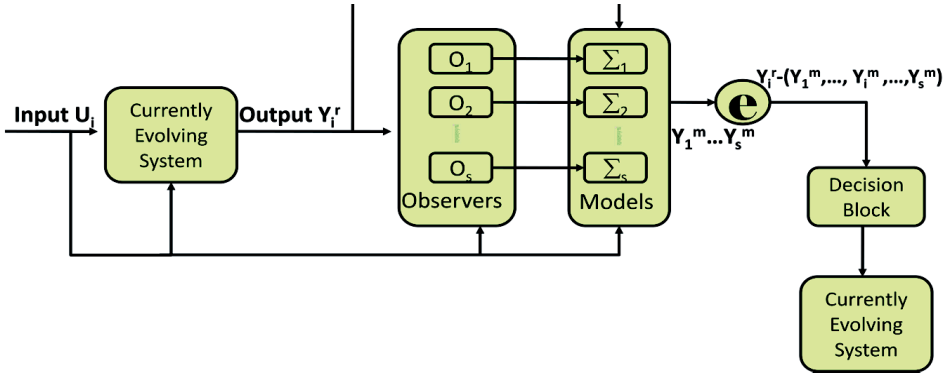



Fig. 3. Distinguishability architecture.

The distinguisher works as follows. The input and output of the current evolving *LS* are introduced to all Luenberger observers of the set of possible evolving *LS*. Notice that one of the Luenberger observers is the observer of the current evolving *LS*, thus at least this observer will compute the current *LS* state. The state estimated by each observer $O_i$ is given as initial condition to the corresponding *LS* model $\Sigma_i$ and a simulation starts. Since all the *LS* are distinguishable from each others, then the output of just one system will be equal to the

current *LS* output. Thus the decision block isolates the model corresponding to the current *LS* evolving.

### 3.3 Event Detectability

**Definition 3.8** An *IPN* given by $(Q, M_0)$ is event-detectable if any transition firing can be uniquely determined by the knowledge of the input given to $(Q, M_0)$ and output signals that it produces.

The following Lemma provides a structural characterization of the $IPN$ exhibiting event-detectability.

**Lemma 3.9** A live *IPN* given by $(Q, \mathcal{M}_0)$ is event-detectable if

1. $\forall \mathbf{t}_i, \mathbf{t}_j \in T$ such that $\lambda(\mathbf{t}_i) = \lambda(\mathbf{t}_j)$ or $\lambda(\mathbf{t}_i) = \varepsilon$ it holds that $\varphi\mathcal{C}(\bullet, \mathbf{t}_i) \neq \varphi\mathcal{C}(\bullet, \mathbf{t}_j)$

and

2. $\forall \mathbf{t}_k \in T$ it holds that $\varphi\mathcal{C}(\bullet, \mathbf{t}_k) \neq 0$.

The proof is presented in (Rivera et al., 2005). ■

**Lemma 3.10** A transition $t_k \in T$ in the $IPN\ N$ of the $SLS$ is event-detectable if:

1. $$\begin{cases} \varphi C(\bullet, t_k) \neq 0 \text{ or} \\ \begin{bmatrix} \Phi(p_i') & & \\ & \ddots & \\ & & \Phi(p_j') \end{bmatrix} \textit{ is distinguishable from } \begin{bmatrix} \Phi(p_k'') & & \\ & \ddots & \\ & & \Phi(p_l'') \end{bmatrix} \\ \textit{where } p_r' \in {}^{\blacksquare}t_k \textit{ and } p_r'' \in t_k^{\blacksquare} \end{cases}$$

and

2. $$\begin{cases} \forall t_i \in T \text{ such that } \varphi C(\bullet, t_k) = \varphi C(\bullet, t_i) \text{ fulfills that} \\ \begin{bmatrix} \Phi(p_i') & & \\ & \ddots & \\ & & \Phi(p_j') \end{bmatrix} \textit{ is distinguishable from } \begin{bmatrix} \Phi(p_k'') & & \\ & \ddots & \\ & & \Phi(p_l'') \end{bmatrix} \\ \textit{where } p_r' \in t_k^{\blacksquare} \textit{ and } p_r'' \in t_i^{\blacksquare} \end{cases}$$

**Proof:** In order to uniquely determine the firing of any transition, their firing must be detected (part 1), i.e. if $\varphi C(\bullet, t_k) \neq 0$ then the firing of $t_k$ is detected using the discrete inputs-outputs information change $y_k - y_{k-1} = \varphi C(\bullet, t_k) \neq 0$, otherwise if the

$$\begin{bmatrix} \Phi(p_i') & & \\ & \ddots & \\ & & \Phi(p_j') \end{bmatrix} \textit{ is distinguishable from } \begin{bmatrix} \Phi(p_k'') & & \\ & \ddots & \\ & & \Phi(p_l'') \end{bmatrix}$$

then by using continuous information it can be determined the firing of $t_k$ .
The change that every firing produces in the output is unique (part 2). Since two transitions producing the same discrete input-output information, $\varphi C(\bullet, t_k) = \varphi C(\bullet, t_i)$ belongs to the same P-Component (because the places of P-components can have the same output symbol), then the property of distinguishability between the LS associated with $t_k^{\blacksquare}$ and the LS associated with $t_i^{\blacksquare}$ can be tested.

Since part 2 states that both systems are distinguishable from each other, then it can be determined if the places of $t_k^{\blacksquare}$ or the places of $t_i^{\blacksquare}$ are marked and consequently conclude which transition has been fired.

### 3.4 Diagnosability characterization

Detecting and isolating a fault in *LS* and *IPN* has been widely addressed in the literature. The results are identical in both cases for additive faults. A residue generator can be built in such a way that the subspace generated by the input can be twisted and placed into the kernel of the output map while the fault resides out of the kernel of the output map. These residue generators (Massoumnia et al., 1989; Hadjicostis et al., 1999; Ramirez et al. 2007) must be built and can be used to detect faults when it is possible. However, there could be cases when the faults cannot be isolated by the residue generators, but still can be isolated, as the following theorem states.

**Theorem 3.11** Let $\{F, N\}$ be a *SLS* and $\{F', N'\}$ be its Complete *SLS* where every fault $f_i$ all T-Components of the *IPN* in the *SLS* share transitions with the P-component containing the risk place $p_i \in {}^{\blacksquare}t_{f_i}$. If the pre risk $t_{ri}$, post risk $t_i$, and faulty $t_{f_i}$ transitions of every fault $f_i$ fulfill that:

1. $\forall t_j \in T$ is $D_H(t_i, t_j) < \infty$,
2. $t_{ri}$ and $t_i$ are event detectable or $t_{f_i}$ is event detectable

then the *SLS* is diagnosable.

**Proof.** Since $\forall t_j \in T$ is $D_H(t_i, t_j) < \infty$, then the firing of $t_i$ appears in all finite transition firing sequences. In order to fire $t_i$, the risk place $p_i$ must be marked. At this marking the fault $f_i$ (represented by the firing of $t_{f_i}$) could occur.

Notice that the moment when place $p_i$ is marked is detected since $t_{ri}$ is event detectable. Eventually either, the transition $t_i$ will be fired and detected (since $D_H(t_i, t_j) < \infty$ and $t_i$ is event detectable) and there is no fault in the *SLS*, or the *IPN* will be blocked. Since $\lambda(t_i) \neq \varepsilon$ then the symbol $\lambda(t_i)$ could be given to the *IPN*, if the firing of $t_i$ is detected, then there is no fault in the *SLS*, otherwise, the *SLS* is in a fault state, moreover, the fault $f_i$ occurred into the *SLS*.

## 4. Diagnoser design

The scheme used to detect and isolate faults when the system is working on-line is presented in Figure 4. Its funcionality is as follows, when the inputs manipulable and not manipulable are applied to the system, the event-detectability implies that generate an output change in the *IPN* model which contains the normal and faulty behavior, the diagnoser model is binary too and only contains the event detectability normal behavior. The transition $t_k \in {}^{\blacksquare}p_k$, where $p_k$ is a risk place in the *SLS*, is event detectable. When no fault is presented in the *SLS* the error between the *IPN* model and the diagnoser model will be zero. Now, if a fault occurs and the *IPN* is event-detectable then the error between the diagnoser and the *Complete SLS* will be different from zero. If a transition is confused and it is necessary to verify which *LS* is evolving, i.e. which place is marked, then the

distinguishable diagnoser is used. The later diagnoser gives the requiered event detectability; when the difference is not zero, it implies that a fault occurs in the *Complete SLS*.

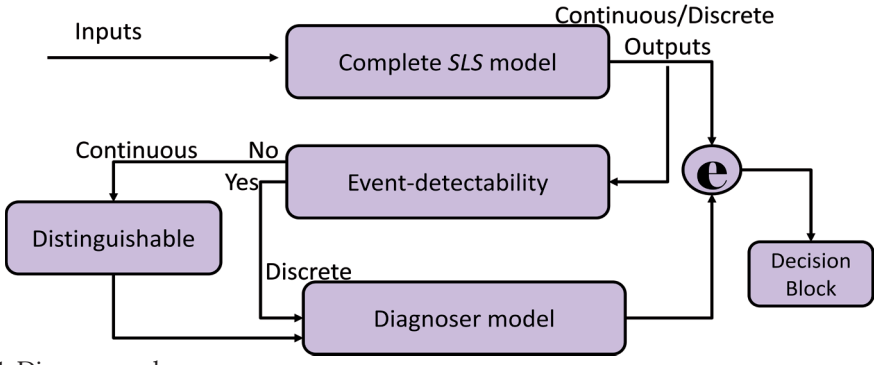



Fig. 4. Diagnoser scheme

## 5. Illustrative example

**Example 5.1** Consider the *SLS* of Example 2.11 where the normal and faulty behavior is depicted. Notice that the *IPN* is not input-output diagnosable using only the discrete information because when the system turns on and A appears, it is not possible to know which one is and in consequence if a fault occurs. The relative distance between the post-risk transition and the others in the *IPN* is finite. The *IPN* system has two places with the same symbol A. As both *LS* associated to $\Phi(P_3)$ and $\Phi(P_6)$ are distinguishable, if in the *IPN* A is detected, the distinguishable design immediately starts its operation and the currently evolving system will be detected. If the place $p_6$ is marked cannot occurs a fault. If the risk place $p_3$ is marked and as the relative distance is finite, $t_3$ must be fired, when no change in the *IPN* is detected implies that the fault $\Phi(P_{10})$ occurs in the system.

To diagnose the fault $\Phi(P_{11})$ the *IPN* gives enough information to know that the system arrives to a risk place, as the *LS* marked with the same symbol C in the *IPN* are distinguishable, this means that $\Phi(P_8)$ is distinguishable from $\Phi(P_{11})$, i. e. the extended system:

$$\bar{A} = \begin{bmatrix} -1 & 3 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -3 & -1 \\ 0 & 0 & 3 & -7 \end{bmatrix}; \; \bar{B} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 1 \end{bmatrix} \; and \; \bar{C} = [2 \quad -9 \quad -3 \quad -1]$$

is observable, controllable and does not have transmission zeros. It is easy to see if a fault occurs using the distinguishable design.

## 6. Conclusions

This chapter addresses the diagnosability problem in *SLS* represented by a family of linear

systems and an interpreted Petri net. It shows that although the results of diagnosability in *IPN* and *LS* can be used, the class of *SLS* that can be analyzed include those where neither the *SLS* nor the *IPN* are diagnosable.

The main idea behind of diagnosability is that the occurrence of a fault can be detected in the output because if the use of residue generator, distinguishability or the expected normal behavior is not carried out. These three ideas are introduced into the *IPN* as the solely concept of event detectability, thus when faulty, pre and post risk transitions are event detectable and the relative distance of post risk transition and other transitions is finite then the *SLS* is diagnosable.

The advantages of the proposed method are that the diagnosability characterization is structural and polynomial, the diagnoser converges to the fault in finite time and the *SLS* model captures the fact that several systems can work coupled or uncoupled, depending on the operation circumstances.

## 7. Acknowledgment

This work was supported by the National Council of Science and Technology of Mexico (CONACYT), Project No. 107195.

## 8. References

C. T. Chen, (1970) Linear System Theory and Design. Holt, Rinehart and Winston.

J. Desel and J. Esparza. (1995) Free Choice Petri Nets. Cambridge University Press.

G. K. Fourlas, K. J. Kyriakopoulos and N. J. Krikelis (2005) Fault Diagnosis of Hybrid Systems, in Proceedings IEEE international symposium on intelligent control, p.p. 832-837

George K. Fourlas (2009) Multiple Faults Diagnosability of Hybrid Systems. , in Proceedings of the 17th Mediterraneean Conference on Control & Automation, pp. 365-370

J. Gertler, M. Staroswiecki and S. Mengbing, (2002) Direct design of structured residuals for fault diagnosis in linear systems, in American Control Conference, vol. 6, pp. 4519-4524.

D. Gomez-Gutierrez, G. Ramirez-Prado, A. Ramirez-Trevino, and J. Ruiz-Leon. (2008) Joint state-mode observer design for switched linear systems, in IEEE International Conference on Emerging Technologies and Factory Automation, pp. 1408-1415.

C. N. Hadjicostis and G. Verghese. (2000) Power Systems Monitoring Using Petri Net Embeddings. IEE Proceedings C: Generation, Transmition, Distribution, vol. 147, no. 5, pp. 299-303.

C. N. Hadjicostis and G. Verghese. (1999) Structured Redundancy for Fault Tolerance in State-Space models and Petri Nets. Kybernetica, vol. 35, no. 1, pp. 39-55.

S. Hashtrudi-Zad, R. H. Kwong and W. M. Wonham, "Fault Diagnosis in Discrete Event Systems: Framework and Model Reduction", IEEE Transactions on Automatic Control, vol. 48, no. 7, pp. 1199-1212, 2003

M. Massoumnia, G. C. Verghese and A. S. Willsky, (1989) Failure Detection and Identication", in IEEE Trans. Autom. Control, vol. 34, no. 3 pp. 316-321.

A. Ramírez-Treviño, E. Ruiz-Beltrán, I. Rivera-Rangel and E. López-Mellado, (2007) On-line fault diagnosis of Discrete Event Systems. A Petri Net Based Approach, in IEEE Transactions on Science and Engineering. Vol. 1 No. 4, January 2007. pp 31-39.

I. Rivera-Rangel, A. Ramirez-Treviño, L.I. Aguirre-Salas, and J. Ruiz-León. (2005) Geometrical characterization of Observability in Interpreted Petri Nets. Kybernetika, 41, 553-574.

M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis, (1995) Diagnosability of discrete event systems, in IEEE Transactions on Automatic and Control, vol. 4, no. 9, pp. 1555-1575.

W. M. Wonham, (1979) Linear Multivariable Control: A Geometric Approach. Springer-Verlag.

17

# Fault diagnosis for complex systems using Coloured Petri Nets

L. Rodríguez, E. García, F. Morant, A. Correcher and E. Quiles
*Universidad Politécnica de Valencia*
*Spain*

## 1. Introduction

At present the research areas deeply studied because of security, reliability, viability and economy issues is the fault diagnosis area, which warrants some behavior states of a system, a machine or a process for detection, isolation and recovery faults, or even prevents these.
Currently one of the processes extensively studied in the area of fault diagnosis are the plants and chemical processes as well as systems for the study of renewable energy generation where both control and diagnostic systems for large number of variables become too complex and need new and robust techniques for the diagnosis of complex systems. These methods among others are the techniques for modeling discrete event (SED's) with Petri Nets (RDP's).
The study of these techniques has increased significantly over time due to the large number of applications that have been found. The formalism contributed by the PN's in concepts such as concurrency, mutual exclusion and resource sharing are of special interest and have provided greater capacity and power of representation in the resulting models as those carried out by Finite State Machines (FSM's), (Sampath et al., 1996).
Additionally, the PN's provide the ability of apply techniques of merging places that allow reducing size of the resulting models. This synthesis capacity in the resulting models is more accented with the Coloured Petri Nets (CPN) (Jensen, 1992), designing these ones with the general purpose of being a graphical structure based on PN's useful for specify, design and analyze concurrent systems that contribute to the possibility of applying merging techniques for representation of different concurrent subprocesses that coexist in the same PN graphical structure. The CPN's allow assigning functions in their arcs with lineal transformations capacity, which allows great functional variability to the final model. All these characteristics of synchronism and concurrency of the PN's, additionally to the merging techniques of the CPN's will give the necessary robustness to be applied to fault diagnosis in any complex system.
In this chapter it will be shown the functionality and advantage of the use of Latent Nestling Method of faults using Coloured Petri Nets (García et al., 2008), (Rodríguez et al., 2008) for the isolate and diagnosis faults in complex systems comparing it against other diagnostic techniques. Also, their mathematical formalization in discrete, continuous and hybrid systems and some application example. In the same way it will be presented the possible future research of this diagnostic tool in subjects as intermittent fault diagnosis (Correcher et

al., 2004), condition monitoring (Caselitz et al., 1996) and techniques of structuring as: folding and clustering (Keller, 2000).

## 2. Latent Nestling Method (LNM)

LNM was proposed according to the procedure used by (García et al., 2008) with the purposed of nestling faults into every place of the initial PN using a folding technique. These Petri Nets for the diagnostic methodology were called, Coloured Petri Net for Fault Diagnosis (DCPNs).
This methodology is defined as:

$$D = (P, T, Pre, Post, M_0, C, PLNf, Tf, PV f) \quad (1)$$

where,

- $P$ is a finite set of places.
- $T$ is a finite set of transitions.
- $Pre$ and $Post$ are input and output arc functions, with an additional argument $C_k$ that is the color of the transition firing $T_j$, thus:
  $Pre(P_i, T_j/C_k)$,
  $Post(P_i, T_j/C_k)$
  Nevertheless these functions are divided in two subsets, it depending of the transition type, behaviour normal transition or fault transition.
  $TF = Tf \cup Tr$
  where $Tf$ and $Tr$ are the fault and recovery transitions corresponding. As well
  $Pre = Pre^T \cup Pre^{TF}$
  $Post = Post^T \cup Post^{TF}$
  where the arc function are the following:
  $Pre^T: PLNf \times T \rightarrow \mathbb{N}$,
  $Post^T: PLNf \times T \rightarrow \mathbb{N}$,
  $Pre^{TF}: PLNf \times Tf \cup PVf \times Tr \rightarrow \mathbb{N}$,
  $Post^{TF}: PVf \times Tf \cup PLNf \times Tr \rightarrow \mathbb{N}$
- $M_0$ is the initial marking.
- $C$ is the colour set assigned to different identifiers. $C = N \cup f$. $N$ is the subset of coloured tokens representing the normal system behaviour.
- $f = \{f_1, f_2, \ldots, f_i\}$ is the subset of coloured tokens representing fault set.
- $PLNf \subseteq P$ is the subset of fault latent nestling places.
- $PVf \subset P$ is the subset of fault verification places.
- $TF \subset T$ is the transition subset including coloured functions.

**Definition 1** A normal transition in a DCPN is **enabled** if each place $PLNf_k$ in ${}^0T_j$ meets the condition:

$$m(PLNf_k) \geq Pre(PLNf_k, T_j) \quad (2)$$

The main idea to start is design of PNs for the correct dynamical behaviour of the system, applying the same modeling techniques that are used for generalized PNs. However for complex systems the synthesis capabilities of CPNs can be used in these first modeling

steps. The next procedure is use the knowledge expert for nestling the respective faults in each place latent nestling fault $PLNf_i$. These faults can be of simple or multiple nature, fault type $f_i$ or $f_if_k$. In Fig. 1. is shown the place of latent nestling fault $PLNf_i$, where the notation means the fault type $i$, $j$ and $p$ that belong to the subnets $q$ and $n$, with the next marking: $M(PLNf_k) = <\bullet n> + <\bullet q> + <f_i^q> + <f_i^n> + <f_j^n> + <f_p^n>$

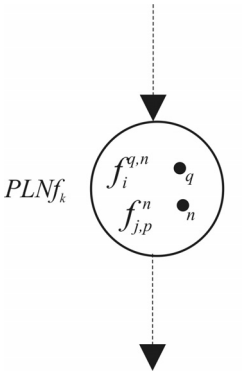



Fig. 1. Latent nestling place

After of the nestling process is necessary realized the trajectories of fault verification as well as fault recovery trajectories. This process is done by building a table, where for each reached marked of the system, exist a fault verification trajectorie by sensors installed.

**Definition 2** A fault or recovery transition in a DCPN is **enabled** if each place $PLNf_k$ in ${}^0TF_j$ meets the condition:

for $Tf$:

$$m(PLNf_k) \geq Pre(PLNf_k, Tf_j) \tag{3}$$

for $Tr$:

$$m(PVf) \geq Pre(PVf_k, Tr_j) \tag{4}$$

In Fig. 2. Is shown these trajectories, where if exist a faulty token in the nestling latent place $PLNf_k$, and verify the activation of a non expected reading $SROVnev$ drives to the marking of the fault verification place $M(PVf(<f_i^q>))$ by a token of colour $f_i^q$ that evidences the occurrence of this fault type. Consequently, if the fault $f_i^q$ is found in the $PVf$, is possible recovery path to initial $PLNfk$ while the expected values $SROVev$ are verified. The same way for the $f_i^n$ fault type.

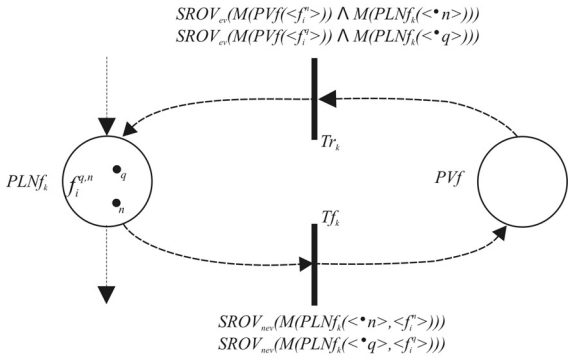



Fig. 2. Verification and recovery faults

The validation of these trajectories would be expressed as:
Verification of a $f_i^q$ fault type:

$$M(PLNf_k(<\bullet q>,<f_i^q>)) \\ [Tf_k=SROVnev(M(PLNf_k(<\bullet q>,<f_i^q>))) \\ >M(PVf(f_i^q)) \tag{5}$$

And $f_i^q$ fault recovery:

$$M(PVf(f_i^q)) \\ [Tf_k=SROVnev(M'(PLNf_k(<\bullet q>)) \wedge M(PVf(f_i^q))) \\ >M(PLNf_k(<\bullet q>,<f_i^q>)) \tag{6}$$

The diagnosability of this methodology is given by following expression:

$$\forall f_i^q \in f \; \exists \, (M(PLNf_k(<\bullet q>,<f_i^q>))) \\ [Tf_k=SROVnev(M(PLNf_k(<\bullet q>,<f_i^q>))) \\ > M(PVf(<f_i^q>)) \tag{7}$$

One question that needs to be asked, however, is that the processes and control systems nowadays are identified by their hybrid and complex nature, allowing the modeling of many methods in different areas of knowledge for process control and fault diagnosis. In this case numerous studies have worked to explain hybrid process in fault diagnosis using differents methodologies, for example (Gertler et al., 1998), (Chen & Patton, 1999), (Patton et al., 1999). As well, authors such as (David & Alla, 2005), analized fault models in Hybrid Petri Nets, other approximation use differential places for represented continuous places with negative markings in a methodology called Differential Petri Nets (Demongodin & Koussoulas, 1998).
For the methology proposal of Latent Nestling, is necessary include places of continuous or differential character, allowing the analysis of continuous dynamical variable. Following, the formal definition and the approaches of this new methodology integrating the use of continuous places. Likewise, an example that clarifies the concepts seen.

## 3. Latent Nestling Method in Hybrid Systems

### 3.1 Normal definition

A Hybrid Coloured Petri Net for fault diagnosis (*DHCPN*) is defined as:

$$DH = (P; T; Pre; Post; M_0; Co; C; PLNf; TF; PVf; OS; tempo) \tag{8}$$

Where $P$; $T$; $M_0$; $TF$; $PVf$; have the same definition as *DCPN*.
$Pre = PreT \cup PreTF$;
$Post = PostT \cup PostTF$
then,

$PreT : (P \times T) \rightarrow \mathbb{Q}$ ó $\mathbb{N}$,

$PostT : (P \times T) \rightarrow \mathbb{Q}$ ó $\mathbb{N}$,
$PreTF : (P \times Tf \cup PV f \times Tr) \rightarrow \mathbb{Q}$ ó $\mathbb{N}$,
$PostTF : (P \times Tr \cup PV f \times Tf) \rightarrow \mathbb{Q}$ ó $\mathbb{N}$
For every places set can be defined that:

$$P = P^D \cup P^C \tag{9}$$

As well as, PLNf ⊆ P y PVf ⊂ $P^D$.

Therefore $\mathbb{N}$ corresponds to the case for all $PLNf_i \in P^D$, and $\mathbb{Q}$ corresponds to the case where $PLNf_i \in P^C$. *C*, remains the coloured tokens set divided into normal behaviours marks "*N*" and representing faults marks "*f*". Being at the same time the normal behaviour marks can be of the discrete or continuous subset, as follows:

$$N = N^D \cup N^C \tag{10}$$

$C_0$: $P \cup T \rightarrow \{D,C\}$, is a composite function, that indicate for every place of the net if is a latent nestling place of discrete type (set $P^D$ y $T^D$) or continuous type (set $P^C$ y $T^C$).
*OS*: Is a set of operating states, and fault signatures.
This set is defined in the paragraph: trajectories of fault verification and fault recovery.
*Tempo*: is a delay function that associates a rational number to each transition that can evolve in time, where:

- if $f(T_j)=D$, $tempo(T_j)=d_i$= is a delay associate at the transition $T_j$, expressed in time units. As in the method defined in the previous chapter.
- if $f(T_j)=C$, $tempo(T_j)=\{V(T_j), d_i\}=\{V_j, h\}$, $V_j$ represent the maximum firing speed associated with the transition $T_j$ (David & Alla, 2005), and $h$ the firing frecuency represent the sampling time.

This delay function "*tempo*", is implemented for continuous places according to the model characteristics. If the markings and weights of the arcs are not of negative values, only be use the function $V_j$, represented the maximum maximum firing speed as a constant value according to the degree of *D*-enable. For this case, the function is implemented with a single discrete place associated to the continuous transition that represent the maximum firing speed. In the opposite, if the model has markings or negative arcs, be use the function $\{V_j, h\}$

with a discrete transition associated to a discrete place that is linked to the continuous transition that represents the maximum firing speed. In the last case the behavior of these places and continuous transitions are represented as (Demongodin & Koussoulas, 1998) with the names of differentials places and transitions.

**Definition 1.** A normal discrete transition in a DHCPN is enable if each place $PLNf_k \in P^D$ in ${}^{o}T_j^D$ meets the condition:

$$m(PLNf_k) \geq Pre(PLNf_k, T_j^D) \tag{11}$$

**Definition 2.** A normal continuous transition in a DHCPN is enable if each place $P_i \in P^C$ in ${}^{o}T_j^C$ meets the condition:

$$m(PLNf_k) \geq Pre(PLNf_k, T_j^C), \quad \text{if} \quad PLNf_k \in P^D \tag{12}$$

or

$$m(PLNf_k) > 0, \quad \text{if} \quad PLNf_k \in P^C \tag{13}$$

or

$$m(PLNf_k) \in \mathbb{Q} \quad \text{if} \quad PLNf_k \in P^C \quad \text{of differential type} \tag{14}$$

### 3.2 Initial Model

The initial model is the same as in the classic method, unlike that represent differential or continuous places where be model the continuous behavior of the system variables. The first step is to model the behavior of the process, both as discrete and continuous variables involved in the process, it uses the techniques of modeling temporary hybrid systems (David & Alla, 2005). Usually, the discrete processes represent the orders or actions to control the system, while the technological processes are continuous, discrete or mixed. As a second step must be a process of folding into subsystems according to the concurrent of these, this is the the coloured net process, that permit implement normal type marks by each subsystem concurrent global model. This folding process is done using the CPNs techniques. If the model allows it can be implemented directly in CPNs.

### 3.3 Fault Set Definition

This is done as in conventional nestling, identifying all the faults sets to diagnose in the system and making an allocation of these faults respect to some colored markings of the type $f=\{f_1, f_2, \ldots, f_i\}$.

Furthermore, this set should define continuous type faults to be determined according to the behavior of the residue and thresholds assigned by either the process or expertise.

If a fault $f_i$ occurs from an abnormal behavior of a continuously variable $h$, being the continuous place is influenced by a normal behavior mark $q$ contain in a $PLNf_k$. The fault is designated as a pair $< f_i^q, S_i >$, where $f_i$ is the fault occurred in the subnet $q$, and $S_i$ is the continuous operating state in which the fault ocurred.

### 3.4 Places of Latent Nestling

Latent nesting place are defined by the method of discrete type, confirming that all the faults in the system must be assigned exclusively to the set places $PLNf$. However, in a hybrid

system, if there is a continuous place $P_i^c$ which represents during a certain time t an operating, according to the state or states as a discrete place, the faults are assigned to this continuous place where $PLNf_i \in P^C$. This means that the generated faults by the anomalous behavior of the continuous variable somewhere $P_i^C$ are nestling in the same continuous place now called $PLNf_i^C$, because this hybrid character.
The representation of the continuous behaviour normal marks are the numeric text type, while the faulty marks and discrete behaviour normal marks are the same according to the method proposed in the previous chapter

### 3.5 Trajectorys of fault verification and fault recovery

These trajectories are defined only by the fault and recovery transitions, adding some restrictions to include the status of the places of normal behavior, and the marks of normal behavior. These restrictions are presented in the status and degree of transitions validation and complexity as for the construction of the fault transitions in continuous places.
**Definition 3.** A fault or recovery transition in a DHCPN is enable for discrete places if each place $PLNf_k^D$ *or* $PVf$ in ${}^{o}TF_j$ meets the condition:
For *Tf*:

$$m(PLNf_k) \geq Pre(PLNf_k, Tf_j) \tag{15}$$

For *Tr*:

$$[m(PVf) \geq Pre(PVf, Tr_j)] \wedge [m(PLNf_k) \geq Pre(PLNf_k, Tr_j)] \tag{16}$$

The possibility of exist continuous variables within the hybrid model, implies the possibility of perform an analysis to obtain new diagnoses on these same variables. The main idea is to use classical techniques of fault diagnosis based on models or based on heuristics, (Isermann, 1997). For example, if be use the technique based on quantitative models that is the residue generation and the subsequent evaluation.
Some techniques used in this area of residue generation are the parity equations, (Gertler, 1998), and observers (Chen & Patton, 1999).
To find the residues its neccesary obtain the dynamic model of operation of the continuous variables (typically in differential equations), and isolate the variables according to obtain a residue. Depending on the complexity can be represented in state variables, as in the hybrid PN analysis, (Demongodin & Koussoulas, 1998).
The idea in this new approach to diagnosis in hybrid systems, consists in obtain in every continuous place a series of residues of the form $r(t) = y(t) - \hat{y}(t)$, being $\hat{y}(t)$ the variable represented by the continuous place, and $y(t)$ the measured variable by the system or process in a real time. The residue is obtained directly in the continuous place, while the residual evaluation is realized in each fault and recovery transition using knowledge expert for defined a signatures fault and isolate the fault occurrence in the place *PVf*. Every one of possibility theseresidues $(r_i(k))$ will be evaluate respect to the $\tau_i$, set according to previous or heuristic knowledge.
To define a systematic approach of this Latent Nestling Method for Hybrid Systems, is necessary raise some new conditions to the continuous analysis called as operating states OS and fault signatures. This new approach will depend on the analized system as for the continuous places influence into themselves.

### 3.6 States of hybrid operation

Due to the continuous nature present in the hybrid models, its important to analyze the continuous places influence into themselves according of the system to treat. This influence in continuous places sites is an important factor in an effect known as coupling faults, which involves erroneous readings of faults by propagation of these faults, (García & Correcher, 2006). This factor is used to analyze the residues of continuous places in a more systematic way, and achieve a more effective fault isolation.

For every hybrid system exist three influence types according to the continuous places behaviour. Fig 3. a) Continuous isolated places b) Continuous places cascade influenced c) Continuous places cyclic influence.

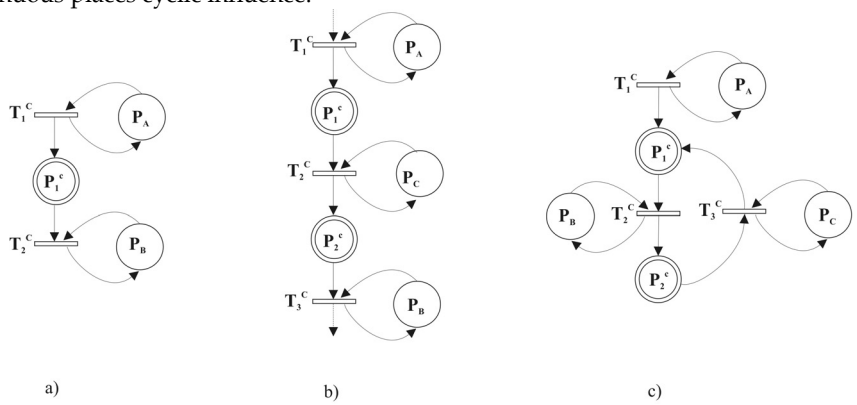



Fig. 3. Types of continuous influence according to the hybrid model

Also, exist an operating states for each continuous place, indicating the behavior of the continuous variable modeled. These operating states depend on the operation of the discrete places that control the continuous place and to include some signatures for the residual faults analysis.

A hybrid model has a set of operating states for the failure and recovery as well:

$$OS=OS_f \cup OS_r \tag{17}$$

Where $OS_f$ are the fault operating states, and $OS_r$ the recovery operating states.

(a) Continuous isolated places
These models usually have only one place continuous in the hybrid model with a single vector of operating states, but depending on the model, may be exist several continuous places not be influence into themselves, which would mean a vector of operating states for each continuous place. Where $OS_f=(os_l,...,os_k)$ being that $l$ and $k$ subscript correspond to the places $P_l^c$ and $P_k^c$, being $|OS_f|$ in this case the number of $P^c$ isolates in the model. Every $os_i$ is a vector that contain many operating states as signature faults for every contiuous place $P_i^c$, thus, $os_i=(Sf_1(k),..., Sf_m(k))$.
The $OS_r$ set has the same definition that $OS_f$, being the vector $os_i$ contains in this case recovery signature faults.

(b) Continuous places mutually influenced

- Cascade influence
  In these models of continuous places are influenced one to one, however the information flow is transferred in an open loop, meaning that the behavior of a continuous place $P_i^c$ directly influences the behavior of the continuous place $P^c_{i+1}$, successively, but not influence in the immediately preceding, $P^c_{i-1}$.
- Cyclic influence
  These model are characterized by has flow behavior in closed or feedback, meaning that exist a mutually influence in every continuous places according to control of the discrete places.
  Both, continuous places cascade influenced as a continuous places cyclic influence exist in the same manner as for every continuous isolated places a single fault transicion $Tf_i$, which defines a number of *Pre* arcs for this $Tf_i$ as:

$$\Pr e^{Tf_i} = \sum_{j=x}^{n} (P_j^c \times Tf_i) \tag{18}$$

  Where, $x$ is the initial continuous place influenced, and $n$ is the last continuous place influenced
- Mixed influence
  This models may have mixed operation structures, as isolates , cascades or cyclic.

To find the operating states of the continuous places mutually influenced greater ease, it is necessary to make a table called "table of continuous places influenced." This table will have the number of operating states of the model. Also, obtain the fault signatures according to the discrete places that influencing each continuous place. This table shows the main influence of a continuous place over the other continuous place according to the discrete places that interact with the continuos places, allowing obtain the fault and recovery transitions.

In every continuous places exist five operating states that depend to the transition enable degree, according to the discrete influence, these are: increase, decrease, mixed or resting.

In table 1 is shown the influenced for the models in figure 3. The X state means that this combination of discrete events its not possible by the control.

| $P_B$ | $P_A$ | $P_1^c$ | |
|---|---|---|---|
| 0 | 0 | $Sf_1$ | $os_1$ |
| 0 | 1 | $Sf_2$ | $os_2$ |
| 1 | 0 | $Sf_3$ | $os_3$ |
| 1 | 1 | X | |

a)

| $P_A$ | $P_B$ | $P_C$ | $P_1^c$ | $P_2^c$ | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | |
| 0 | 0 | 1 | $Sf_1$ | $Sf_2$ | $os_1$ |
| 0 | 1 | 0 | $Sf_3$ | $Sf_4$ | $os_2$, $os_3$ |
| 0 | 1 | 1 | X | X | |
| 1 | 0 | 0 | $Sf_5$ | $Sf_6$ | $os_4$, $os_5$ |
| 1 | 0 | 1 | X | X | |
| 1 | 1 | 0 | X | X | |
| 1 | 1 | 1 | X | X | |

b)

| $P_A$ | $P_B$ | $P_C$ | $P_1^c$ | $P_2^c$ | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | |
| 0 | 0 | 1 | $Sf_1$ | $Sf_2$ | $os_1$ |
| 0 | 1 | 0 | $Sf_3$ | $Sf_4$ | $os_2$ |
| 0 | 1 | 1 | X | X | |
| 1 | 0 | 0 | $Sf_5$ | $Sf_6$ | $os_3$, $os_4$ |
| 1 | 0 | 1 | X | X | |
| 1 | 1 | 0 | X | X | |
| 1 | 1 | 1 | X | X | |

c)

Table 1. Continuous places mutually influenced

The table 1 a) shown the fault signatures by the behaviour of the continuous place $P_1^c$, where:

$os_1=(Sf_1(k),Sf_2(k),Sf_3(k))$
$Sf_1$= Fault signature for resting state
$Sf_2$= Fault signature for increase state (filling)
$Sf_3$= Fault signature for decrease state (emptying)

Determining a fault type $<f_i, S_i>$, where each $S_i$ correspond to a fault signature. Analyzing b), is interesting to note that the operating state $os_1$ is mixed, being that $P_1^c$ influence the behaviour of $P_2^c$, this influence indicate that the residue must be analized together for a better isolation faults. Likewise, tha fault signatures $Sf_3$ and $Sf_4$ are part of different operating states, one place in resting and the other place in decrease. Analyzing c) be shown that the fault signatures $Sf_3$ and $Sf_4$ are part of the same operating state, and is necessary analizing its residues together for isolate the faults.

Figure 4 shows the same figure 3 but with fault transitions implemented according to previous analysis.

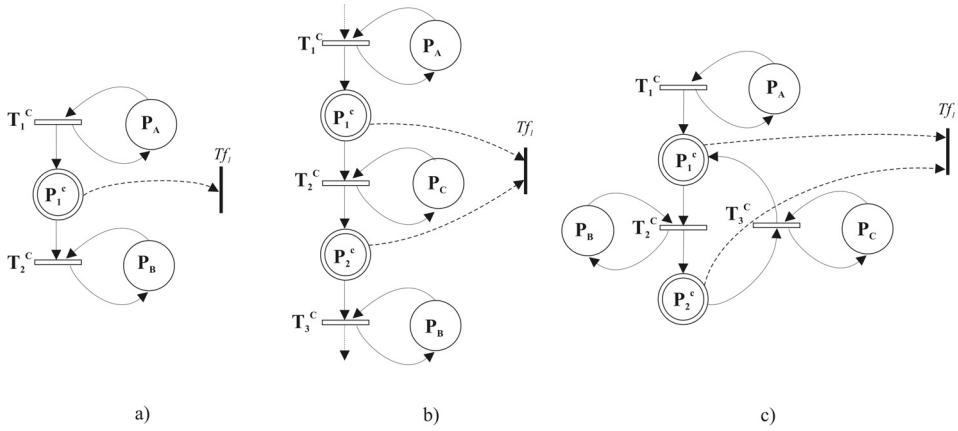



Fig. 4. Fault transitions of continuous places influenced

### 3.7 Fault signatures

These fault signatures represent the faults in the place *PVf* isolate of recurrent manner according to the residual behavior like using the threshold for every operating state. For example:

$$Sf_n(k)=\begin{cases} < f_i, s_n > if \rightarrow r_i(k) > \tau_i \\ < f_i f_k, s_n > if \rightarrow r_i(k) < \tau_j \\ \vdots \\ < f_m, s_n > \cdots \end{cases} \tag{19}$$

For this case the detection and isolation of individual faults $f_i$, or simultaneous faults $f_i f_k$, is determinated by the dynamic conditions in the discrete or continuous marking and consequently in each state reached by the system. Likewise, by the set of not expected

readings from discrete sensors and the signatures faults, according to the current operating state in the continuous state.

As seen in the LNM section, exist a set of not expected values of sensor readings $SROV_{nev}(M(k))$, and a set of expected values of sensor readings $SROV_{ev}(M(k))$ for a given discrete marking, that permit associate the fault verification from latent nestling place *PLNf* to the verification place *PVf*, or otherwise to recovery faults.

Due to the possibility of include faults from the continuous dynamics, the set $SROV_{nev}(M(k))$ will include a fault signature $Sf_n(k)$ in a case of single continuous place, or a $os_i$ fault signatures vector in a mutually influenced places, according to the faults obtained by residues, by the dynamic behavior of the continuous place $P_i^c$ and the place of latent nestling faults *PLNf* that influence this continuous dynamic.

Also, include a recovery signature $Sr_n(k)$ to the normal behavior of the residue, according to the dynamic behavior of the continuous place $P_i^c$ and the latent nestling places $PLNf_n$ that influence this continuous dynamic. The recovery signatures are defined in the same manner as the fault signatures described above, changing the label "*f*" of fault by "*r*" for recovery. For a more compact notation in terms of fault and recovery transitions, the enable "E" for any transitions is given as:

For *Tf*:

$$Ef=SROV_{nev}(M_k),\ Sf_n(k) \tag{20}$$

For *Tr*:

$$Er=SROV_{nev}(M_k),\ Sr_n(k) \tag{21}$$

Finally, to define the fault trajectory traced from a continuous place $P_i^c$, which contains a faulty mark of this type $<f_i,S_n>$, a continuous mark of normal behaviour $<h>$, verified the discrete state with a behavior normal mark $<\bullet q>$, and is influenced by a residue outside of a designated threshold, would be expressed the form:

$$\begin{gathered}(M(PLNf_k(<\bullet q>)) \wedge M(P_i^c(<h>,(f_i,S_n))))\\ [Tf_j/\ Sf_n(k)(M(PLNf_k(<\bullet q>)) \wedge M(P_i^c(<h>,(f_i,S_n))))\\ > M(PVf((f_i,S_n)))\end{gathered} \tag{22}$$

The figure 5 represents this behaviour

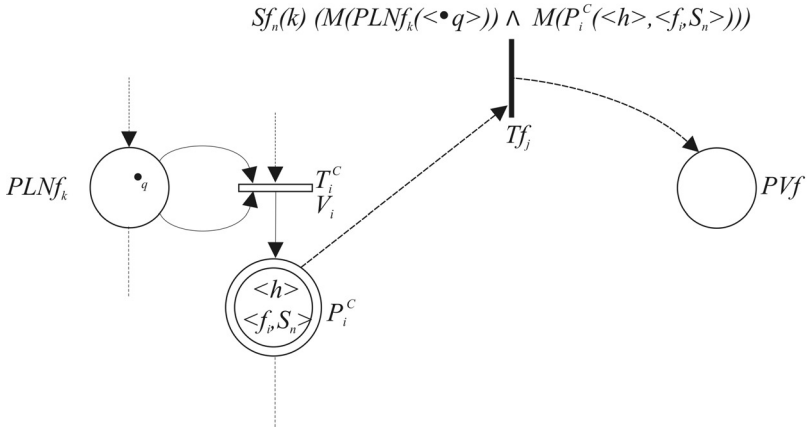



Fig. 5. Trajectory for fault verification (abnormal behavior of the residue in a continuous variable)

Eventually, the same fault $(f_i,S_n)$, can experiment a recovery process to the origin place $P_i^c$. In this way the model can be receptive to the treatment of intermittent faults. This recovery is expressed as:

$$\begin{gathered} M(PVf((f_i,S_n))) \\ [Tr_j / Sr_n(k)\ (M'(PLNf_k(<\bullet q>))\ \wedge\ M'(P_i^c(<h>)\ \wedge\ M(PVf((f_i,S_n)))) \\ > M(P_i^c(<h>,(f_i,S_n))) \end{gathered} \tag{22}$$

The figure 6 represents this behaviour.

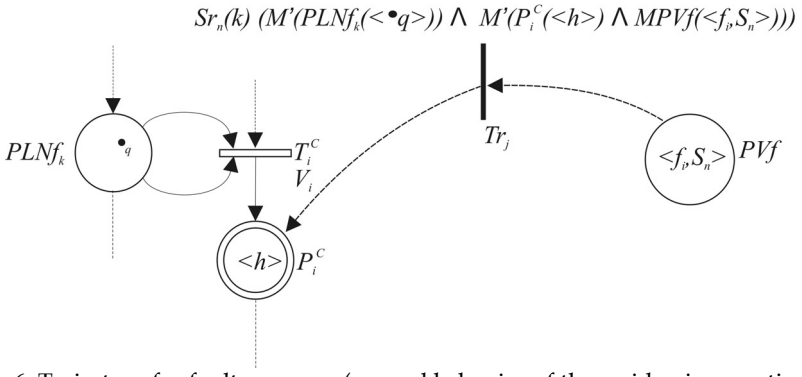



Fig. 6. Trajectory for fault recovery (normal behavior of the residue in a continuous variable)

### 3.8 Diagnosability of model

The diagnosability concept is maintained according to the previous paragraph, but be must include the fault signatures for each operating state of the continuous places analyzed $P_i^c$.

$$\begin{gathered} \forall f_i \in f \quad \exists (M(PLNf_i^c(<h>,<f_i,S_n>))) \\ [Tf_j / Sf_n(k)(M(PLNf_i^c(<h>,<f_i,S_n>))) \\ >M(PVf(<f_i,S_n>)) \end{gathered} \tag{23}$$

Likewise, its necessary satisfy the condition that at least one fault signature $Sf_n(k) \subseteq os_i$ must exist for each continuous place $P_i^c$.

$$\forall\ PLNf_i^c \in P^c\ \exists Sf_n(k) \subseteq os_i \tag{24}$$

### 3.9 Methodology example

The example system consists of a liquid storage tank, where it has: A storage system or tank, 3 actuators (2 pass valves, 1 mixer), 3 sensors (2 flow binary, 1 level type ultrasound). In Figure 7 shows the physical structure of the system.

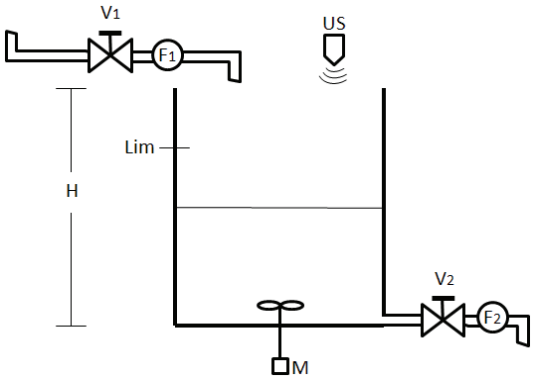



Fig. 7. Example of hybrid system

The process starts giving orders to open valve $V_1$ for filling tank with a flow ratio 2v.u/t.u (volume units per time unit) until the position $Lim = 30$ indicated in the figure (this position is a level indicator for the discrete measured by ultrasonic sensor). Then, the mixer $M$ is activated during $t_1 = 20$ seconds and close valve $V_1$ for not deposit more product. Finally, its necessary opening valve $V_2$ to empty the tank with a flow ratio 3v.u/t.u, and deactivate the mixer. Both input and output flow is a fixed ratio, which indicates that the function of filling and emptying is linear. In the real model is used $K\sqrt{h(t)}$, as an outflow, but in the previous simulation, the flow ratio is indicated above. The process runs on a cyclical mode. In figure 8 can be observed the hybrid model using the Sirphyco software (David & Alla, 2005).

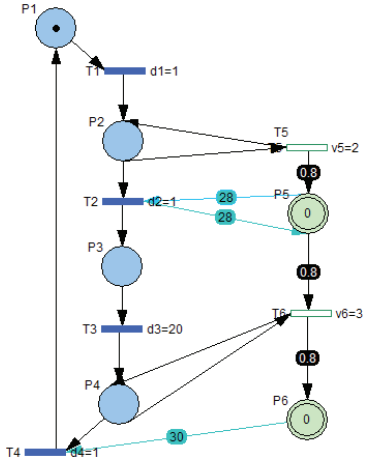



Fig. 8. Hybrid model using Sirphyco tool

To analyze the behavior of discrete dynamic system, will be to obtain four reachable markings o the normal behavior thus:
$M_0$= initial condition, close valve $V_1$

$M_1$= the valve $V_1$ is ordered to open
$M_2$= the valve $V_1$ is ordered to close, and the mixer is activated
$M_3$= the valve $V_2$ is ordered to open, and the mixer is deactivated
Getting the initial vector of reachable markings for the system. *$M_0$= ($M_0$, $M_1$, $M_2$, $M_3$.)
The dynamic analysis of the continuous system is governed by a simple differential equation.

$$\frac{dh(t)}{dt} = \frac{1}{A}(q_1(t) - q_2(t)) \tag{25}$$

Where A is the area of the tank, $\frac{dh(t)}{dt}$ the height variation respect to the time, *q1* and *q2* the input and output flow for the valves $V_1$ and $V_2$ respectively.
To define the faults is used to knowledge of the proposed system.
Where the faults can be: stuck valves fault, leakage in the tank fault, sensors fault. Classifieds these faults in identifiers using coloured marks, we have:
$f_1$= stuck open valve $V_1$ fault
$f_2$= stuck close valve $V_1$ fault
$f_3$= stuck open valve $V_2$ fault
$f_4$= stuck close valve $V_2$ fault
$f_5$= leakage in the tank fault
$f_6$= level sensor fault
The readings of discrete behavior sensor are:
$srov_{11}(M_k)=\{F_1,NF_1\}$
$srov_{12}(M_k)=\{F_2,NF_2\}$
Using the level sensor as a measured discrete
$srov_2(M_k)=\{L,NL\}$
Where *L*= Exist a level and *NL*= the tank is empty.
In table 2 shown the faults according the sensors readings and discrete marking

| $F_1$ | $F_2$ | L | $M_0$ | $M_1$ | $M_2$ | $M_3$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | $SROV_{ev}$ | $f_2$ | $f_6$ | $f_4f_6$ |
| 0 | 0 | 1 | $f_6$ | $f_6f_2$ | $SROV_{ev}$ | $f_4$ |
| 0 | 1 | 0 | $f_3f_6$ | $f_2f_3f_6$ | $f_3f_6$ | $f_6$ |
| 0 | 1 | 1 | $f_3$ | $f_2f_3$ | $f_3$ | $SROV_{ev}$ |
| 1 | 0 | 0 | $f_1f_6$ | $f_6$ | $f_1f_6$ | $f_1f_4f_6$ |
| 1 | 0 | 1 | $f_1$ | $SROV_{ev}$ | $f_1$ | $f_1f_4$ |
| 1 | 1 | 0 | $f_1f_3f_6$ | $f_3f_6$ | $f_1f_3f_6$ | $f_1f_6$ |
| 1 | 1 | 1 | $f_1f_3$ | $f_3$ | $f_1f_3$ | $f_1$ |

Table 2. Fault behaviour according the discrete sensor readings

For continuous analysis must be examine continuous variables that influence the process (in this case one variable) and discrete sensors that influence this continuous place. For this case there is a single continuous place of isolated type, implying that exist a single operating state and a series of fault signatures for each discrete place that influence the behavior of the continuous place.

$OSf=osf_5$. The operating state $osf_5$ corresponds to the vector that contain the number of operating states of the continuous place $P_5^c$ identified with a fault signature for each fault as follows:
$mf_5=(Sf_2(k),Sf_3(k),Sf_4(k))$, known that $PLNf_2$, $PLNf_3$, $PLNf_4$ are the places that influence the behavior of the continuous place.
Due to the presence of the continuous type sensor for the height measured, is possible a comparison between the measured height $h$ and the estimated height $h'$.
Using the equation 25:
Case 1: first operating state, increase "filling"

$$h' = \frac{1}{A}\int q_1(t)\cdot dt \tag{26}$$

Where the residue $r_1$ is obtained:
$r_1=h-h'$

$$Sf_2(k) = \begin{Bmatrix} \langle f_5, S_2\rangle if \rightarrow r_1 > \tau_{11} = 0.3 \\ \langle f_6, S_2\rangle if \rightarrow r_1 < \tau_{12} = -1 \end{Bmatrix}$$

$$Sr_2(k) = \begin{Bmatrix} \langle f_5, S_2\rangle if \rightarrow r_1 > \tau_{41} = 0.15 \\ \langle f_6, S_2\rangle if \rightarrow r_1 < \tau_{42} = -0.5 \end{Bmatrix}$$

Case 2: second operating state "resting"
Where the height is a constant and the residue $r_2$ is obtained:
$r_2=h-h'$

$$Sf_3(k) = \begin{Bmatrix} \langle f_5, S_3\rangle if \rightarrow r_2 > \tau_{21} = 0.1 \\ \langle f_6, S_3\rangle if \rightarrow r_2 < \tau_{22} = -0.5 \end{Bmatrix}$$

$$Sr_3(k) = \begin{Bmatrix} \langle f_5, S_3\rangle if \rightarrow r_2 > \tau_{51} = 0.08 \\ \langle f_6, S_3\rangle if \rightarrow r_2 < \tau_{52} = -0.4 \end{Bmatrix}$$

Case 3: third operating state, decrease "emptying"

$$h' = \frac{1}{A}\int q_2(t)\cdot dt \tag{27}$$

Where the residue $r_3$ is obtained:
$r_3=h-h'$

$$Sf_4(k) = \begin{Bmatrix} \langle f_5, S_4\rangle if \rightarrow r_3 > \tau_{31} = 1 \\ \langle f_6, S_4\rangle if \rightarrow r_3 < \tau_{32} = -0.4 \end{Bmatrix}$$

$$Sr_4(k) = \begin{Bmatrix} \langle f_5, S_4\rangle if \rightarrow r_3 > \tau_{61} = 0.8 \\ \langle f_6, S_4\rangle if \rightarrow r_3 < \tau_{62} = -0.3 \end{Bmatrix}$$

The thresholds set $\tau = (\tau_{11},\ldots,\tau_{32})$ are given by knowledge expert and it analyze is according to different factors as: hysteresis, disturbances, noise, as well as the sensor sensitivity and sensor resolution.
Just as there are a fault operating states for each continuous place, too there are a recovery operating states. In these recovery states the $\tau$ values changes because the sensor hysteresis. For example, if $r_1=0.4$ when the process is filling, the isolate and recovery fault $f_5$ are given by the expression:
The fault isolation $f_5$ in this condition occurs if:

$$(M(PLNf_2(<\bullet n>)) \wedge M(P_5^c(<h>,(f_5,S_2))))$$
$$[Tf_3 / r_1 > 0.3\ (M(PLNf_2(<\bullet n>)) \wedge M(P_5^c(<h>,(f_5,S_2))))$$
$$> M(PVf((f_5,S_2)))$$

The fault recovery $f_5$ in this condition occurs if:

$$M(PVf((f_5,S_2)))$$
$$[Tr_3 / r_1 < 0.15\ \ (M'(PLNf_2(<\bullet n>)) \wedge M'(P_5^c(<h>) \wedge M(PVf((f_5,S_2))))$$
$$> M(P_5^c(<h>,(f_5,S_2)))$$

At it is observed , the diagnosis system is able to detect the isolate fault of individual type $f_1$, $f_2$, $f_3$, $f_4$, $f_6$, and simultaneous type $f_1f_6$, $f_1f_3$, $f_1f_4$, $f_1f_3f_6$, $f_1f_4f_6$, $f_2f_3$, $f_2f_3f_6$, $f_4f_6$, $f_6f_2$, as well as process fault of type $(f_5,S_2)$, $(f_6,S_2)$, $(f_5,S_3)$, $(f_6,S_3)$, $(f_5,S_4)$, $(f_6,S_4)$.
In figure 9 be shown the final model for the tank example

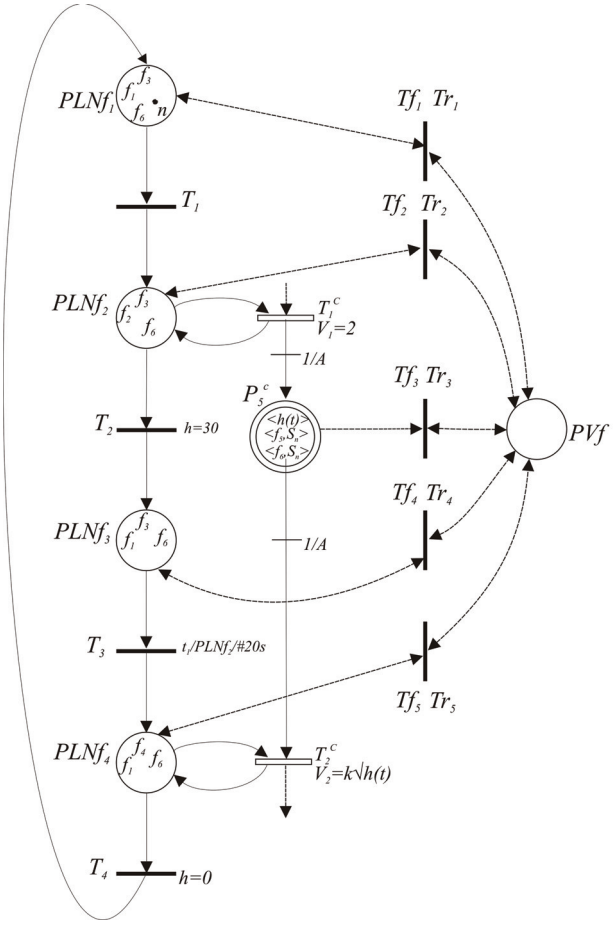



Fig. 9. DHCPN model (example of filling tank)

## 4. Conclusions

The method shows the reduction and simplicity of the system models are discrete, continuous or hybrid, giving them characteristics of readability, implementability treatability and no matter how many sensors to treat or how many faults to diagnose; imposible features to obtain with other methodologies such as MEFs.

The hybrid nestling technique shows the need to analyze the residues with the information of the discrete state in normal behavior for characterize the type of fault, its location and subsequent isolation.

Operating states, and the influence tables of continuous places, offer an overview of the system's behavior as sharing your information, being this information a continuous variable to treat in the model. This overview provides the possibility of locating the fault transitions thus analyze the fault coupling, to avoid false warnings in the verification place.

The Latent Nestling Methodology focused in continuous and hybrid systems presents an excellent and clear solution to fulfill the objectives of diagnosis and isolation for any faults type that may arise in the system.

## 5. References


Caselitz, P.; Giebhardt, J.; Mevenkamp, M. Development of a Fault Detection System for Wind Energy Converters, *Proceedings of the EUWEC '96*, Göteborg, pp. 1004 – 1007.

Chen, J. & Patton, R. J. Robust model-based fault diagnosis for dynamic systems *Kluwer Academic Publishers,* 1999.

Correcher, A.; García, E.; Morant, F.; Quiles, E. & Blasco. R. Intermittent Failure Diagnosis based on discrete event models, *Proceeding of 7'Th Workshop On Discrete Event Systems WODES04*, pp. 151-157, 2004.

David, R. & Alla, H. Discrete, Continuous, and Hybrid Petri Nets*, Springer-Verlag*, Berlin. 2005.

Demongodin, I. & Koussoulas, N. Differential Petri nets: Representing Continuous Systems in a Discrete Event World, *IEEE transactions on Automatic Control.* 1998*, 38*, pp. 573-579

García, E.; Morant, F.; Blasco, R.; Correcher, A.; Quiles, E. Centralized Modular Diagnosis and the Phenomenon of Coupling, *Workshop On Discrete Event Systems WODES02,* Zaragoza, Spain October 2002.

García, E.; Rodríguez., L.; Morant., F.; Correcher., A.; Quiles., E. & Blasco, R. Fault Diagnosis with Coloured Petri Nets Using Latent Nestling Method, *Proceedings of ISIE08 Cambridge UK,* 2008.

García, E.; Rodríguez., L.; Morant., F.; Correcher., A. & Quiles., E. Latent Nestling Method: A new fault diagnosis methodology for complex systems *IECON08, The 34th Annual Conference of the IEEE Industrial Electronics Society, Orlando, Florida, U.S.A,* 2008.

Gertler, J. Fault Detection and Diagnosis in Engineering Systems, *Marcel Dekker,* 1998.

Isermann, R. Supervision, fault-detection and fault-diagnosis methods -- An introduction *Control Engineering Practice,* 1997*, 5*, pp. 639 - 652

Jensen, K. Coloured Petri Nets 1: Basic Concepts, Analysis Methods and Practical Use*, Springer-Verlag,* 1992 Berlin, Germany.

Keller, W. Petri Nets for Reverse Engineering, *Universidad de Zurich,* 2000.

Patton, R. J.; Lopez-Toribio, C. J. & Uppal, F. J. Artificial intelligence approaches to fault diagnosis for dynamic systems, *International Journal of Applied Mathematics and Computer Science,* 1999*, 9(3)*, pp. 471-518.

Rodríguez., L.; García., E.; Morant., F.; Correcher., A. & Quiles, E. Application of Latent Nestling Method using Coloured Petri Nets for the Fault Diagnosis in the Wind Turbine Subsets *Proceedings of ETFA'08, Hamburg, Germany,* 2008.

Sampath, M.; Sengupta, R.; Lafortune, S.; Sinnamohideen, K. & Teneketzis, D. Failure diagnosis using discrete-event models, *Control Systems Technology IEEE Transactions on*, 1996, 4, pp. 105-124.

# Modelling and Fault Diagnosis by means of Petri Nets. Unmanned Aerial Vehicle Application

Miguel Trigos [1,2] , Antonio Barrientos [1] ,
Jaime del Cerro [1] and Hermes López [2]
[1] *Universidad Politécnica de Madrid, (Robotics and Cybernetics Group)*
[2] *Universidad Santo Tomas de Bucaramanga, (Mechatronics Engineering Faculty)*
*Spain-Colombia*

## 1. Introduction

The safe and reliable operation of technical systems is very important not only for the protection of humans but also for the protection of environment and economic investments.
The proper functioning of these systems has profound impact on production costs and product quality. Early fault[1] detection is critical in preventing a deterioration of behavior, damage to equipment or human life. The diagnosis must then help to make correct decisions in emergency actions and repairs.
This necessity has motivated the Robotics and Cybernetics group of Universidad Politécnica de Madrid to develop a methodology for developing embedded FD systems.
Techniques of Fault Diagnosis (FD) have been usually developed within a large area of research at the intersection of control and systems engineering, Artificial Intelligence, Mathematics and Statistics applied to fields such as Chemical, Electrical, Mechanical and Aerospace Engineering.
Due to FD methodology was initially developed for discrete event systems (DES's), an adaptation to the hybrids (composed of discrete and continuous processes) has been required.
Petri Nets (PN) have been the tool used to build the model and diagnoser, due to it is an excellent platform, which solves the limitations of combinational explosion presented in previous work of FD using to model finite state machines (FSM).
The FD algorithm presented here, begins with the definition of the PN model of each one of the system components, which must integrate the normal and failure operation modes.
Next step consist of building the general integration model of the system, it will support the construction of the diagnoser, who is responsible for overseeing the system in an online way

[1] Often, the term failure is used to denote a complete operational breakdown, whereas the term fault is used to denote any abnormal change in behavior; in this chapter we will use the two terms synonymously.

and informing the operator of the presence of a fault. The construction is a simple and robust process; its main advantages are the simultaneous detection of failures and the flexibility to expand its application to another components.
This tool has been implemented in several industrial applications, such as a ventilation system, heating and air conditioning systems (Trigos & Garcia, 2008 (A)), and liquids packaging processes (Trigos & Garcia, 2008 (B)) among others, but in this chapter, it is applied to a novel application: "Unmanned Aerial Vehicle (UAV)".
The proposed FD method is suitable for this application due to the hybrid nature of the unmanned aerial vehicles (UAV) and their high complexity, which requires a fault detection system.
The new legislative trends in the use of UAS (Unmanned Aerial System) will probably require having security systems where FD techniques are applicable. Furthermore, based on the report about reliability of UAVs in the military field of United States (Office of the Secretary of Defense USA, 2003), can be summarized that the UAVs are highly vulnerable not only to unexpected mishaps on the devices that make up the system (aircraft and control station) but also to the test environment.
Usually, the causes of these problems are unknown, but in addition to this, there is a lack of methods to prevent these failures. This problem is intrinsic to the UAV due to they have strong mechanical requirements and the consequences of a small failure can be enormous in comparison to ground vehicles.
In section 2 of this chapter, a state of the art about fault diagnosis is presented, starting with the work developed in the context of discrete event systems, connecting to continuous and finally hybrid systems. Section 3 summarizes the theory of Petri nets, due to they are intensely used in the work. Section 4 describes the methodology for building the model and the diagnoser by using PN applied to FD hybrid systems.
The application used to deploy the FD method is an unmanned aerial vehicle which is described in Section 5; it highlights important concepts in the operation of UAVs and data reliability in the military. After that, a model and diagnoser are constructed. Finally, section 6 sets out the conclusions of this investigation of FD in the field of UAVs, which is an excellent platform for implementing the tool.

## 2. State of the Art of Fault Diagnosis

The fault diagnosis is one of the major areas of research in Automatic and Control Engineering. Automatic processes are more demanding and complex, by this reason, fault diagnosis is analyzed from different fields. Algorithms for detection and isolation of faults can be classified in two major groups: related to the dynamics involved in the process and algorithms applied to processes of continuous and discrete dynamics. Real processes are composed of elements of the two dynamics, continuous and discrete, known as systems or processes hybrid. To expand the state of the art of researches in continuous systems, consulting (Venkatasubramanian et al., 2003).
In fault diagnosis of DES`s exist developments implemented by means of Regular Languages, State Graphs, Finite State Machines (FSM's) (Sampath et al., 1995) and the most used, Petri Nets (PN) (Ramirez et al., 2007). Also, there are researches where the benefits of FSM's and PN are mixed (Giua & Seatzu, 2005) (Chung & Jeng, 2003) (Ushio et al., 1998). The basis of the works mentioned below is made of FSM's (Sampath et al., 1995). This model

has one major limitation is that the number of states of the composition model, is given by the multiplication of the events of the system components, leading to if the components of systems increases, this construction is impossible of realize. In general, this methodology has several drawbacks: it is rigid (the failures have to happen in a certain way), only allows the diagnosis of one fault, for multiple failures, simultaneous and dependents can not be applied, and finally the biggest disadvantage is combinational explosion, this means that only can be applied to small processes, when the complexity of the process increases, it is impossible to apply this methodology.

Other contributions in line with DES's are developed by (Giua & Seatzu, 2005) (Chung & Jeng, 2003) (Ushio et al., 1998). These researchers have in their development a combination of tools, the model built with PN and diagnosis made with FSM's. To work (Chung & Jeng, 2003) (Ushio et al., 1998), the disadvantages given by (Sampath et al., 1995) are held almost entirely. (Giua & Seatzu, 2005) In the construction of the diagnoser have a better harnessing the mathematics power of PN, but ultimately the problem of combinatorial explosion is presented yet. It also presents the work of (Ramirez et al., 2007), the model is made with PN Interpreted, gives a better use to mathematic power of PN; Presents a systematic algorithm for constructing the model and diagnoser, its diagnosis is difficult because only identify a fault and its model of PN enters a sink state (deadlock). Finally, there is research (Genc & Lafortune, 2006), it makes fault diagnosis using PN with limited places, this technique is complex to implement and less possible to apply to industrial processes with medium level of complexity.

In Fault Diagnosis of Hybrid Systems, investigations can be classified according to the techniques used in its implementation, there are tools where already have made high progress, such as: Hybrid Automata, Hybrid Petri nets, among others, and other have not defined a specific technique and on the contrary, do FD by mean of combining different techniques.

The work cited by (Krogh, 2002) is a document that diagnosis dynamic complex systems, which continuous systems are examining with Supervisory Controller, experimenting partial or final failures on the devices of the system. (Zhao et al., 2005) conducted one of the most interesting applications developed to date in FD of hybrid systems; all work is carried out in the paper feeder of a Xerox printer. His contributions are great because it makes a hybrid integration of discrete and continuous FD techniques: Hybrid automata, Timed Petri Nets, Fault Trees and signal processing techniques that together solve a problem of diagnosis. (Narasimhan et al., 2000) works FD on hybrid systems combining model-based diagnosis with signal processing. (Fourlas et al., 2005) discusses the notion of diagnosis of hybrid systems in the workspace of Hybrid Automata, other works that guide its development from DES's to Hybrid Systems are the (Cassandra, 2002) and (Krogh, 2002). They base their work on (Henzinger, 1996) and discrete analyze and hybrid system control.

In the area of fault diagnosis of UAS (Unmanned Aerial Systems), according to (Hayhurst et al., 2006), the dangers that may represent an unmanned aircraft, is related to three key domains: design domain, flight crew domain and operational domain. In these domains can reveal hazards such as: impacts on ground with collateral damage to persons and property, and midair collision with manned aircraft or another UAS. Although at first instance it seems that the problems are the same as a manned aircraft, it must need great attention to the risks involved in the separation of the cabin of the aircraft.

From the viewpoint of fault diagnosis, the majority of investigations (Mancini et al., 2007) (Elgersma & GlavaSki, 2001) (GlavaSki & Elgersma 2001) are focused on assessing the faults in the hardware located on the aircraft ( Bonfa et al., 2006) (Heredia et al., 2005) (Zhang et al., 2006) (Bateman et al., 2007) (sensors and actuators), but must take into account failures regarding to links communication and the control station. On the other hand, (GlavaSki & Elgersma, 2001) (Cork et al., 2005) (Bateman et al., 2007) (Drozeski et al., 2005) focus your efforts on identifying failures and find a reconfiguration of the control system to bring the aircraft a normal operating state or in the worst case abort the mission. Most of the techniques used are based on parameter estimation (Samar et al., 2006), neural networks (Qi et al., 2007) and in some cases apply redundancy (Bateman et al., 2008). Practically in this work, the implementation of Petri nets is a pioneer in its application in the field of UAVs; there are no references which cite the work of Petri nets applied to the UAS.

## 3. Petri Nets

Petri Nets (PN) are a graphical and mathematical modeling tool applied to many systems. It is a tool with great projection in the field of automatic, which you can study and describe information-processing systems that are characterized as being concurrent, parallel, asynchronous, distributed, and not deterministic or stochastic. PN as graphical tool can be used as an aid of visual communication, similar to flow charts, block diagrams and networks. In addition, the marks are used in these nets to simulate the dynamics and activities of multiple systems. As a mathematical tool it is possible do state equations, algebraic equations and other models that govern the behavior of systems.
This section of the document is to provide basic concepts of PN that are required to cover the following topics. Below are the issues of Petri nets with their most important features, in addition, presents the concept of Hybrid Petri Nets, which is the basis for developing the diagnoser of the item later. To search for a better understanding of the subject of PN you can read (Silva 1985) (David & Alla, 1992) (Murata, 1989).

### 3.1 Petri Nets

A Petri Net (PN) has two types of nodes, called places and transitions. A place is represented by a circle and a transition by a bar. The places and transitions are connected by arcs. The number of places and transitions are finite and not zero. An arc is connected directly from one place to a transition or a transition to a place. In other words a PN is a bipartite graph, i.e. places and transitions alternate on a path made up of consecutive arcs.
**Definition 1.** A ordinary PN or a structure of PN is a bipartite graph represented by the 4-tuple $G = (P, T, I, O)$ such that:

$P = \{p_1, p_2, ..., p_n\}$ is a finite, not empty, set of places;
$T = \{t_1, t_2, ..., t_m\}$ is a finite, not empty, set of transitions;
$P \cap T = \phi$ , i. e. the sets $P$ and $T$ are disjointed;
$I : P \times T \to \{0,1\}$ is the input incidence function;
$O : T \times P \to \{0,1\}$ is the output incidence function;

$I(p_i, t_j)$ is the weight of the arc. $p_i \to t_j$ . This weight is 1 if the arc exists and 0 if not. $O(p_i, t_j)$ is the weight of the arc $t_j \to p_i$ . $I$ and $O$ thus relate to transition $t_j$ of the

pair $(p_i, t_j)$. The symbol $\circ t$ ($t \circ$) denotes the set of all points $p_i$ of entry/exit, $t_j$ such that $I(p_i, t_j) \neq 0$ ($O(p_i, t_j) \neq 0$). Similarly, $\circ p$ ($p \circ$) denote the set of all transitions $t_j$ input/output $p_i$ such that $O(p_i, t_j) \neq 0$ ($I(p_i, t_j) \neq 0$).

### 3.1.1 Marked PN

Each place contains an integer (positive or zero) marks. The number of tokens in one place $p_i$ is called $M(p_i)$. The marked net $M$ is defined by the marked vector of this marked, i.e. $M = (m_1, m_2, ..., m_n)$. The marking at a certain moment defines the state of the PN, or more precisely the state of the system described by the PN. The evolution of the state therefore corresponds to an evolution of the marking, caused by the firing of transitions. A transition can be fired only if each of the input places of this transition contains at least one token. The transition is then said to be fireable or enabled. The firing of a transition $t_j$ is to remove a token from each of the input places of transition $t_j$ and adding a token to each of the output places of transition $t_j$. When a transition is enabled, this does not imply that it will be immediately fired, this only remains a possibility. The firing of a transition is indivisible; it is useful to consider that the firing of a transition has duration of zero.

**Definition 2.** A marked Petri Net is a par $N = (G, M_0)$ in which $G$ is unmarked PN and $M_0$ is an initial marking. The matrix of pre-incidence $G$ is $C^- = `[c_{ij}^-]$ where $c_{ij}^- = I(p_i, t_j)$; the post-incidence matrix $G$ is $C^+ = `[c_{ij}^+]$ where $c_{ij}^+ = O(p_i, t_j)$, then the matrix of incidence $G$ is $C = C^+ - C^-$.

In a system of PN, a transition $t_j$ is enabled to the marking $M_k$ if $\forall p_i \in P, M_k(p_i) \geq I(p_i, t_j)$; an enabled transition $t_j$ can be fired reaching a new marking $M_{k+1}$ which can be computed as $M_{k+1} = M_k + C$, where $C$ is the incidence matrix of the PN, this equation is called state equation of PN. $R(G, M_0)$ is the set of all markings reachable from $M_0$ firing only enabled transitions.

Let $\sigma$ a firing sequence of transitions which can be performed from a marking $M_i$, which can be written as $M_i[\sigma\rangle$. The characteristic vector of sequence $\sigma$, written as $\vec{\sigma}$ is the m-component vector whose component number $j$ correspond to the number of firings of transition $t_j$ in the sequence $\sigma$. If the firing sequence $\sigma$ is such that $M_i[\sigma\rangle M_k$, then the state equation is obtained by

$$M_k = M_i + W.\vec{\sigma} \tag{1}$$

A sequence of transitions firing of a PN $(G, M_0)$ is a sequence of transition $\sigma = t_i, t_j, ...t_k ...$ such that $M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} M_x \xrightarrow{t_k} ...$. The set of all firing sequences is called the language:

$$L(G,M_0)=\left\{\sigma=t_i,t_j,...t_k...\wedge M_0\underset{\rightarrow}{t_i}M_1\underset{\rightarrow}{t_j}M_x\underset{\rightarrow}{t_k}...\right\} \quad (2)$$

### 3.2 Hybrid Petri Nets

The concepts of Hybrid Petri nets presented here are a synergy of the work carried out by (Silva 1985) (David & Alla, 1992). The places continuous of the PN represent the equation of the continuous dynamic of the process, or a real number that represents a number of tokens of place continuous. Therefore, for hybrid PN used in this chapter, symbolizes the continuous places and transitions with the letter (C) and discrete places and transitions with the letter (D).

As shown in Figure 1, the representation of places and transitions of the discrete and continuous is different; moreover, the marking of a continuous place is represented by an equation or a real number as opposed to a discreet place to stay tokens.

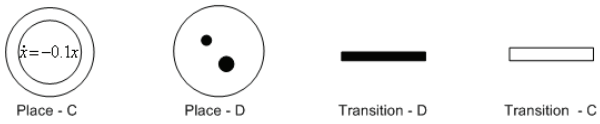



Fig. 1. Places and Transitions PN Hybrid

**Definition 3.** An Unmarked Hybrid PN is a pair $H^*=\langle Q,h\rangle$ fulfilling the following conditions:

- Q is an unmarked PN, $G=(P,T,I,O)$ where
  $P=\{p_1,p_2,...,p_n\}$ is a finite, not empty, set of places;
  $T=\{t_1,t_2,...,t_m\}$ is a finite, not empty, set of transitions;
  $P\cap T=\phi$ , i. e. the sets $P$ and $T$ are disjointed;
  $I:P\times T\rightarrow\{0,1\}$ is the input incidence function;
  $O:T\times P\rightarrow\{0,1\}$ is the output incidence function;
- $h:P\cup T\rightarrow\{D,C\}$, called hybrid function, indicates for every node if it is a discrete node or continuous one.
- $I$ and $O$ function must meet the following criterion: If $p_i$ and $t_j$ are a place and a transition such that $h(p_i)=D$ and $h(t_j)=C$ , then $I(p_i,t_j)=O(p_i,t_j)$ must be verified.

This last condition states that an arc must join a C transition to a D place as soon as a reciprocal arc exists. This ensures marking of D place to be an integer whatever evolution occurs.

**Definition 4.** A **Marked Hybrid PN** is a par $H=\langle H^*,M_0\rangle$ where $H^*$ is an Unmarked Hybrid PN and $M_0$ is the initial marking. The initial marking of a D place is a positive or null integer while the initial marking of a place-C is an equation or a real number.

**Definition 5.** A **Generalized Hybrid PN** is defined as a Marked Hybrid PN, except that:

- If $p_i$ is a D place, $I(p_i,t_j)$ and $O(p_i,t_j)$ are positive integers.
- If $p_i$ is a C place, $I(p_i,t_j)$ and $O(p_i,t_j)$ are positive real numbers.

An incidence matrix $C$ is associated with each network:

$$C = \left\lfloor C_{ij} \right\rfloor_{n \times m}, \text{ where } C_{ij} = O(p_i, t_j) - I(p_i, t_j) \quad (3)$$

**Definition 6.** A **D transition** is enabled if each place $p_i$ in $\circ t_j$ verifies the $M(p_i) \geq I(p_i, t_j)$. You can see that this definition does not separate the case where $p_i$ is a D place of a case where $p_i$ is a C place.

**Definition 7.** A **C transition** is enabled if the two following conditions are met:

- For each D place, $p_i$ in $\circ t_j$, $M(p_i) \geq I(p_i, t_j)$
- For each C-place, $p_i$ en $\circ t_j$, $M(p_i) > 0$

For a C transition, the kind of place preceding the transition must be specified because the enabling conditions are different according to whether it is a place between C place or D place.

Let $\sigma$ a sequence of firing and $\vec{\sigma}$ be characteristic vector of $\sigma$. The dimension of vector $\vec{\sigma}$ is equal to the number $m$ of transitions. The j-th component of $\vec{\sigma}$ represents the number of firings of transitions $t_j$ and will be denoted by $\vec{N}_j$. If $t_j$ is a D transition, then $\vec{N}_j$ is an integer and if $t_j$ is a C transition, then $\vec{N}_j$ is a real number.

A marked $M$ can be deduced from a marking $M_0$ due to a sequence $\sigma$, using the fundamental relation:

$$M = M_0 + C.\vec{\sigma} \quad (4)$$

The fundamental relation of a Hybrid PN is identical with the fundamental relation of a Discrete PN. We can so deduce that every property PN discrete resulting from this relation can be transposed to Hybrid PN.

## 4. Algorithm of Construction of Model and Diagnoser with *PN.*

In other investigations the model of the system is building with FSM's, presenting great difficulties in construction that grows as we increase the system's components, becoming the be unfeasible due to the problem of combinational explosion, which improves with the implementation of the model using Petri nets.

### 4.1 Building the Model

The model represents the real dynamics of the process, including the faults. The model of the DES's of the system is represented by PN Hybrid. The fundamental theory of the PN is based on identifying individual components of the system (DES's) and the relation between them; it must include the normal behavior of the process together with the failure behavior. $G = (P, T, I, O)$ be the PN that represents the discrete event model of the system to diagnose. Transitions $T$ are classified as unobservable $T_{UO}$ and observable $T_O$. Observable means that these transitions are given by the control events (command supervisor) or the instrumentation deployed in the process, not observable concerns to transitions that happen and the system can not normally detect. Within the unobservable transitions can include fault transitions $Tf$ , in other words, fault transitions is a subset of the unobservable

transitions $Tf \subseteq T_{UO}$, the objective set out by any system of FD is identify $Tf$, because the $T_O$ can be easily identified by the system.

The $Tf$ are classified into disjoint sets corresponding to different types of failure that may occur in the system, being important distribute failures in groups to facilitate their identification to diagnosis system, therefore, all fault transitions $Tf$ is composed of different subsets of faults given in the process, $Tf = Tf_1 \cup ... \cup Tf_m$. $\prod f$ is the faults distribution.

**Classification in Subsystems**. We must classify the system $H$ into subsystems depending on their performance $H = H_1 \cup H_2 \cup ... \cup H_n$, and although there is close relationship between them, this classification allows us to make better use of the FD algorithm.

**Petri Nets Model Building of the Components.** When the system is divided into subsystems, the first step is building the discrete event model of each of the components of the process, assuming that the system has $N$ individual components, be the expression:

$$Q_i = (P_i, T_i, I, O, M_0) \tag{5}$$

$i = 1,...., N$, $Q_i$ represents the PN of the i-component, it is important to note that should have a large knowledge of the process, since the model should include the normal and failure behavior of each component, and keep the synchrony of operation of the process whole.

**Integration Operation**. Refers to seek representation through a PN model the system behavior, which include different models of PN components, $\widetilde{Q} = (\widetilde{P}, \widetilde{T}, \widetilde{I}, \widetilde{O})$ is the denotation of the integrating operation of the PN models of $N$ components. This model integrates the normal and fault behavior of the system. From every place of the model transitions can occur normal function $T_O$ and failures transitions, that are $T_{UO}$, in every place of the PN will be give the integration of places of system components as follows:

$$\widetilde{P} = \bigcup_i P_i \qquad \text{and} \qquad \widetilde{T} = \bigcup_i T_i \tag{6}$$

$\widetilde{P}$ is composed of the union of the places of each individual $P_i$, and $\widetilde{T}$ by normal transitions $S$ ($T_O$), transitions are given by the supervisor or the process control system, and the transitions observable $T_{UO}$.

**Refined General Model**. It becomes necessary to consider only the observable part of $\widetilde{Q}$, therefore, $\widetilde{Q} = (\widetilde{P}, \widetilde{T}, \widetilde{I}, \widetilde{O})$ must be transformed to $Q = (P, T, I, O)$, it should rule out reaching transitions and unobservable transitions must be replaced by observable transitions. A place P is not achievable, when by the operating conditions of the system will never be present, this for say, marking the PN is not achievable.

$$\forall p_i : p_i \in P \rightarrow M(p_i) \notin R(Q, M_0) \tag{7}$$

$R(Q, M_0)$ is the set of all markings reachable system. The refinement is based on the construction of the integration table of $M$ sensors of the system. Given the set of $M$ sensors of the system of interest, we next identify the integrating sensors table

$h_j = \widetilde{P} \to \widetilde{Y}_j$, $j = 1,...,M$, where $Y_j$ denote the discrete set of outputs possible of the $j^{th}$ sensor, it define:

$$Y = \prod_{j=1}^{M} Y_j \tag{8}$$

And $h = \widetilde{P} \to Y$ denote the integrating sensors table, defined as follow.

$$h(p) = (h_1(p), h_2(p), ..., h_M(p)) \tag{9}$$

Finally, model is compound by normal and fault places, $P = P_N + P_F$. Transitions are compound by controller events $S$ and resulting event of the integrating sensors table $\sum$, $T = S \bigcup \sum$. Of this way, general model is compound of only observable transitions.

## 4.2 Diagnoser and Diagnosability

To build the diagnoser and to establish conditions necessary to diagnosability, system model should account with only observable transitions $T_O$ and observable places $P_O$, making the diagnoser simply and robust, we assume:

- There is a transition defined at each place $p \in P$, so the RdP will not reach anywhere sink place, avoiding that the net reach in a state of deadlock..
- It does not exist in $Q$ unobservable transitions $T_{UO}$

$tf$ be the final transition from a sequence $s$, define:

$$\ell(T_f) = \{st_f \in L : t_f \in Tf_i\} \tag{10}$$

$\ell(T_f)$ denote the set of all sequences of $L$ (languages representing system behavior), just in a transition belonging to the ruling class $Tf_i$, consider $t \in T$ and $s \in T^*$, we will use the notation to denote that $t$ is a transition of the sequence $s$, also writing $Tf \in T$ to any $tf \in Tf_i$.

**Diagnosability.** A system is diagnosable when identifying not only normal faults but also can define when a critical failure can occur, a critical or superior failure $fs$ is which belongs to the faults distribution of the system, such that, when the PN that represents the system reaches fault marking superior, the system enters a critical state or total failure.

$$\exists f_i \in \prod f \to M(p_{f_s}) \tag{11}$$

A PN is diagnosable in relation to the distribution of faults if it satisfies:

$$(\sigma \in \ell(Q, M_0), \wedge, \sigma \in T_o), (\forall i, k \in \prod f), \ \|\sigma\| < \infty : M(p_{f_i}) + M(p_{f_k}) \to M(p_{f_s}) \tag{12}$$

Where $\sigma$ is the sequence of observable transitions, therefore, a PN that represents the system is diagnosable if in a finite number of observable transitions, it reaching a fault marking $M(p_f)$ alone or joined with other fault marking $M(p_{f_k})$ can identify a superior or critical fault.

**Diagnoser.** The diagnoser is a PN implemented taking as a starting point the refined model of the system, conducting an on-line observation of the model, in order to perform a diagnostic on the system behavior. we will first have to define fault labels. $\Delta f = \{F_1, F_2, ..., Fm\}, \left|\prod f\right| = m$, the set of failure labels $\Delta$ is compound for normal labels $N$ and fault labels $F$, $\Delta = \{N\} \cup \{F\}$. Diagnoser for $Q$ is a PN of the form $G_d = (P_d, T_d, I, O, P_O, t_O, t_{end})$, the sets of places, transitions, input arcs and output arcs keep the same definitions of the PN, adding a starting place $P_O$, a starting transition $t_O$ and a end transition of supervision $t_{end}$. All will be operated by the supervisor of the system to diagnose. The starting place $p_O$ always start with the normal label, followed in this is the starting transition $t_O$ which do the task of start the PN diagnoser, also is adding the end transition $t_{end}$ for receiving the command from the operator to end the operation of the diagnoser.

The set of places $P_d$ of the diagnoser is a extension of the set of places of general model, a place $p$ of $G_d$ it is of the form $(p_i, l_i)$ where a place belong to observables places, $P_i \in P_O$ and the label belong to labels set, $l_i \in \Delta$, then places are of the form $l_i = \{N\} \vee \{F\}$, a place $P_d$ take the label of normal or fault operation.

An observer of $Q$ provides an estimate of current location of the system after the onset of each transition observed, the diagnosis $G_d$ can be understood conceptually as an extended observer, which is added to each estimate place a label instead of the kind mentioned above, the labels attached indicate the status of the component, if it is in fault mode or normal mode, faults are diagnosed validation labels.

We define functions essential for the construction of diagnosis:

**Label Assigned Function:** $LA : P_O \times \Delta \times T^* \to \Delta$, given $P \in P_O$, $l \in \Delta$ and $s \in L(Q, p)$, $LA$ assigns the label $l$ over $s$ starting from $p$ and following the dynamics of $Q$, according to:

$$LA(p, l, s)\begin{cases} \{N\} & si \quad \forall i[Tf_i \notin s] \\ \{F\} & si \quad \forall i[Tf_i \in s] \end{cases} \tag{13}$$

In the Q model was integrated the operation of the system, which are derived the faults in sink places, this makes PN model is blocked, to correct this problem, we leverages the capabilities concurrence of the PN and provides the fault expanding function of (FE).

**Fault Expanding Function**, $EF = R_N \times F_i \to R_F$ where $R_N$ is the normal operating branch and $R_F$ is the fault operating branch. For each set of failure $F_i$ of the distribution of failure $\prod f_i$ will create a new branch of failures in the PN to fulfill the role of overseeing the failures individually. The diagnosis $G_d$ will have as many branches as the system possesses faults, $R_G$ is the total number of branches of the diagnoser.

$$R_{G_d} = \sum_{i=1}^{M} R_{f_i} \tag{14}$$

PN diagnoser in each branch is evaluated possible changes in event unexpected or expected faults. Thanks to the function $LA$ , diagnoser evolves in normal or failure operation. The diagnoser evaluates each fault separately and takes into account in their transitions to the failures that are caused by other failures, while failures can be detected simultaneously and regardless of the order in which failures occur.
In summary, the algorithm must perform the following steps:

- Classification of the system into subsystems to diagnose
- Building of the PN model of each component subsystem, identifying the faults that may occur in each component.
- Construction of the PN general model, integrating the components of each subsystem.
- Building of the integration sensors table, combining state of the general model and combinations of the outputs of the sensors.
- Refinement of the general model based on the integration sensor table.
- Construction of the diagnoser. Once all the models of each subsystem PN are refined, the diagnoser is constructed, which integrates monitoring system.

## 5. Application: Unmanned Aerial Vehicles - UAVs

Several terms are frequently used in order to define aircrafts that are able to perform a mission without necessity to have a crew onboard. Thus, UAV (Unmanned Aerial vehicle), UAS (Unmanned Aerial system) or UAVs (Unmanned Vehicle Aerial System) are the most commonly used.
It should be understood that this condition does not preclude the existence of pilot, controller of the mission or other operators due to they can perform their work from the ground. The term UAVs reflects not only of the aircraft properly instrumented, but also a ground station, which complements the instrumentation and capabilities on board, see Figure 2.
Unmanned aircraft have been a field of interest for these past two decades particularly in the military, which started from testing equipments and currently to suitable professional application. There is an evident opportunity for growth in the application of UAV in non-military fields. Nowadays, a big number of companies have their R&D efforts focused on this area. Alongside the interest in military applications, extending their use to civilian missions led to the rise in the number of research groups and small businesses dedicated to developing of subsystems by integrating them or implementing applications and services based on unmanned aircraft.

Civilian applications for UAVs are available in various areas such as: border and coast patrol, obtaining data for mapping, fire fighting, monitoring of energy infrastructure, supporting law enforcement, search and rescue, maritime traffic control, monitoring of hazardous materials and crisis management, among others.

Fig. 2. Unmanned Aerial Vehicle

At present there is no regulation about the use of UAVs. Considering the increase in their application and operations, guidelines that define their use and classification have to be implemented in order to regulate their use. This action aims to avoid endangering persons, by defining flying areas and respecting the norms of aviation.
There is a source of information about reliability of the UAVs and it is in the military field (Office of the Secretary of Defense USA, 2003). Although there is currently some research on UAVSs in FD (Bateman et al., 2008)(Qi et al., 2007)(Drozexki aet al., 2005). This aims to make efforts in the FD of UAVs, which are complex systems and therefore vulnerable to failures without a posterior diagnosis.
According to data taken from The Office of the Secretary of Defense USA, 2003, reported failures in the UVS can be classified by deficiencies in: Power / Propulsion, Flight Control, Communication, Ground Control / Human Factors, Miscellaneous (Other), see Figure 3.
As shown in the figure 3, the highest number of failures given in UAVs is in the field of Power/Propulsion, followed by the flight control area. The FD algorithm presented in this chapter has been focused on this study in order to reduce the failure rate to the minimum.

### 5.1 Description of the UAV Used

A Vario Benzin Trainer model shown in figure 4 has been designed to test the FD algorithm, which has been used as a tool for a large number of applications in research on Automatic control at Cybernetics and Robotics group of the Universidad Politécnica de Madrid (Barrientos et al., 2009).
The helicopter is made up of three fundamental systems: the engine, the main rotor (plate) and the tail, see figure 5. If one of any these three systems fail, the mission has to be aborted immediately since the aircraft will definitely crash.

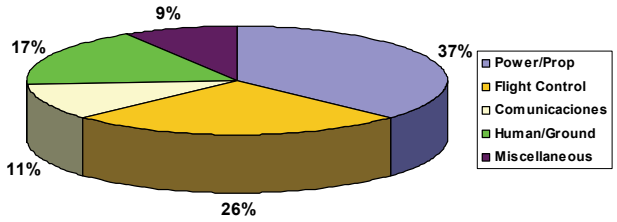



Fig. 3. Average sources of System Failures for U. S. Military UAV

Fig. 4. Vario Benzin Trainer Helicopter.

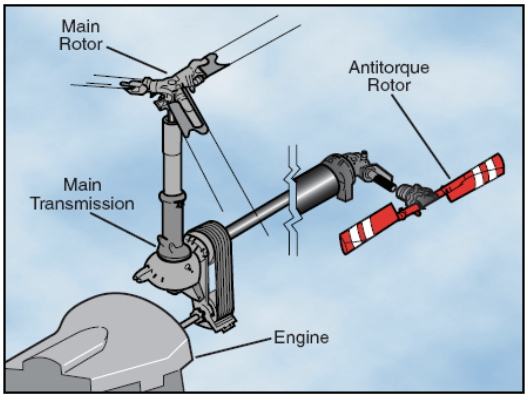



Fig. 5. Helicopter Components.

The motor is responsible for generating the movement of the rotors of the helicopter, see Figure 6. The combustion motor is powered by gasoline and fuel injection for the operation is done through a servo. This system has a controller that is responsible for maintaining the

rotor speed constant during the flight . It is then important to monitor the level of fuel in order to react in time. It is also vital to check that the servo is working properly.

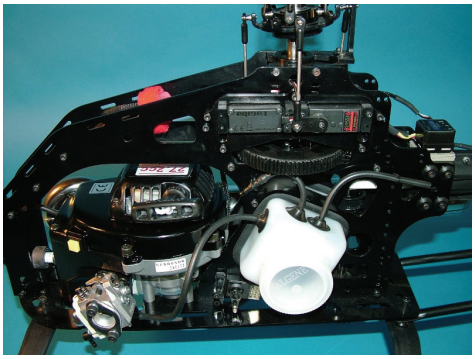


Fig. 6. Motor of the Helicopter Varior Benzin Trainer.

The main rotor system, see Figure 7, is controlled by four servos that are in charge of driving the blades so as to direct the helicopter according such as desired trajectory. The main rotor and its respective servos are connected to the motor through a mechanical transmission. Although there is a redundancy in the use of four servos for controlling the main swash plate (only three servos are required), in case of any failure in any of them, the pilot will probably lose the control. Therefore, it is important to monitor these servos.
The Tail Rotor is made up of two small blades and a servo that controls their tilt angle. The Yaw angle of the helicopter can be modified by changing this tilt angle in the tail rotor blades. If the tail rotor servo is damaged, the aircraft will lose the control.

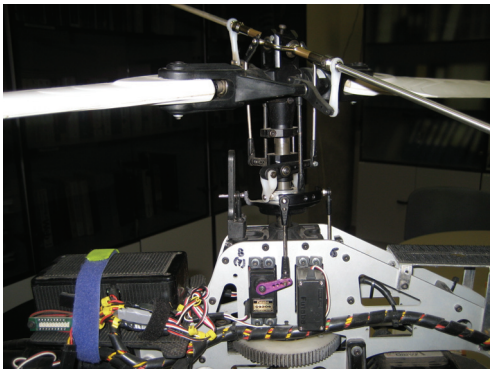


Fig. 7. Main Rotor System.

The Helicopter relies on additional devices that are also relevant in order to maintain flight plan, such as: The voltage of the Power Supply, sensors (IMU, gyroscopes, GPS, etc.), controllers, communications, ground control station and so on. The payload can also be considered as a relevant part of the aircraft.

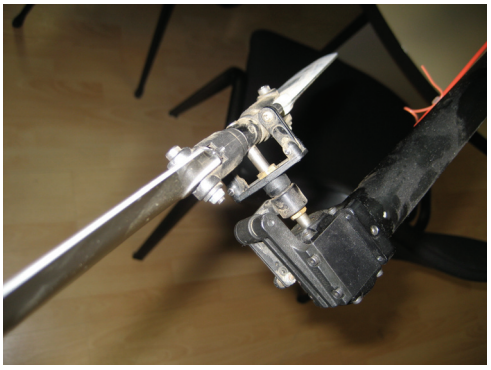


Fig. 8. Tail Rotor System.

### 5.2 Application of the Fault Diagnosis Algorithm.

After analyzing the importance of the three systems that make up the helicopter and finding a simple way to implement the FD tool, next step in based on the implementation of the algorithm to the helicopter.

Some assumptions must be done during the development of the FD algorithm:

- The helicopter has to be started manually.
- No failure on the controller happens.
- No failure on the power supply.

The algorithm starts with the implementation of the methodology in each subsystem individually and after that, all of them are integrated into the diagnoser.

#### 5.2.1 Classification of Subsystems in Helicopter.

The helicopter can be classified into three subsystems $H = H_1 \cup H_2 \cup H_3$, see Figure 9, the motor subsystem, main rotor subsystem and tail rotor subsystem.

#### 5.2.2 Construction of the PN Model for each of the Components of the Subsystem.

**The subsystem motor** is made up of controllers, servos, fuel storage tank and sensors.

The measure variables are: The level of fuel in the tank (L), the motor temperature (T) and the motor revolutions per minute (RPM). The faults to diagnose are: Fault Warming Motor (FWM), that is the maximum temperature allowed in the motor for the helicopter to fly. Lack of gasoline in the fuel tank (FLF). The level of fuel in the tank should not move below a minimum threshold. Stuck failure in Servo (FSS1). It could appear when the controller gives a command for opening or closing the passage of the fuel servo, and does not respond accordingly, i.e. the RPM falls below a minimum threshold, it may be due to a blockage of the servo. Faults can occur in any place of the devices.

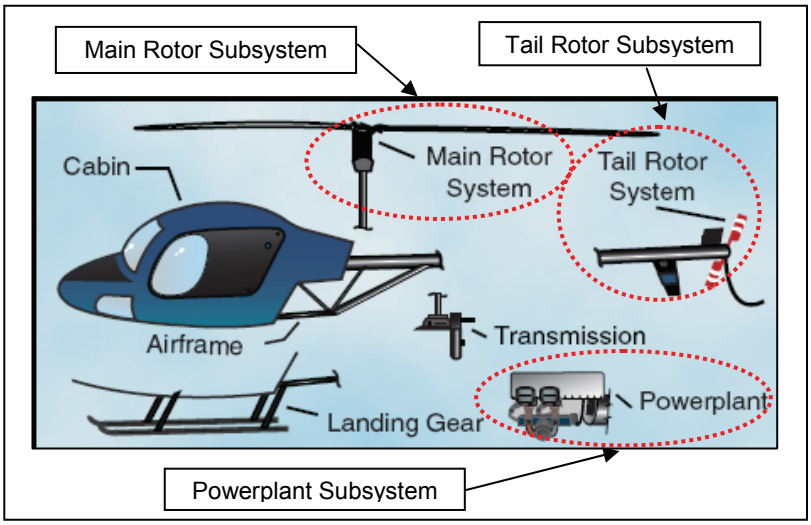



Fig. 9. Classification of Subsystems of the helicopter.

Figure 10 defines the PN model of the servo and controller. The integration of normal functioning and the three kind of failures listed above have been taken into account in each PN. The fault transitions are unobservable (Tuo) and are represented by bars and shaded circles. The PN model of the controller is an abstraction of its operation. Considering C1 as a idle state of the controller where it is waiting for a command of the pilot through the servo (AS1), when it happens, the controller changes to another state (C2). When the controller is located at C2 and receives a new command AS2 it returns to place C1. In the same way as the controller, the PN model of servo takes into account the normal and the failure behavior. Starting from the idle place SNA1, when an order of the controller is received (AS1), it has to move to required action place (SRA1).

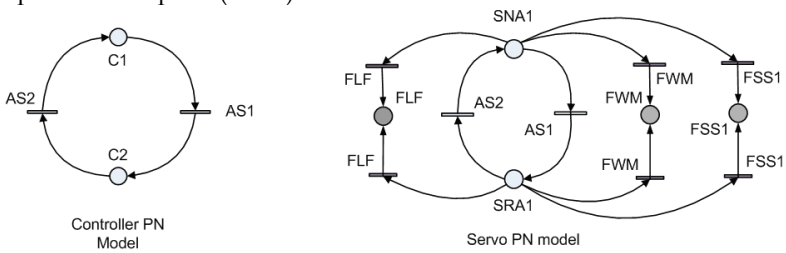



Fig. 10. Components PN Model of the Motor Subsystem

**The Main Rotor subsystem** is made of four servos and a controller (in autonomous systems) that sends information to them in order to control the attitude of the plate and therefore the attitude of the helicopter that is the way to control the velocity, see Figure 11.
The controller model moves from a state of an idle position (C3) to an expected position (C4). A single model of servo PN has been defined, which represents the four servos that control the plate. In addition to this, the model considers that when the servos are in a idle position (SNA2) and a change is required (AS3) the position (SRA2) is reached. The fault to

diagnose in this subsystem is the servo stuck fail (FSS2), which is perceived when the helicopter should go to an expected position and the sensors showed wrong reaction.

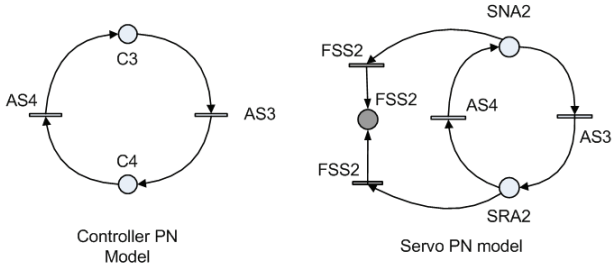



Fig. 11. Components PN Model of the Main Rotor Subsystems

**The Tail Rotor subsystem** consists of the servo which controls the pitch angle of the tail blades, the transmission system to the blades of the tail rotor and the controller, see Figure 12. The abstract model of the controller is defined by an idle state (C5). When it receives the turning forward to a new reference, a new pitch angle in the tail blades are required (AS5) and them it moves to C6. The tail rotor servo is defined by an idle state (SNA3) and the order of the controller (AS5, AS6) changes it to SRA3 state. The failure to diagnose in this subsystem is the fault of servo stuck (FSS3).

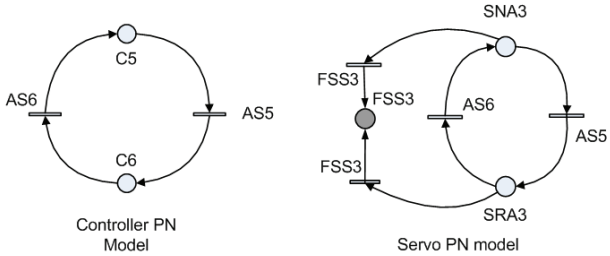



Fig. 12. Components PN Model of the Tail Rotor Subsystem.

### 5.2.2 Building of the General PN Model.

The general PN model integrates the models of each individual components, it allows seeing in a single PN model the normal and failure operation of each subsystem. In this new model the places and transitions failure remain as in individual models, but the union of the normal places has been performed. In the general PN model of the motor subsystem two new places have been considered (P1, P2). The normal places of the controller C1 and the current action of the servo SNA1 are synchronously integrated in P1. By other hand, P2 integrates the places of the controller C2 and required action of servo SRA2, see Figure 13.
In the general PN model of the main rotor subsystem, two new places are added (P3, P4). In P3 the normal operation of the controller C3 and current position of all servos (SNA2) are integrated, and P4 integrates the normal operation of the controller C4 and position of all servos expected (SRA2) , as figure 14 shows.

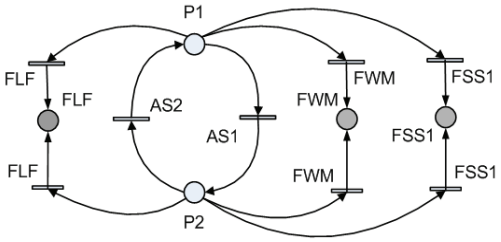



Fig. 13. General PN Model of Motor Subsystem.

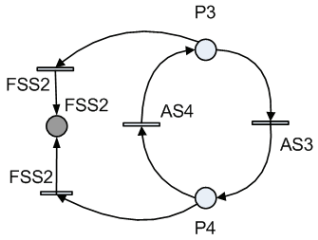



Fig. 14. General PN Model of the Main Rotor Subsystem

There are two new places in the general **PN** model of the **tail rotor subsystem**, (P5, P6). In P5 the normal operation of the controller C5 and the current position of the servo (SNA3) are integrated. P6 integrates the normal operation of the controller C6 and the servo required position (SRA3), as Figure 15 shows.

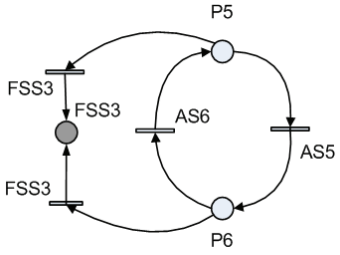



Fig. 15. General PN Model of the Tail Rotor Subsystem.

#### 5.2.3 Building of the Sensors Integration Table.

The subsequent step in the implementation of the FD algorithm, is to refine the PN general model, due to the fault transition (Tuo) have to be replaced with observables transitions (To). This process was made based on measures variables (sensors) with which system relies on. For this reason, the sensors integration table has to be define, it summarizes the possible outputs of the sensors. When this subsystem is in anywhere place of the normal operation, sensors can provide with different measures to those expected, indicating the presence of failure. These sensorial readings replace the failure transition and, in this way, the general PN model can be refined. The sensor integration table is developed for each subsystem.

The following concepts have been considered in the fault diagnosis system. Fuel Level tank (FLF), motor warming (FWM) and servo Fault (FSS1), as Table 1 shows. The measures from the sensors are defined as follows: Tank Level L=0, if the tank level is below the threshold and else L=1. This means that when the level L is equal to 0, the helicopter indicates a fault. The nominal temperature T of the motor must be under a threshold, thus when T=0 indicates that the motor temperature is in the normal range of operation, and T=1 means an overheated motor. The revolutions of the motor RPM are also evaluated by using a threshold. Thus RPM=0 means that motor revolutions is over this value (normal behavior), and if the RPM=1 means that the motor is not responding to controller orders and possibly there is a fault of servo or FLF, i.e. fuel injection failure. Table 1 shows all the possible combinations of the outputs from the sensors that define if the PN that represents the system falls into a fault or not. As general model defines, the subsystem of the motor has two places P1 and P2. In a normal operation of the motor, sensor readings should be at L=1, T=0 and RPM=0, therefore, if the system is in either P1 or P2 and the state of the sensor changes, a fault have been detected, indicating that the fault is no longer an unobservable transition, and it is moved a transition observable, defined by the corresponding sensors Outputs.
Applying the same concepts, the integrating sensor table for main rotor subsystem has been defined, see Table 2. We assess the fault of the servos FSS2, which is represented by the signals taken from the position sensors P and a time on the expected response t1. P=0 if the position given by the sensors is normal (no difference greater than 5% of the expected position), and P=1 if the difference exceeds this position. The time t1=0 if the response time of the expected position is less than 5 ms. and t1=1 if the response time is above the threshold of 5 ms. For places P3 and P4 readings measures should be P=0 and t1=0. If there is an unexpected change in the readings taken, the PN indicates a fault of stuck of any of the servos.

| *L* | *T* | *RPM* | *P*1(1,0,0) | | | *P*2(1,0,0) | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | *FLF* | *N* | *N* | *FLF* | *N* | *N* |
| 0 | 0 | 1 | *FLF* | *N* | *FSS*1 | *FLF* | *N* | *FSS*1 |
| 0 | 1 | 0 | *FLF* | *FWM* | *N* | *FLF* | *FWM* | *N* |
| 0 | 1 | 1 | *FLF* | *FWM* | *FSS*1 | *FLF* | *FWM* | *FSS*1 |
| 1 | 0 | 0 | *N* | *N* | *N* | *N* | *N* | *N* |
| 1 | 0 | 1 | *N* | *N* | *FSS*1 | *N* | *N* | *FSS*1 |
| 1 | 1 | 0 | *N* | *FWM* | *N* | *N* | *FWM* | *N* |
| 1 | 1 | 1 | *N* | *FWM* | *FSS*1 | *N* | *FWM* | *FSS*1 |

Table 1. Integrating Sensor Table of the Motor Subsystem.

The integrating sensors table for Tail Rotor Subsystem is shown in Table 3. The fault to diagnose is the servo stuck FSS3. It is evaluated by reading the yaw angle (Yaw) and the expected response time t2. The yaw angle y=0 if and angle of the expected movement of the helicopter is less than 5 degrees and y=1 if the yaw angle exceeds the threshold. Time t2=0 if the response time of the expected position is less than 5 ms. and t2=1 if the response time is above the threshold of 5 ms. In places P5 and P6 for the normal operation the readings must be y=0 and t2=0. When the PN reach from any place a variation of normal measures, the PN indicates a fault of tail rotor servo stuck.

| $P$ | $t1$ | $P3(,0,0)$ | $P4(,0,0)$ |
|---|---|---|---|
| 0 | 0 | $N$ | $N$ |
| 0 | 1 | $FSS2$ | $FSS2$ |
| 1 | 0 | $FSS2$ | $FSS2$ |
| 1 | 1 | $FSS2$ | $FSS2$ |

Table 2. Integration Sensors Table of the Main Rotor Subsystem.

| $y$ | $t2$ | $P5(,0,0)$ | $P5(,0,0)$ |
|---|---|---|---|
| 0 | 0 | $N$ | $N$ |
| 0 | 1 | $FSS3$ | $FSS3$ |
| 1 | 0 | $FSS3$ | $FSS3$ |
| 1 | 1 | $FSS3$ | $FSS3$ |

Table 3. Integration Sensor Table of the Subsystem tail rotor.

#### 5.2.4 Construction of the Refined PN Model.

The general PN model is composed of observable To and unobservable Tuo transitions, equivalent to faults transitions. These transitions unobservable have to be replaced by observable transitions; this is known as a refinement of the general model. After building the integration sensors table for each subsystem, is simply replace the transitions unobserved by the measure reading that indicates that PN falls within fault corresponding. For the motor subsystem, the transition of fault level fuel FLF have been replaced with reading the fuel level L. transition fault motor warm FWM is replaced by a reading of temperature T and the transition of fault servo stuck FSS1 have been replaced with reading the RPM and fuel level L, see Figure 16.

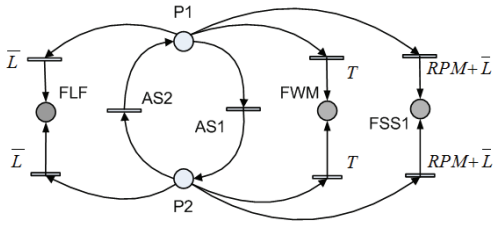



Fig. 16. Refined PN Model of the Motor Subsystem

The refined PN model of main rotor subsystem is shown in Figure 17. The transition fault servos stuck FSS2 has been replaced by the reading position P and the response time t1. The refined PN model of tail rotor subsystem is shown in Figure 18. The transition fault servo stuck of the tail rotor FSS3 has been replaced by the readings of the yaw angle and response time t2.

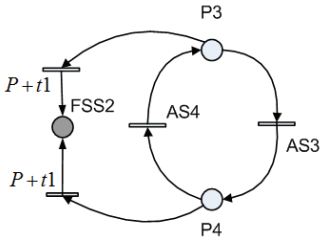



Fig. 17. Refined PN Model of the Main Rotor Subsystem

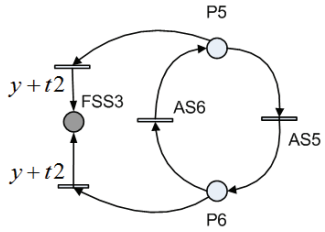



Fig. 18. Refined PN Model of the Tail Rotor Subsystem-

### 5.2.5 Building of the Diagnoser

The PN that represents the diagnoser is mainly composed of three branches, corresponding to each subsystem: motor branch, main and tail rotor branch, see Figure 19. the final goal of algorithm is integrated in one single PN the FD helicopter. The construction of each branch is based on the functions of Fault Expansion EF and Label Assigned AL. In PN diagnoser there are not sink places that can block the operation of the PN. The diagnoser makes an online assessment of whole system and serves as the supervisor, indicating where any of the branches fell into failure. If a branch falls on failure, the other branches continue assessing the system, although due to vulnerability of the helicopter must be taken to a place safe or landing for their repair.

Diagnoser has a normal place of start START and start transition START, the transition START is activated by the pilot of the ground control station to start the PN diagnoser and move a token for each of the branches of the helicopter subsystems. Likewise, the PN diagnoser has a transition end END, which allows the pilot to finish the diagnoser. The diagnoser is showed in the display of the ground control station. As in any system may occur intermittent fault, in the diagnoser have been added recovery transitions necessary to that if a fault is returned to its normal place can be observed by the pilot and take the necessary precautions.

The diagnoser is a tool that is added to display of the ground control station and is monitored by the pilot. Although the diagnoser has direct relation with the flight control, where it receives signals to assess faults, does not send any signal to the flight control, which could alter the functioning of the planned mission.

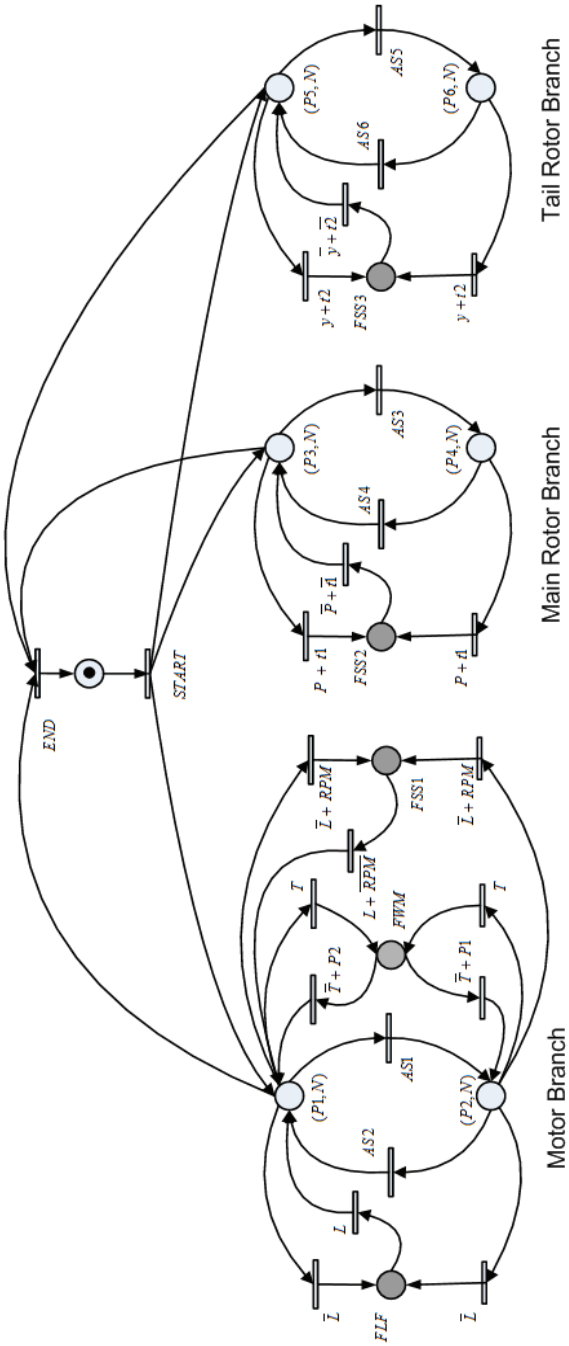



Fig. 19. Helicopter Diagnoser

## 6. Conclusion

This chapter has addressed the issue of fault diagnosis of hybrid systems using PN. An algorithm for construction of the model and a Diagnoser has been presented. The process has been classified into subsystems, which assesses the failure of independent way.
The proposed methodology turns out to be easy to implement and its construction incorporates devices that handle both continuous and discrete variables. They main advantages are the reduction of the combinational explosion, a systematic construction, the ability to be implemented in complex processes and the flexibility to make changes or add additional devices to be diagnosed.
As a demonstration, an application of a real hybrid system has been presented. The implementation on a radio control helicopter, which is a quite vulnerable and requires a robust fault diagnosis method.

## 7. References

Barrientos, A., Gutiérrez, P. & Colorado, J. (2009). Advanced UAV Trajectory Generation: Planning and Guidance, In: *Recent Advances in Signal Processing,* IN-TECH, pp. 56-82, ISBN 978-953-7619-41-1, Austria.

Bateman, F.; Noura, H. & Ouladsine, M. (2007). Actuators Fault Diagnosis and Tolerant Control for an Unmanned Aerial Vehicle*, 16th IEEE International Conference on Control Applications Part of IEEE Multi-conference on Systems and Control Singapore*, October 2007.

Bateman, F.; Noura, H. & Ouladsine, M. (2008). Active Fault Detection and Isolation Strategy for an Unmanned Aerial Vehicle with Redundant Flight Control Surfaces, *16th Mediterranean Conference on Control and Automation Congress Centre*, Ajaccio, France, June 2008.

Bonfe, M.; Castaldi, P.; Geri, W. & Simani, S.(2006). Fault detection and isolation for on-board sensors of a general aviation aircraft, *International Journal Of Adaptive Control And Signal Processing Int. J. Adapt. Control Signal Process*, May 2006.

Cassandras, C. (2002). From Discrete Event to Hybrid Systems, *Boston University*, IEEE, 2002.

Chung, S. & Jeng, M. (2003). Failure Diagnosis: A case Study on Modeling and Analysis by Petri Nets, IEEE, 2003.

Cork, L.; Walker, R. & Dunn, S. (2005). Fault Detection, Identification and Accommodation Techniques for Unmanned Airborne Vehicle, *Australian International Aerospace Congress*, Melbourne, march 2005.

David, R. & Alla, H. (1992). *Petri Nets & Grafcet: Tools for modelling discrete event*, Prentice Hall, Great Britain, 1992.

Drozeski, G., Saha, B. & Vachtsevanos, G. (2005). A Fault Detection and Reconfigurable Control Architecture for Unmanned Aerial Vehicles, *Aerospace Conference 2005* IEEE, March 2005.

Elgersma, M. & GlavaSki, S. (2001). Reconfigurable Control for Active Management of Aircraft System Failures, *Proceedings of the American Control Conference Arlington*, VA, June 2001.

Fourlas, G. Kyriakopoulos, K. & Krikelis, N. (2005). Fault Diagnosis of Hybrid Systems, *International Symposium on the Intelligent Control, IEEE*, Limassol Cyprus, 2005.

Genc, S. & Lafortune, S. (2006). Distributed Diagnosis of Places-boundered Petri Nets, *Department of Electrical Engineering and Computer Science*, University of Michigan, USA, 2006.

Giua, A. & Seatzu, C. (2005). Fault detection for discrete event systems using Petri Nets with unobservable transitions , *44th IEEE Conference on Decision and Control*, Seville, Spain, December 2005.

GlavaSki, S. & Elgersma, M. (2001). Active Aircraft Fault Detection and Isolation, *AUTOTESTCON Proceedings 2001 IEEE Systems Readiness Technology Conference*, 2001.

Hayhurst, K.; Maddalon, J. & Miner, P. (2006). Unmanned Aircraft Hazards And Their Implications For Regulation, *25th Digital Avionics Systems Conference*, NASA Langley Research Center, Hampton, Inc., Eastsound, WA, October 2006.

Henzinger, T. A. (1996) The Theory of Hybrid Systems, *Proccedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, pp.278-292, Lics, 1996.

Heredia, G.; Ollero, A.; Mahtani, R.; Béjar, M.; Remuß, V. & Musial, M. (2005). Detection of Sensor Faults in Autonomous Helicopters, *Proc. of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, Barcelona, Spain. April 2005.

Krogh, B. H. (2002). Recent Advances in Discrete Analysis and Control of Hybrid Systems, *Carnegie Mellon University*, Pittsburgh, USA, IEEE, 2002.

Mancini, A.; Caponetti, F.; Monteri`u, A.; Frontoni, E.; Zingaretti, P & Longhi, S. (2007) Safe flying for an UAV Helicopter, *Mediterranean conference on control and automation 2007*, Athens – Greece, July 2007

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications, *Proc. IEEE*, Apr. 1989, Vol. 77, no. 4, pp. 541-580.

Narasimhan, S.; Zhao, F.; Biswas, G. & Hung, E. (2000). Fault Isolation In Hibrid Systems Combining Model Based Diagnosis and Signal Proccesing, *Vanderbilt University*, IFAC 2000.

Office of the Secretary of Defense USA. (2003), Unmanned Aerial Vehicle Reliability Study, United States of America, February 2003.

Qi, J., Jiang, Z.; Zhao, X. & Han, J. (2007). Fault Detection Design for RUAV with an Adaptive Threshold Neural-Network Scheme, *2007 IEEE International Conference on Control and Automation*, Guangzhou, CHINA, May 2007.

Ramírez, A.; Ruíz, E.; Rivera, I. & López, E. (2007). Online Fault Diagnosis of Discrete Event Systems. A Petri Net Based Approach, *IEEE Trans. On Autom. Science and Engineering*, Vol. 4, no. 1, January 2007.

Samar, S.; Gorinevsky, D. & Boyd, S. (2006). Embedded Estimation of Fault Parameters in an Unmanned Aerial Vehicle, *Proceedings of the 2006 IEEE International Conference on Control Applications Munich*, Germany,, October 4-6, 2006.

Sampath, M., Sengupta, R.; Lafortune, S.; Sinnamohidee, K. & D. Teneketzis, (1995). Diagnosability of Discrete Event Systems, *IEEE Trans Autom. Contr*, Vol. 40, no 9, pp. 1555-1575, 1995.

Silva, M. (1985). *Las Redes de Petri: en la Automática y la Informática*, Editorial AC, Madrid - España, 1985.

Trigos, M. & García, E. (2008-A). Diagnóstico De Fallos De Sistemas De Eventos Discretos Basado En Redes De Petri, *5 Conferencia Internacional de Ingenierías Eléctrica FIE-08*, Julio 2008, Santiago de Cuba – Cuba.

Trigos, M. & Garcia, E. (2008-B). Faults Diagnosis and Modelling of the Liquid Packaging Process. A Research Based on Petri Nets, *Proceeding from the 10th International Conference of Robotics & Automation IEEE*, December 2008, Hanoi – Vietnam.

Ushio, T.; Onishi, I. & Okuda, K. (1998). Fault Detection Based on Petri Net models with Faulty Behaviors , IEEE, 1998

Venkatasubramanian, V.; Raghunathan, R.; Kemen, Y. and Surya, K. (2003). A review of process fault detection and diagnosis: Quantitative, Qualitative and History Process methods, Computer and Chemical Engineering, no. 27, pp. 293-346.

Zhang, X.; Liu, Y.; Rysdyk, R.; Kwan, C. & Xu, R. (2006). An Intelligent Hierarchical Approach to Actuator Fault Diagnosis and Accommodation, *Aerospace Conference,* 2006 IEEE.

Zhao, F.; Koutsoukos, V.; Haussecker, H.; Reich, J. & Cheung, P. (2005). Monitoring and Fault Diagnosis of Hybrid Systems, *IEEE Trans. Actions on Systems, Man, and Cybernetics- Part B: Cybertnetics*, Vol. 35, no. 6, December 2005.

19

# Design and Implementation of Hierarchical and Distributed Control for Robotic Manufacturing Systems using Petri Nets

Gen'ichi Yasuda
*Nagasaki Institute of Applied Science*
*Japan*

## 1. Introduction

To realize control systems for flexible manufacturing systems, it is necessary to provide effective tools for describing process specifications and developing control algorithm in a clear and consistent manner. Conventionally, process modeling at the conceptual level is based on flowcharts, time diagrams, state machine diagrams, etc., and the control algorithms at the execution level are generally developed with relay ladder diagrams or procedural languages for programmable logic controllers. However, in the area of real-time control of discrete event manufacturing systems the main problems that the system designer has to deal with are concurrency, synchronization, and resource sharing problems. For this class of problems, Petri nets have intrinsic favorable qualities and it is very easy to model sequences, choices between alternatives, rendezvous and concurrent activities by means of Petri nets. The network model can describe the execution order of sequential and parallel tasks directly without ambiguity. Moreover, the formalism allowing a validation of the main properties of the Petri net control structure (liveness, boundedness, etc.) guarantee that the control system will not fall immediately in a deadlocked situation. In the field of flexible manufacturing systems, the last aspect is essential because the sequences of control are complex and change very often. Furthermore, a real-time implementation of the Petri net specification by software called a token player can avoid implementation errors, because the specification is directly executed by the token player and the implementation of these control sequences preserves the properties of the model. In this approach, the Petri net model is stored in a database and the token player updates the state of the database according to the operation rules of the model. For control purposes, this solution is very well suited to the need of flexibility. When the control sequences change only the database must be changed.

Motivated by the potential that Petri nets have for modeling parallel and concurrent processes, some techniques derived from Petri nets have been successfully introduced as an effective tool for describing control specifications and realizing the control in a uniform manner (Komoda, et al. 1984), (Murata, et al. 1986). However, in the field of flexible manufacturing systems, the network model becomes complicated and it lacks for the readability and comprehensibility. Therefore, the flexibility and expandability are not

satisfactory in order to deal with the specification change of the manufacturing system. Depending on the size and complexity of the system, these models can become very difficult to understand and treat. Despite the advantages offered by Petri nets, the synthesis, correction, updating, etc. of the system model and programming of the controllers are not simple tasks. Furthermore, the increasing use of robots and other automated machines has generated control software written in different levels and it is impossible for one person to understand all the control specifications in a large and complex manufacturing system.

The overall structure of the working area in a large and complex manufacturing system consists of one or more lines, each line consists of one or more stations, and each station (shop or cell) consists of one or more machines such as robots and intelligent machine tools. Inside of a cell, machines execute cooperation tasks such as machining, assembling and storing. Inside of a shop, cells cooperate mutually and execute more complicated tasks. Furthermore each machine consists of several motion elements. A task executed by a robot or an intelligent machine tool can be seen as some connection of more detailed subtasks. For example, transferring an object from a start position to a goal position is a sequence of the following subtasks; moving the hand to the start position, grasping the object, moving to the goal position, and putting it on the specified place. Thus the manufacturing system handles complicated tasks by dividing a task hierarchically in this structure, which is expected to be effective in managing cooperation tasks executed by great many machines or robots (Hatvany, 1985). Since in the large and complex systems, the controllers are geographically distributed according to their physical (hardware) structure, it is desirable to realize the hierarchical and distributed control. Distributed implementation can bring about the simplicity, easy modification, and relatively efficient execution of the Petri net based control scheme because the size of each Petri net model is not so large. Conventional Petri net based control systems were implemented based on an overall system model. The hierarchical and distributed control for large and complex manufacturing systems has not been implemented so far. If it can be realized by Petri nets, the modeling, simulation and control of large and complex discrete event manufacturing systems can be consistently realized by Petri nets (Lee, 2006), (Queiroz, 2000).

In a manufacturing system such as workstation, robots often must interact with automated machines, other robots or operators (Hoermann, 1989), (Merabet, 1986), (Jones, et al. 1989). These external processes are executing in parallel and asynchronously. It is not possible to predict exactly when events of interest to the robot program may occur. The signal lines are supported by most robot systems to coordinate multiple robots and machines, but this is a very limited form of communication between processes. Sophisticated tasks require efficient means for coordination and for sharing the state of the system between processes. The programming system should provide a mechanism for specifying the behavior of systems more complex than a single robot. Existing robot programming systems are based on the view of a robot system as a single robot weakly linked to other machines. Many machines may be cooperating during a task. The interactions between them may be highly dynamic. No existing robot programming system adequately deals with all of these interactions. No existing computer language is adequate to deal with this kind of parallelism and real-time constraints (Silva, 1990).

In this chapter, the author presents a methodology by extended Petri nets for hierarchical and distributed control of large and complex robotic manufacturing systems, to construct the control system where the cooperation of each controller is implemented within a

coordinator mechanism so that the behavior of the overall system is not deteriorated and the task specification is completely satisfied. Based on the hierarchical and distributed structure of the system, the Petri net based specification procedure is a top-down approach from the conceptual level to the detailed level of the discrete event manufacturing systems. The macro representation of the system is broken down to generate the detailed Petri nets at the machine control level. Then the Petri nets are decomposed and assigned to the machine controllers. The proposed procedure is demonstrated through an example of robotic manufacturing cell.

## 2. Modeling of Robotic Manufacturing Systems using Petri Nets

A manufacturing process is characterized by the flow of workpieces or parts, which pass in ordered form through subsystems and receive appropriate operations. From the viewpoint of discrete event process control, an overall manufacturing process can be decomposed into a set of distinct activities (or events) and conditions mutually interrelated in a complex form. An activity is a single operation of a manufacturing process executed by a subsystem. A condition is a state in the process such as machine operation mode. To represent discrete event manufacturing systems a modeling technique was derived from Petri nets. Considering not only the modeling of the systems but also the actual manufacturing system control, the guarantee of safeness and the additional capability of input/output signals from/to the machines are required (Masuda, et al. 1981). From the viewpoint of real-time control, controlling a process consists in driving its devices from a state to another one. In discrete event system control domain, three steps are necessary for each evolution of the process: first, the control system sends a request to the process actuators, second, the process evolves according to the request and, at the end of the evolution, returns an execution report to the control system, third, the control system updates the states of the control models according to the report; then it is ready to send another request. This outline points out that, each time the control system sends a request to the process, it must wait for the associated report. Thus the function of the description method based on common Petri net technique is enhanced it may not move the tokens until the operation, shown by each place, is finished, even if the marking satisfies the ignition condition.
The extended Petri net consists of the following six elements: (1)Place (2)Transition (3)Directed arc (4)Token (5)Gate arc (6)Output signal arc. A transition is enabled if and only if it satisfies all the following conditions:

(1) It does not have any output place filled with a token.
(2) It does not have any empty input place.
(3) It does not have any internal permissive arc signaling 0.
(4) It does not have any internal inhibitive arc signaling 1.

An enabled transition may fire when it does not have any external permissive arc signaling 0 nor any external inhibitive arc signaling 1. The firing of a transition removes tokens from all its input places and put a token in each output place connected to it. The assignment of tokens into the places of a Petri net is called marking and it represents the system state. In any initial marking, there must not exist more than one token in a place. According to these rules, the number of tokens in a place never exceeds one, thus, the Petri net is essentially a safe graph.

If a place has two or more input transitions or output transitions, these transitions may be in conflict for firing. When two or more transitions are firable only one transition should fire using some arbitration rule. By the representation of the activity contents and control strategies in detail, features of discrete event manufacturing systems such as ordering, parallelism, asynchronism, concurrency and conflict can be concretely described through the extended Petri net.

## 3. Design of Hierarchical and Distributed Control

The overall procedure for the design and implementation of hierarchical and distributed control is summarized as shown in Fig. 1. A global, conceptual Petri net model is first chosen which describes the aggregate manufacturing process. At the conceptual level each task specification is represented as a place of the Petri net, where the activity of each equipment is also represented as a place. Based on the hierarchical approach, the Petri net is translated into detailed subnets by stepwise refinements from the highest system control level to the lowest machine control level (Suzuki, 1983). At each step of detailed specification, some parts of the Petri net, places, are substituted by a subnet in a manner, which maintains the structural properties. Then, the detailed Petri net is decomposed into subnets, which are executed by each machine controller.

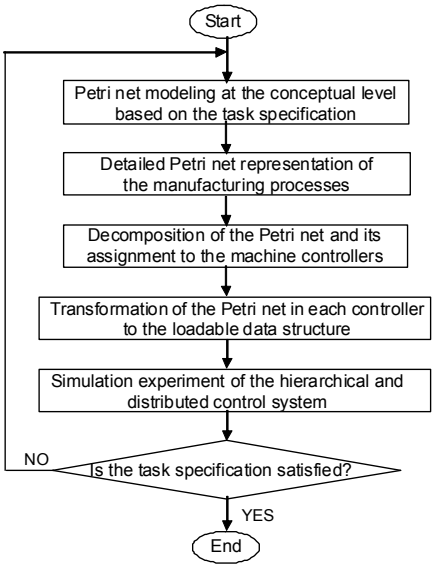



Fig. 1. Flow chart of Petri net based implementation of hierarchical and distributed control

In the decomposition procedure, a transition may be divided and distributed into different machine controllers as shown in Fig.2. The machine controllers should be coordinated so that these transitions fire in union.

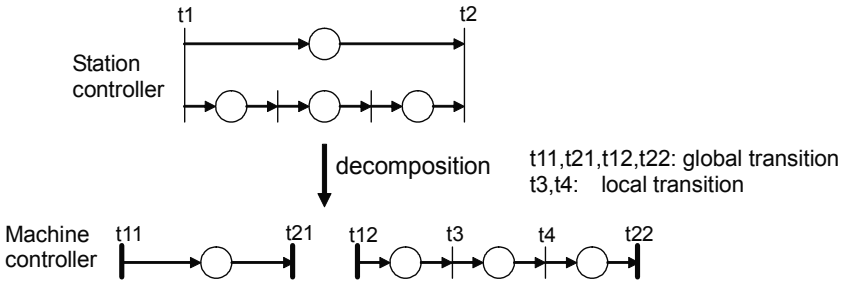



Fig. 2. Decomposition of transition

Decomposed transitions must function in union, that is, the aggregate behavior of decomposed subnets should be the same as that of the original Petri net. Decomposed transitions are called global transitions, and other transitions are called local transitions. By the Petri net model, the state of the discrete event system is represented as the marking of tokens, and firing of any transition brings about change to the next state. So the firing condition and marking before decomposition should be the same as those after decomposition. The firability condition and external gate condition of a transition before decomposition are described as follows:

$$t_j(k) = \bigcap_{m=1}^{M} p_{j,m}^{I}(k) \wedge \bigcap_{n=1}^{N} \overline{p_{j,n}^{O}(k)} \wedge \bigcap_{q=1}^{Q} g_{j,q}^{IP}(k) \wedge \bigcap_{r=1}^{R} \overline{g_{j,r}^{II}(k)} \tag{1}$$

$$g_j^{E}(k) = \bigcap_{u=1}^{U} g_{j,u}^{EP}(k) \wedge \bigcap_{v=1}^{V} \overline{g_{j,v}^{EI}(k)} \tag{2}$$

where,

$M$ : input place set of transition $j$

$p_{j,m}^{I}(k)$ : state of input place $m$ of transition $j$ at time sequence $k$

$N$ : output place set of transition $j$

$p_{j,n}^{O}(k)$ : state of output place $n$ of transition $j$ at time sequence $k$

$Q$ : internal permissive gate signal set of transition $j$

$g_{j,q}^{IP}(k)$ : internal permissive gate signal variable $q$ of transition $j$ at time sequence $k$

$R$ : internal inhibitive gate signal set of transition $j$

$g_{j,r}^{II}(k)$ : internal inhibitive gate signal variable $r$ of transition $j$ at time sequence $k$

$U$ : external permissive gate signal set of transition $j$

$g^{EP}_{j,u}(k)$ : external permissive gate signal variable $u$ of transition $j$ at time sequence $k$

$V$ : external inhibitive gate signal set of transition $j$

$g^{EI}_{j,v}(k)$ : external inhibitive gate signal variable $v$ of transition $j$ at time sequence $k$

The addition or removal of a token of a place is described as follows:

$$p^{I}_{j,m}(k+1) = p^{I}_{j,m}(k) \wedge \overline{(t_j(k) \wedge g^{E}_{j}(k))} \tag{3}$$

$$p^{O}_{j,n}(k+1) = p^{O}_{j,n}(k) \vee (t_j(k) \wedge g^{E}_{j}(k)) \tag{4}$$

The firability condition of a transition after decomposition is described as follows:

$$t_{jsub}(k) = \bigcap_{m=1}^{Msub} p^{I}_{jsub,m}(k) \wedge \overline{\bigcap_{n=1}^{Nsub} p^{O}_{jsub,n}(k)} \wedge \bigcap_{q=1}^{Qsub} g^{IP}_{jsub,q}(k) \wedge \overline{\bigcap_{r=1}^{Rsub} g^{II}_{jsub,r}(k)} \tag{5}$$

$$g^{E}_{jsub}(k) = \bigcap_{u=1}^{Usub} g^{EP}_{jsub,u}(k) \wedge \overline{\bigcap_{v=1}^{Vsub} g^{EI}_{jsub,v}(k)} \tag{6}$$

From eq.(1) and eq.(5),

$$t_j(k) = \bigcap_{sub=1}^{S} t_{jsub}(k) \tag{7}$$

From eq.(2) and eq.(6),

$$g^{E}_{j}(k) = \bigcap_{sub=1}^{S} g^{E}_{jsub}(k) \tag{8}$$

where,

$S$ : total number of subnets

$M_{sub}$ : input place set of transition *jsub* of subnet *sub*

$p^{I}_{jsub,m}(k)$ : state of input place $m$ of transition *jsub* of subnet *sub* at time sequence $k$

$N_{sub}$ : output place set of transition *jsub* of subnet *sub*

$p^{O}_{jsub,n}(k)$ : state of output place $n$ of transition *jsub* of subnet *sub* at time sequence $k$

The addition or removal of a token of a place connected to a decomposed transition is described as follows:

$$p^{I}_{jsub,m}(k+1) = p^{I}_{jsub,m}(k) \wedge \overline{(t_j(k) \wedge g^{E}_{j}(k))} \tag{9}$$

$$p^{O}_{jsub,n}(k+1) = p^{O}_{jsub,n}(k) \vee (t_j(k) \wedge g^{E}_{j}(k)) \tag{10}$$

Consequently it is proved that the firability condition of the original transition is equal to AND operation of firability conditions of decomposed transitions. If and only if all of the decomposed transitions are firable, then the global transitions are firable. The coordinator program has been introduced to coordinate the decomposed subnets so that the aggregate behavior of decomposed subnets is the same as that of the original Petri net.
In case that a transition in conflict with other transitions is decomposed, these transitions should be coordinated by the station controller (Fig. 3).

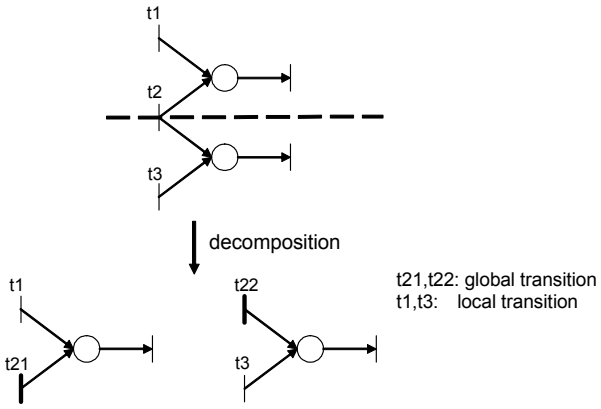



Fig. 3. Decomposition of transition in conflict

The Petri net based control structure with introduction of coordinator is shown in Fig. 4. The control software is distributed into the station controller and machine controllers. The station controller is composed of the Petri net based controller and the coordinator. The conceptual Petri net model is allocated to the Petri net based controller for management of the overall system. The detailed Petri net models are allocated to the Petri net based controllers in the machine controllers. Each machine controller directly monitors and controls the sensors and actuators of its machine.
The control of the overall system is achieved by coordinating these Petri net based controllers. System coordination is performed through communication between the coordinator in the station controller and the Petri net based controllers in the machine controllers as the following steps.

(1) When each machine controller receives the start signal from the coordinator, it tests the firability of all transitions in its own Petri net, and sends the information on the global transitions and the end signal to the coordinator.
(2) The coordinator tests the firability of the global transitions, arbitrates conflicts among global and local transitions, and sends the names of firing global transitions and the end signal to the machine controllers.
(3) Each machine controller arbitrates conflicts among local transitions using the information from the coordinator, generates a new marking, and sends the end signal to the coordinator.
(4) When the coordinator receives the end signal from all the machine controllers, it sends the output command to the machine controllers.
(5) Each machine controller outputs the control signals to its actuators.

Multilevel hierarchical and distributed control for large and complex manufacturing systems can be constructed such that the control system structure corresponds to the hierarchical and distributed structure of the general manufacturing system. The overall system is consistently controlled, such that a coordinator in a layer coordinates one-level lower Petri net based controllers and is coordinated by the one-level upper coordinator.

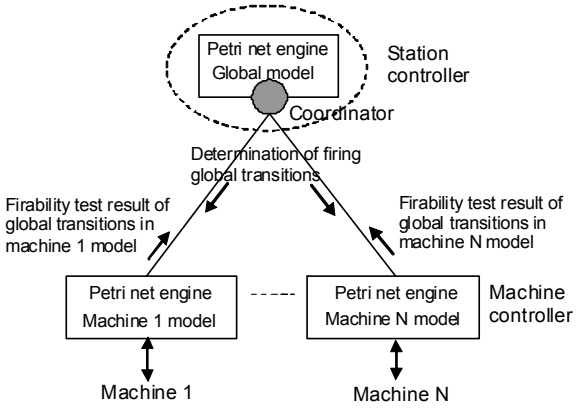



Fig. 4. Petri net based control structure with coordinator

## 4. Implementation of Example Control System

The basic procedures of modeling and decomposition of robotic manufacturing systems are shown through a simple example. The manufacturing system has two robots, a machining center, and two conveyors, where one is for carrying in and the other is for carrying out. The example manufacturing system is shown in Fig. 5. The main execution of the system is indicated as the following task specifications:

(1) A workpiece is carried in by the conveyor CV1.
(2) The robot R1 loads the workpiece to the machining center MC.
(3) The machining center MC processes the workpiece.

(4) The robot R2 unloads the workpiece from the machining center and places it on the conveyor CV2.
(5) The workpiece is carried out by the conveyor CV2.

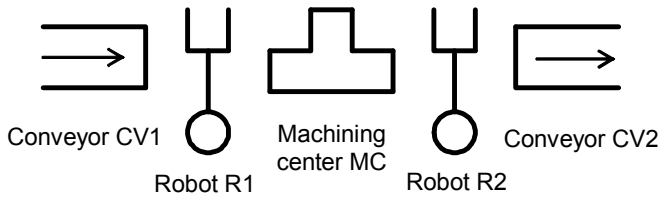



Fig. 5. Example of robotic manufacturing system

A global, conceptual Petri net model is first chosen which describes the aggregate manufacturing process. The places which represent the subtasks indicated as task specifications are connected by arcs via transitions in the specified order corresponding to the flow of subtasks and a workpiece. The places representing the existence of machines are also added to connect transitions which correspond to the beginning and ending of their subtasks. Thus at the conceptual level the manufacturing process is represented as shown in Fig. 6. In this step, if necessary, control conditions such as the capacity of the system between the respective subtasks must be connected to regulate the execution of the Petri net. Next, each place representing a subtask at the conceptual level is translated into a detailed subnet. Fig. 7 shows the detailed Petri net representation of loading, processing and unloading in Fig. 6.
For the manufacturing system, various control structures can be considered as shown in Figs. 8 and 9. In the centralized control system, the station controller directly controls all the machines using detailed Petri net (Fig. 8). For an example structure of hierarchical and distributed control composed of one station controller and three machine controllers (Fig. 9(b)), the Petri net executed in each machine controller is shown in Fig. 10.

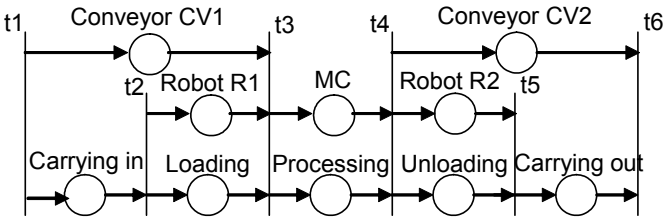
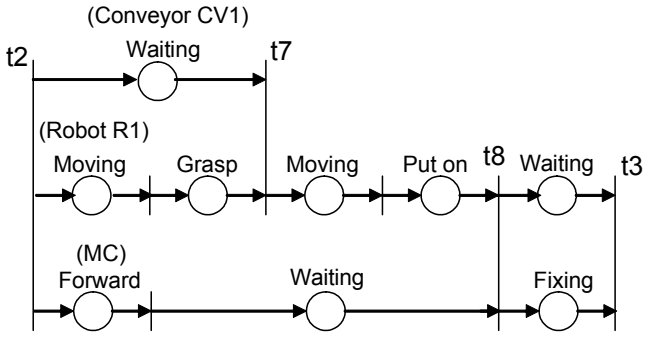
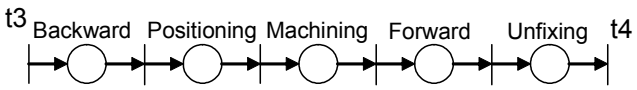


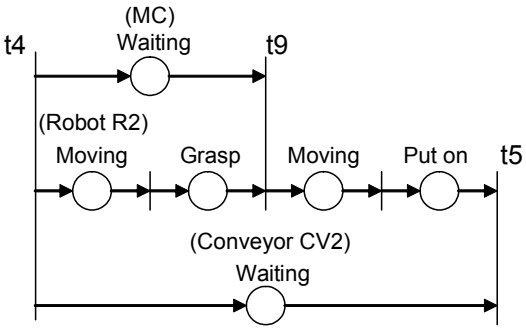

Fig. 6. Petri net representation of the example system at the conceptual level




(a) Loading




(b) Processing




(c) Unloading

Fig. 7. Detailed Petri net representation of subtask operations

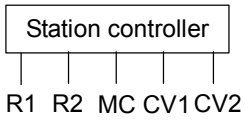
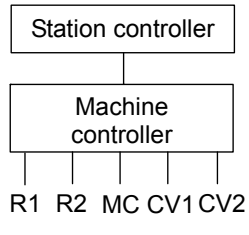
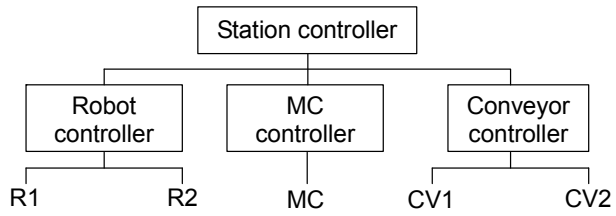


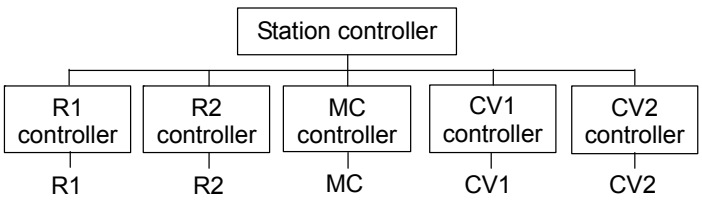

Fig. 8. Example structure of centralized control system




(a)

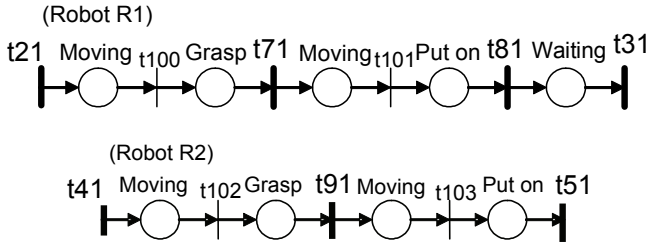



(b)

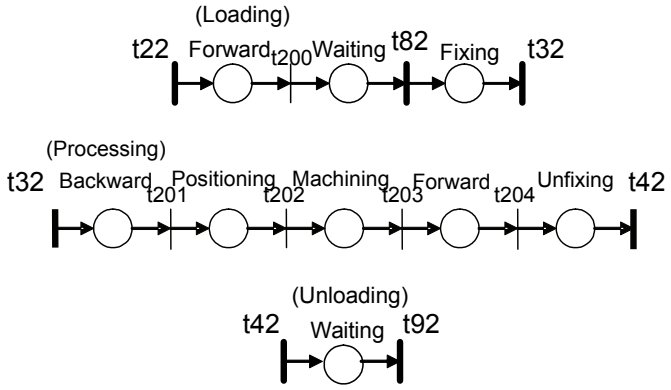



(c)

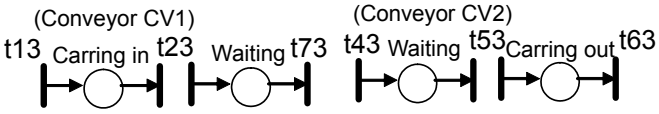
Fig. 9. Example structures of hierarchical and distributed control system




(a) Robot controller




(b) MC controller




(c) Conveyor controller

Fig. 10. Petri net representation of machine controllers
(▌: global transition, ▏: local transition)

For the example system, the hierarchical and distributed control system has been realized using a set of PCs. Each machine controller is implemented on a dedicated PC. The robot controller executes robot motion control through the transmission of command. The station controller is implemented on another PC. Communications among the controllers are performed using serial communication interfaces. The machine controllers control two conveyors or robots, so control software on each PC is written using multithreaded programming.
The names of global transitions and their conflict relations are loaded into the coordinator in the station controller. The connection structure of a decomposed Petri net model and conflict relations among local transitions are loaded into the Petri net based controller in a machine controller. In the connection structure, a transition of a Petri net model is defined using the names of its input places and output places; for example, t1-1=b1-1, -b1-11, where the transition no.1 (t1-1) of Robot controller (subsystem no.1) is connected to the input place no.1 and the output place no.11. For the distributed control system shown in Fig. 9(b), information inputted to the loader is as follows.

(Robot controller)
t1-21=-b1-3 t1-100=b1-3,-b1-4 t1-71=b1-4, -b1-5 t1-101=b1-5, -b1-6
t1-81=b1-6, -b1-7 t1-31=b1-7 t1-41=-b1-13 t1-102=b1-13,-b1-14
t1-91=b1-14, -b1-15 t1-103=b1-15, -b1-16 t1-51=b1-16

(MC controller)
t2-22=-b2-8 t2-200=b2-8, -b2-9 t2-82=b2-9, -b2-10 t2-32=b2-10, -b2-12
t2-42=b2-11, -b2-17 t2-92=b2-17

(Conveyor controller)
t3-13=-b3-1 t3-23=b3-1, -b3-2 t3-73=b3-2 t3-43=-b3-12
t3-53=b3-12, -b3-19 t3-63=b3-19

Using the names of transitions in the subsystems, global transitions are defined; for example, G2: t0-2, t1-21, t2-22, t3-23 indicates that the global transition G2 is composed of the transition no.2 of Station controller (subsystem no.0) , the transition no.21 of Robot controller, the transition no.22 of MC controller (subsystem no.2), and the transition no.23 of Conveyor controller (subsystem no.3). Then, the coordinator information for the example distributed control system is as follows.

G1: t0-1, t3-13 G2: t0-2, t1-21, t2-22, t3-23 G3: t1-71, t3-73
G4: t1-81, t2-82 G5: t0-3, t1-31, t2-32 G6: t0-4, t1-41, t2-42, t3-43
G7: t1-91, t2-92 G8: t0-5, t1-51, t3-53 G9: t0-6, t3-63

By executing the coordinator and Petri net based controllers algorithms based on loaded information, simulation experiments have been performed. Experimental results show that the decomposed transitions fire at the same time as the original transition of the detailed Petri net of the whole system task. Firing transitions and marking of tokens can be directly observed on the display at each time sequence using the Petri net simulator.

## 5. Conclusions

A methodology to construct hierarchical and distributed control systems, which correspond to the hardware structure of manufacturing systems, has been presented. The overall control structure composed of one station controller and several machine controllers has been implemented using a communication network of PCs for the example robotic manufacturing system. The conceptual Petri net model of task specification is allocated to the Petri net based controller in the station controller for management of the overall system. The detailed Petri net models are allocated to the Petri net based controllers in the machine controllers. By introduction of the coordinator, the Petri net based controllers are arranged according to the hierarchical and distributed nature of the manufacturing system. Experimental results show that the decomposed transitions fire at the same time as the original transition of the detailed Petri net of the whole system task. The Petri net model in each Petri net based machine controller is not so large and easily manageable. Thus, modeling, simulation and control of large and complex manufacturing systems can be performed consistently using Petri nets.

## 6. References

Hatvany, J. (1985). Intelligence and cooperation in heterarchical manufacturing systems. Robotics and Computer Integrated Manufacturing, Vol. 2, No. 2, 101-104

Hoermann, A. (1989). A Petri net based control architecture for a multi-robot system, Proceedings of IEEE International Symposium on Intelligent Control, 493-498

Jones, A. & Saleh, A. (1989). A decentralized control architecture for computer integrated manufacturing systems, Proceedings of IEEE International Symposium on Intelligent Control, 44-49

Komoda, N.; Kera, K. & Kubo, K. (1984). An autonomous, decentralized control system for factory automation. IEEE Computer, Vol. 17, No. 12, 73-83

Lee, E. J., Togueni, A. & Dangoumau, N. (2006). A Petri net based decentralized synthesis approach for the control of flexible manufacturing systems, Proceedings of the IMACS Multiconference Computational Engineering in Systems Applications

Masuda, R. & Hasegawa, K. (1981). Mark flow graph and its application to complex sequential control system, Proceedings of the 13th Hawaii International Conference on System Science, 194-203

Merabet, A. (1986). Synchronization of operations in a flexible manufacturing cell: the Petri net approach. Journal of Manufacturing Systems, Vol. 5, No. 3, 161-169

Murata, T.; Komoda, N. & Matsumoto, K. (1986). A Petri net based controller for flexible and maintainable sequence control and its applications in factory automation. IEEE Transactions on Industrial Electronics, Vol. IE-33, No.1, 1-8

Queiroz, M. H. & Cury, J. E. R. (2000). Modular supervisory control of large scale discrete event systems, Proceedings of the Fifth Workshop on Discrete Event Systems

Silva, M. (1990). Petri nets and flexible manufacturing, In: Advances in Petri Nets 1989, Lecture Notes in Computer Science, Vol. 424, Rozenberg, G., (Ed.), 374-417, Springer-Verlag, Berlin

20

# Performance Evaluation of Distributed Systems: A Component-Based Modeling Approach based on Object Oriented Petri Nets

Aladdin Masri[1, 2], Thomas Bourdeaud'hui[2] and Armand Toguyeni[2]
[1]*An-Najah National University*
*Palestine*
[2]*Ecole Centrale de Lille*
*France*

## 1. Introduction

Distributed systems (Tanenbaum, 1995) (Coulouris et al., 2001) are increasing with the development of networks. The development of computer networks has enabled the emergence of new applications benefiting from the power and flexibility offered by the distribution of their functions on different computers. We are interested more particularly in this work on the networked control of manufacturing systems. Manufacturing systems are a class of discrete event systems whose elements are interacting together to build products or to perform services. The concept of flexible manufacturing systems FMS has been introduced to develop new manufacturing systems able to produce small or average series of products.

Modeling such systems is very important to verify some properties especially performance issues. In the literature, many models have been proposed to model manufacturing systems (Toguyeni, 2006) (Sarjoughian et al., 2005) (Berruet, 2005). However, the classical modeling paradigm is generally based on a centralized point of view. Indeed, this kind of modeling does not take into account the fact that the system will be distributed when implemented over different machines, sensors, actors, etc. So, the properties obtained at the design stage are not necessarily guaranteed at the implementation stage.

In addition, the proposed models do not take into account the underlying network and protocols in terms of performance and information exchange. The behavior and design of manufacturing systems are affected by the underlying network features: performance, mobility, availability and quality of service characteristics.

A way to overcome such problems is to model these systems in a distributed way. A distributed system-model offers means to describe precisely all interesting forms of unpredictability as they occur. It takes into account each part of the system, available resources, and system changes together with the underlying network. Once this model is made, its implementation is easier since it has the same characteristic as the desired system. Nevertheless, these systems are complex: they show massive distribution, high dynamics,

and high heterogeneity. Therefore, it is necessary to model these systems in a way that provides higher degree of confidence and rigorous solutions.
To cope with this challenge, we propose the use of a component-based methodology which is consistent with the principle of distributed systems in which elements are reusable and composable units of code. The component-based approach uses generic, hierarchical and modular means to design and analyze systems. It shows that the system model can be assembled from components working together and the designer needs only to identify the good components that offer suitable services with regard to applications requirements. This methodology allows the reusability and genericity of the components which reduces the cost of the systems development.
In this chapter, we propose to model these systems with High-Level Petri Nets which is a powerful tool particularly dedicated to concurrent and distributed formalism, allowing to model both protocol and service components. The work presented in this paper is part of a larger approach on the design of distributed systems by the evaluation, in the design phase, of the impact of network protocols on the distribution of the functions of a distributed system on different computers (Masri et al., 2008-a) (Masri et al., 2008-b) (Masri et al., 2009).

## 2. Modeling with Petri nets

Petri nets have been proposed by C. A. Petri in 1962 in his PhD thesis "Communications with Automata" (Petri, 1966). Petri nets are a mathematical and graphical tool used for modeling, formal analysis, and design of different systems like computer networks, process control plants, communication protocols, production systems, asynchronous, distributed, parallel, and stochastic systems; mainly discrete event systems.
As a graphical tool, Petri nets provide a powerful communication medium between the user and the designer. Instead of using ambiguous textual description, mathematical notation difficult to understand or complex requirements, Petri nets can be represented graphically. The graphical representation makes also Petri nets intuitively very appealing.
A Petri net graph contains two types of nodes: Places "p" and Transitions "t". Graphically, places are represented by circles, while transitions are represented by rectangles, Fig. 1. Places and transitions are directly connected by arcs from places to transitions and from transitions to places. A place P0 is considered as an input place of a transition *t* if there is an arc from P0 to *t*. A place P1 is considered an output place of a transition *t* if there is an arc from *t* to P1.
Places can contain tokens represented by dots. These tokens are the marking of places. The initial marking of places is represented in the initial marking vector m0. The graphical presentation of Petri nets shows the static properties of the systems, but they also have dynamic properties resulting from the marking of a Petri net.
As a mathematical tool, a Petri net model can be described by a set of linear algebraic equations, linear matrix algebra, or other mathematical models reflecting the behavior of the system. This allows performing a formal analysis of the model and a formal check of the properties related to the behavior of the system: deadlock, concurrent operations, repetitive activities…

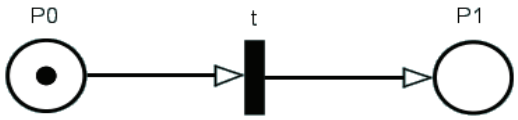



Fig. 1. Simple Petri Net

### 2.1 Communication Systems Architecture

Communication systems are designed to send messages or information from a source to one or more destinations. In general, a communication system can be represented by the functional block diagram shown in Fig. 2. The original telecommunication system was developed for voice communications.

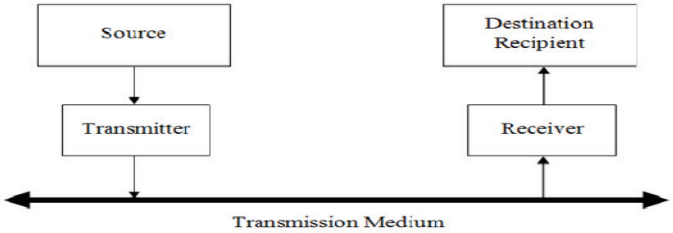



Fig. 2. Functional Diagram of Communication System

Today communication networks include all types of voice, video and data communication over copper wire, optical fibers or wireless medium. Networks (Mir, 2007) (Stallings, 2007) are organized into a hierarchy of layers where each layer has a well defined function and operates under specific protocols. The number of layers can vary from one network reference model to another but the goal of a layered structure remains common to all models. OSI model (Zimmermann, 1980) is structured in a series of 7 layers, while the TCP/IP model includes only four layers, Fig. 3.

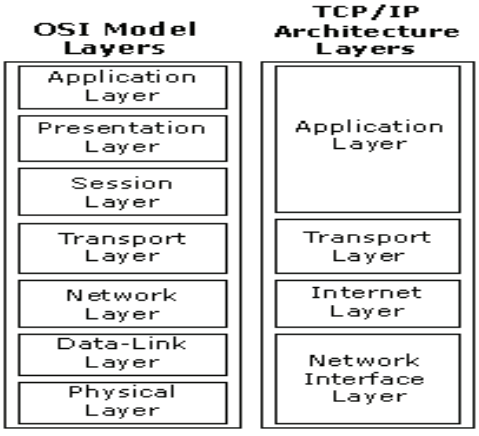



Fig. 3. OSI and TCP/IP Reference Models

Each layer consists of hardware or software elements and provides a service to the layer immediately above it. With Internet, an increasing numbers of computer networks are now connected. The concept of telecommunication system has increased the complexity significantly.

### 3.2 Proporties of our High-Level Petri Nets

In this subsection we will give a brief definition on the desired high-level Petri nets. This definition is not far from the definition of colored Petri nets (Jensen,1991). However, we add to this definition a time notation.

**Definition:** A High-Level Petri Net is a tuple N= (P, T, A, $m_0$, Σ, Λ, G, E, D) where:

- Σ is a finite set of non-empty color sets.
- Λ is a color function, Λ: P → Σ
- G is a guard function, G: T → Boolean expression, where:
  $\forall t \in T$: [Type (G(t)) = $B_{expr} \wedge$ Type (Var (G(t))) $\subseteq \Sigma$], where:
  *Type* is the color type of the guard function,
  $B_{expr}$ is a Boolean function
  Var is the variables of the guard function.
- E is an arc expression function, E: A→E(a), where:
  $\forall a \in A$: [Type(E(a)) = Λ(p(a)) $\wedge$ Type (Var (E(a))) $\subseteq \Sigma$], p(a) is the place of arc a.
- D is a delay function, D: E → TS, where TS is a delay associated to the arc inscription with the annotation symbol "@".

The arc expression function can contain any sign and/or mathematical or logical functions, such as programming language expressions. The delay function can be associated to both output arcs (from places to transitions) and input arcs (from transitions to places). The implementation of this definition will be given by different examples.

**1- Inscriptions, Guards and Tuples**

Arcs are the connectors between places and transitions. Arcs can have arc inscriptions. When a transition fires, its arc expressions are evaluated and tokens are moved according to the result. Arc inscriptions can be simple, tuples or even mathematical operators. They can be also variables or constants. However, inscriptions do not have the same meaning on both input arc and output arc.

Fig. 5 shows different arc inscriptions. In Fig. 4 (a), the arc inscription contains mathematical operation. The resulting of firing T1 is a token with value 8. While in (b) T2 can fire only if place P4 contains a token with value 5. Also tokens can be numbers or strings as in place P5. The resulting of firing T2 is a token with value "hello" in place P6. However in (c), T3 can fire with any value in place P7, but the resulting of this firing is a token with the value 5 put in place P9. Other Java signs can be also used like the "!" sign which means the not-equality, while " | " is an "*or*" sign.

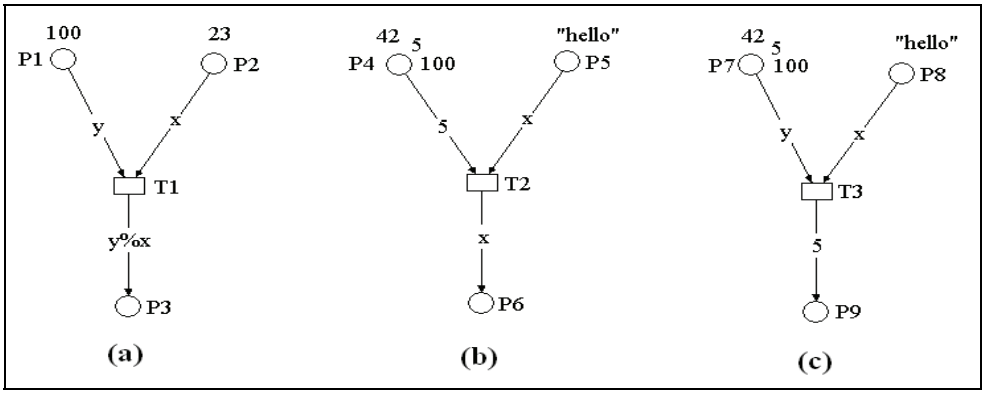



Fig. 4. Arc Inscriptions

Not like the arc inscriptions, guard inscriptions are expressions that are prefixed with the reserved word guard associated to the transitions. A transition may only fire if all of its guard inscriptions evaluate to true. Guards are the conditions that must be satisfied to fire transitions. They can be used as if statements.

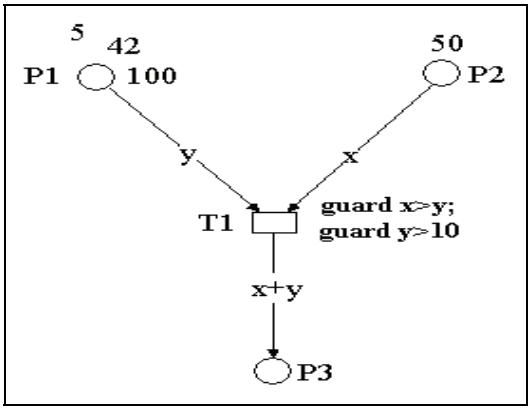



Fig. 5. Guard inscription

Fig. 5 shows an example of the guard inscription. To fire T1 both conditions must be true: *y* greater than 10 and *x* greater than *y*. The tokens with value 42 and 100 in place P1 satisfy the second condition. However, the value of token *x* is 50. So, only the token with value 42 can be used to satisfy the first condition. The resulting of firing T1 is a token with value 50 + 42 = 92 put in place P3. Guards are also useful to identify the tokens.

A tuple is denoted by a comma-separated list of expressions that is enclosed in square brackets. [1,"abc",1.4] denotes a 3-tuple which has as its components the integer 1, the string "*abc*", and the double precision float 1.4. Tuples are useful for storing a whole group of related values inside a single token and hence in a single place. A tuple, [[1,2],[3,4,5]], is a 2-tuple that has a 2-tuple as its first component and a 3-tuple as its second component. This might be useful if the components are hierarchically structured.

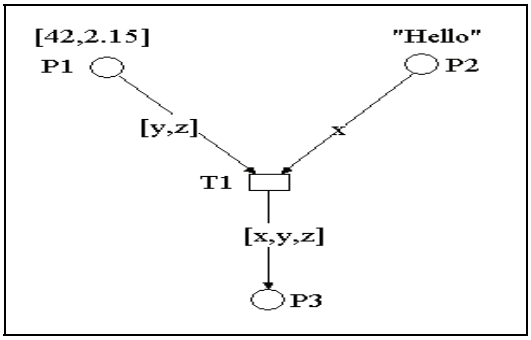



Fig. 6. Tuples

Arc inscription can modify tokens and the structure of a tuple. Fig. 6 shows an example of tuples. Tuples can be used to represent *Protocol Data Unit PDU* in communication protocols.

**2- Stochastic and Probability Function**

A stochastic process or random process is a collection of random variables. In Stochastic Petri nets, the function **Γ** is a set of firing rates that maps the set transitions T into a probability density function *f*. The entry $\delta_i \in \Gamma$ is an exponential distributed random variable, whose *f* is a negative exponential, associated with transition $\mathbf{t}_i$.

*F* is a function that represents a probability distribution in terms of integrals such as:

$$p(a \leq x \leq b) = \int_a^b f(x)\,dx = 1, \text{ for any two numbers } a \text{ and } b \quad (1)$$

The probabilistic measure **P** is a function transforming the random variables to the interval [0, 1] such that:

- P(x) is non-negative for all real *x*.
- The sum of P(x) over all the possible values that *x* can have is 1:

$$\sum_i P_i = 1 \quad (2)$$

Where $\boldsymbol{i}$ represents all the possible values of *x* and $P_i$ is the probability at $x_i$, consequence P(x) ∈ [0, 1].

Fig. 7 shows a possible probabilistic process with the *Random()* function. The function represents the generating of a random variable that can be easily implanted in *Java* to create any type of random variable (class ***RandomVariable()*** in the package ***java.lang.object*** or any Java random function). In the figure, the firing of transition T generates a 2-tuple token [x, i] in place S. In this token, *x* models the type of the object and *i* is, for example, the type of the measure of a characteristic of this object. Let us assume that *i* is a random variable in the interval [0, 1]. Because of the guards on the transitions, the token in place S can only enable one of the three transitions T1, T2 and T3. The value of *i* equals to randomly generated value of this function. The firing of the enabled transition depends on the value of *i*:

- If the value of *i* is less than 0.2, T1 can be fired and hence a token of value *x* is put in place D1.
- If the value of *i* is greater than or equals to 0.2 and less than 0.55, then T2 is the enabled and the fired transition.
- However, if the value of *i* is greater than or equal to 0.55 then T3 can be fired and hence the token *x* is put in place D3.

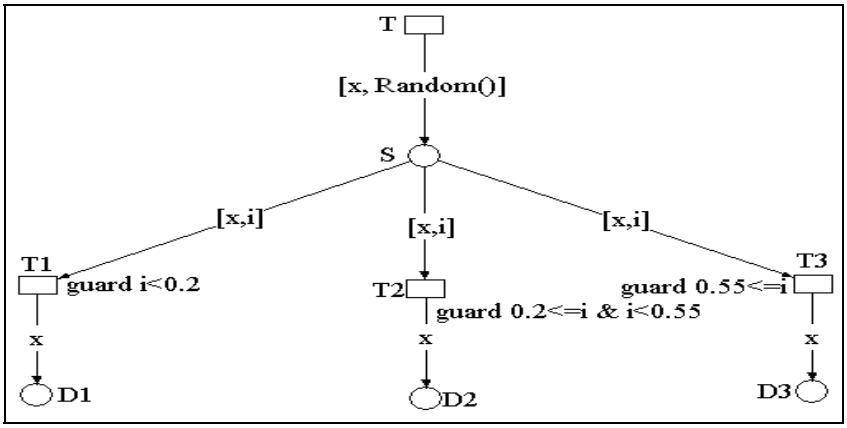



Fig. 7. Probabilistic Process with the *Random()* Function

**3- Token Identification**

Workstations exchanging messages put the source and destination addresses in the header of the message. The workstation which has the destination address can pick up the message. Token identification is very important to model this process. High-level Petri nets allow the identification of tokens.

The guard inscription on the transitions can be used to identity a token depending on its fields (in the input places). Consider the example in fig. 8, a workstation, sensing the channel for reception, can only pick up the packet if its destination address is "1" (assumed to be its address). Other verifications can be done such as the identification of the contents of the packet if it is an acknowledgement packet or data packet.

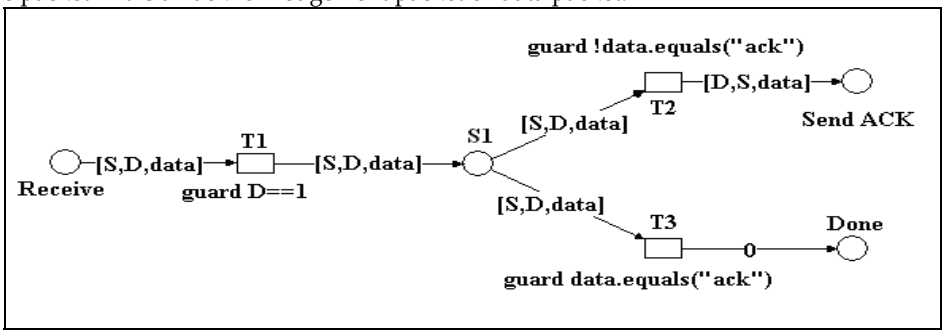



Fig. 8. Token Identification

**4- Timing**

Time notation is added to PN formalism to model time dependencies. A time stamp is attached to each token. Delays are associated to arc inscriptions in order to control the time stamps of token and the firing times of transitions. To add a delay to an arc, the symbol "@" and an expression representing the number of time units are added to the arc inscription. For example, the inscription x@5 indicates that the token must stay or will be available after 5 units of time.

Delays at the input arcs (from places to transitions) mean that a token must remain available for given time before firing the transition (timed transition). However, delays at the output arcs specify that a token is only available after some time (immediate transitions). Delays can be created by a random number generator or depend on the result of an action. Delays may depend also on the token values to delay the input token itself, which means that [x, t]@t is legal.

Timing adds another firing rule. Immediate transitions have more priority over timed transitions. To construct the vector of enabled transitions *V(t)* in the net, *local remaining time of the tokens LRT in the input places* with respect to the *arrival time* of token in the place is used. The time inscription at the output arcs of a place (input arc for a transition) only indicates the time a token must stay in that place before firing the transition. The time for each place is computed locally for each arc-transition delay, but to compute the *effective remaining time* $\alpha_t$ for each enabled transition, the *maximum local remaining time* for each input place of that transition is used:

$$\alpha_t = \max\{LRT(p_i),\ \forall p_i \in {}^\circ t\} \tag{3}$$

Where $^\circ t$ is the set of input places of transition *t*, with LRT = 0 for the input arcs with no time inscription.

Once V(t) is constructed, the transition with the *minimum remaining time* is first fired:

$$F_{Fired}(t)=\min\{\alpha_{t_i}, \text{ such that } t_i \in V(t)\} \tag{4}$$

Where $t_i \in V(t)$ is an enabled transition in the vector V(t).

In Fig. 9, transitions T1 and T2 are immediate. The inscription on the output arc between T1 and P2 indicates that the token is put (available) in place P2 after 10 units of time, but it is immediately removed from place P1. So, the arrival of a token to place P3 during the 10 units of time would not have any effect on the net since the token in place P1 has been already removed by the fire of T1. This case is similar to the firing rules found in Timed Petri nets.

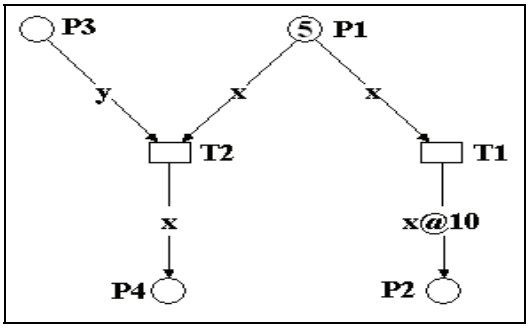



Fig. 9. Time Inscription on the Output Arcs

In Fig. 10, transition T1 is enabled but cannot fire before 10 units of time, (tokens in place P1 must stay available for 10 units of time before firing T1). After firing T1, a token with value 5 is put in place P2. However, T2 is an immediate transition since time delays are not added to any of its input arcs. So, if a token is put in place P3 during the 10 units of time, it is fired immediately and transition T1 is no longer enabled. In this special case, the firing of transition T1 is as the firing rule of a T-time Petri net with interval [10, 10].

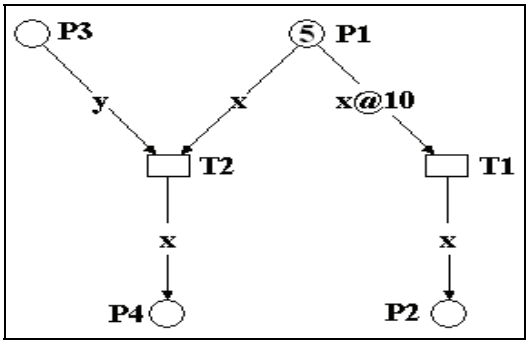



Fig. 10. Time Inscription on the Input Arcs

Fig. 11 shows the general case to find the fired transition. In the figure, the firing of T1 or T2 depends on the token arrival time in each input place (for T1: places P1 and P3; for T2: places P3 and P5). If we assume that one token is put in each place at the same time, both T1 and T2 are enabled. To compute the effective firing time, we get:

$\alpha_{T1} = \max \{2, 7\} = 7$, $\alpha_{T2} = \max \{3, 5\} = 5$

$F_{Fired}(t) = \min \{\alpha_{T1} = 7, \alpha_{T2} = 5\} = \alpha_{T2}$

So, T2 is the fired transition.

However, if we assume that a token is put in place P1 3 units of time before the arrival of the other tokens, we get:

$\alpha_{T1} = \max \{2, 4\} = 4$, $\alpha_{T2} = \max \{3, 5\} = 5$

$F_{Fired}(t) = \min \{\alpha_{T1} = 4, \alpha_{T2} = 5\} = \alpha_{T1}$

Here, we used the local remaining time for place P1 (7 – 3 = 4 units of time). Thus, the fired transition is T1 since the token in place P1 has already resided part of its staying time (time inscription on the arc).

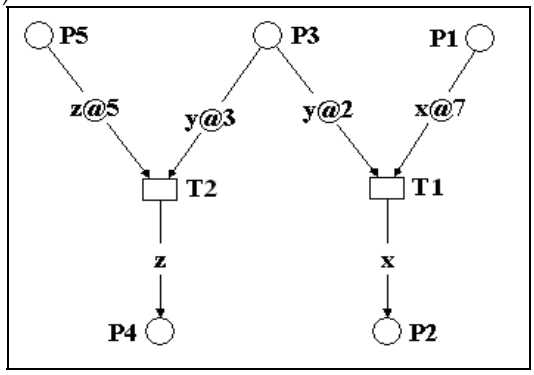



Fig. 11. Computing the Effective Firing Time

## 4. Componenet-Based Modeling

Component-based engineering (Brereton & Budgen, 2000) has a huge importance for rigorous system design methodologies. It is based on the statement which is common to all engineering disciplines: complex systems can be obtained by assembling components,

ideally commercial-off-the-shelf (COTS) (Carney and F. Long, 2000). Reusability and genericity are key factors that contribute to this success and importance. Component-based development aims at decreasing development time and costs by creating applications from reusable, easily connectible and exchangeable building blocks.
In component-based engineering research literature, several approaches (Gössler et al., 2007) (Bastide & Barboni, 2004) have focused on the aspects of the development of components. However, reusing available, ready-to-use components decreases time-to-market for new systems and applications. This may be done by selecting the appropriate components from the available components based on the needs and then assembling them to build a new component system-model.
Different methods of component specification software exist; from the Interface Description Language IDL (Object Management Groups' CORBA, java based components such as JavaBeans and Microsoft's .Net) to formal methods, by design-by-contract methods. Despite their widely difference in the details, they have a common concept: a component is a black box that is accessed through exposed interfaces.

### 4.1 Components interfaces

Components abstraction is useful for reducing the design complexity by decomposing a problem into connected components. Abstraction (or specification) describes the functional behavior of the components, i.e. components are considered to be specific to an application. Abstraction focuses on the important characteristics of component upon the designer point of view. This definition supports the abstraction of data, hiding internal function, reusability and self-contained component behaviour descriptions. Thus, during the design of components we must focus on well-defining the service offered by the component at its interfaces and the parameters that can be adapted to the application requirements, rather than spending the time on describing its internal behaviour. This can be achieved by giving appropriate names to the interfaces and parameters and documenting these interfaces and parameters.
Components can be built according to the needs of the user and different requirements and points of view. However, these components are characterized by:

- The service they offer: each component has its own functionality and service. The resulting of this service depends on the parameters and value given to the component.
- The hidden implementation: the service and functionality are hidden. However, the designer has the access to the internal code but there is no need to modify the code.
- The interfaces: to access the component service or to connect the components, interfaces are used. Several modes of connection between the different components in the model can be defined.

The component interfaces declare the services that a component offers. They are used as an access point to the component functionality by other components. Since we use Petri nets to model the different component behaviors, we used *places* to be the input interfaces of components and the output interfaces are *transitions*. The input interfaces (places) receive as many tokens as the producer components. The output interfaces (transitions) generate as many tokens as the consuming components, Fig.12.

Fig. 12. (a) input interfaces (b) output interfaces

This choice is coherent with the traditional way to model asynchronous communication between processes modeled by Petri Nets. Moreover it guarantees the genericity of the components and facilitates the connection between the different components.
The connection between interfaces of two blocks can be 1-to-many, many-to-1 or 1-to-1. As an example, Fig. 13 shows a many-to-1 and a 1-to-many connections. To illustrate the interest of this choice of interfaces, let us consider the modeling of workstations connected to a communications bus. A many-to-1 connection is used to connect workstations output transitions to a medium input place since workstations put their data on the medium only. A 1-to-many connection is used to connect the medium output transitions to workstations input places, since all the workstations can see the signals propagating on the medium.

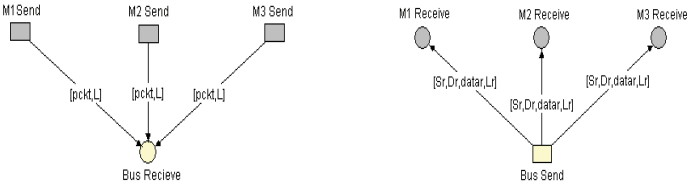



Fig. 13. (a) input interfaces (b) output interfaces

This approach is very useful to deal with the complexity due to the size of a system. Indeed, if one has already a model of some workstations connected on a bus and one wants to increase the size of its model, the connection of new workstations can be done easily just by adding an arc between the output transition of the bus model and the input place of the station model. So this does not require any modification of the bus or the workstation component. Conversely, if the transitions are used as input interfaces and places as output interfaces, the addition of a new workstation would need to add a new token in the output place, and hence modify the internal code, so we loss the genericity.

## 5. Modeling Communication Protocols

In our approach, we want to model reusable components. In this section, we will build the components that will be used to model the communication protocols. The modeling will be hierarchical since we build first the basic components. Then, with these components, we construct composite-components.
Before starting the construction of modeling components, we will analyze the data link layer protocols that we are interested in this work. These analyses will help to identify the basic common behaviors of the different protocols that lead to define basic components. These basic components are the initial brick of the library that will serve to model all the complete behavior of the different protocols.

### 5.1 A top-down analysis methodology

To build the basic components one must identify these components to be reused in different models. Since we are interested in manufacturing systems, the analyses will be made at the Data Link Layer protocols. The *Data Link Layer DLL* is the second layer in the OSI model. The data link layer is often split in two sub-layers: the *logical link control LLC* and the M*edia Access Control MAC,* Fig. 14.

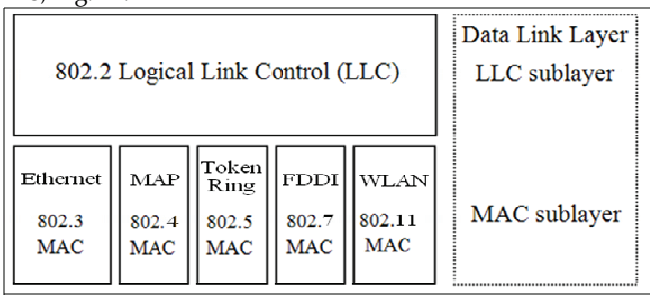



Fig. 14. IEEE MAC Sublayer

The next step is to define the protocols that have the same functionality. Here, one can find two protocols Ethernet IEEE 802.3 (IEEE, 2009) and wireless IEEE 802.11 *Distributed Coordination Function DCF* (IEEE, 2007) protocols that share the carrier sense multiple access CSMA procedure (IEEE, 2002) to send the data over the shared medium. Finally, one must find the common behaviors to associate basic components to it. The resulting of these analyses is three basic common elements:

1) Channel check:

A workstation attempting to send data must at first check if the channel is free or not. Ethernet uses the CSMA/*CD* Protocol. Here CD means collision detection. The workstation must check if the channel is free for a period of 9.6μs before it starts its transmission.

The IEEE 802.11 DCF uses the CSMA/CA protocol. Here CA means collision avoidance. To use the network, a workstation must before check if the channel is free for more than a period of time called Distributed Inter-Frame Space DIFS, Fig. 15. If so, the workstation starts a random backoff before starting its transmission. If the channel status is changed in both Ethernet and IEEE 802.11 deferring and backoff times, the workstation must restart the process of sensing the channel.

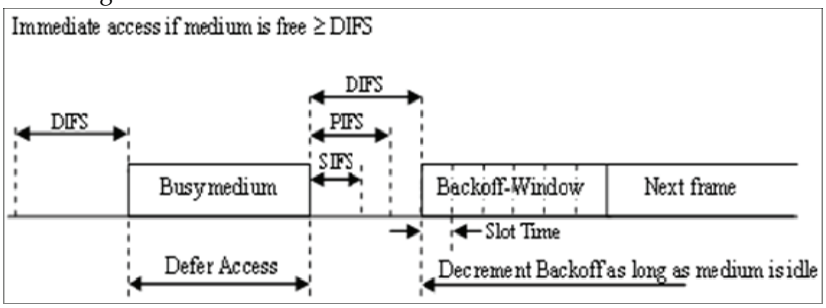



Fig. 15. Channel Access in IEEE 802.11 DCF

2) Sending and Receiving: Data, Acknowledgments and JAM:
Workstations send and receive packets. These packets can be data packets, acknowledgment packets or JAM frame (a 32-bit frame, put in place of the correct MAC CRC). In Ethernet networks, workstations receive either a data packet or a JAM after a collision. The destination workstation does not need to send an acknowledgment to the transmitter at the MAC layer.

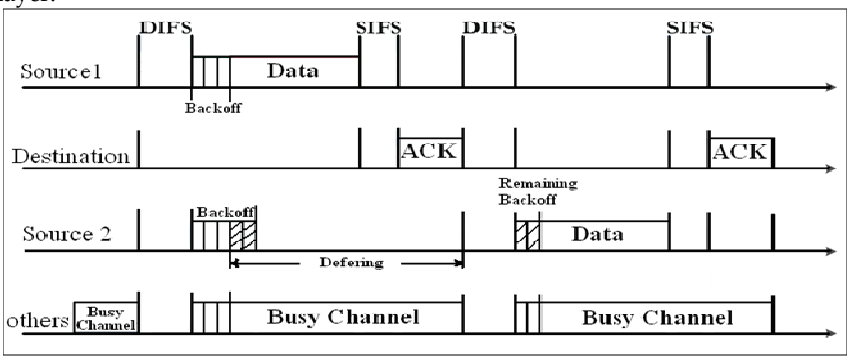



Fig. 16. Backoff mechanism in IEEE 802.11 DCF without RTS/CTS

However, in wireless LANs, the destination workstation must send an acknowledgment to the transmitter after a successful reception of a packet, Fig. 16. Otherwise, the transmitter will consider that its packet is lost or a collision has occurred, so it will retransmit this packet causing an extra load on network worthlessly. On the other hand, to send data, workstations need only to put the destination address in the packet. Since the medium is shared in most LAN technologies, all the workstations will see the packet. However, only the workstation that has the destination address reads the packet and the others will either forward it, or drop it.

3) Random and Binary Exponential Backoffs
In communication networks errors can occur. This is due to many factors like the surrounding environment, noise and interference, or because of collisions. Ethernet and IEEE 802.11 networks use the channel check and the inter-frame space to decide the medium access. Thus, collisions may occur when more than one workstation transmit on the shared medium at the same time. In Ethernet, the maximum time needed to send the first bit from one end to the other end of a 10BaseT medium is 25.6 μs. During this time, (an)other workstation(s) may attempt to send its data, as that the channel is considered as free.
As a result, a JAM signal is propagated over the shared medium informing the occurrence of a collision. Each workstation concerned by a collision starts a binary expositional backoff procedure, called *BEB*, to decide when it can do a new attempt to access the medium. The BEB algorithm computes randomly a waiting delay that increases with the number of the attempts *Tn* of the workstation.
At the beginning *Tn* equals zero. Each time a collision occurs, the workstation increments *Tn* counter until it reaches 15. Before trying to transmit its data again, the workstation starts a BEB by taking a random value between 0 and $2^X$ and multiplies it by 51.2 μs, where:

$$X = \begin{cases} \text{[illegible]}, & \text{if } 0 < \text{[illegible]} \leq 10 \\ 10, & \text{if } 10 < \text{[illegible]} \leq 15 \end{cases} \quad (5)$$

This helps in decreasing the possibility for a collision occurrence. In case of no collision, the workstation continues transmitting and when it is done it leaves the channel. However, If *Tn* reaches 15, (the load on the channel is very high), then the workstation aborts its transmission and tries it again later.
In wireless LANs, after a collision, no JAM signal is sent. However, if the workstation does not receive an acknowledgment after a period of time equals to Short IFS *SIFS* (Fig. 15), it considers that a collision has occurred and starts a backoff procedure. For each retransmission attempt, the backoff grows exponentially according to the following equation:

$$ST_{backoff} = R(0,CW) * Slot\text{-}time \quad (6)$$

Where:

- *ST* is the backoff time.
- *CW* is the *Contention Window*.
- *R* is a random function.

In general, the initial value of CW ($CW_{min}$) is 16. After each unsuccessful transmission attempt, CW is doubled until a predefined maximum $CW_{max}$ is reached (often 1024).

There are two major differences between Ethernet and IEEE 802.11 backoff processes:

1- The wireless LAN starts a backoff procedure even at the first attempt to send its data (Fig. 10), while Ethernet does not. This is one of the mechanisms used to implement the Collision Avoidance feature of CSMA/CA.
2- Ethernet starts its BEB algorithm after a collision (without conceding the status of the channel) and then restarts checking the channel to send its data.
While in IEEE 802.11, the workstation checks first the channel status and then it decrements its backoff by:

$$R = \begin{cases} R-1, & \text{if the channel is free during 1 time slot} \\ R, & \text{if the channel becoms busy} \end{cases} \quad (7)$$

The design of CSMA protocol offers fair access in a shared medium. This means that all the workstations have a chance to use the network and workstations cannot capture the channel for ever. The remaining value of R is reused after the channel status becomes free for more than a DIFS period. The workstation starts sending its data when R equals zero.

### 5.2 A bottom-up construction methodology

As one can see in the last subsection, three elements are in common. These elements can now be used to model the basic components.

*1- The channel-check component*

Fig. 17 shows a channel check component. Elements in light gray represent the places and transitions used to build the component. Elements in dark gray represent the interfaces of the component. Initially, the channel is idle for all workstations. This is represented by a token in place "Idle". A workstation that wants to send data (a token in place "Data send") must first check the channel.
In wireless LANs, the channel must be free for a period more than DIFS, while in Ethernet, it is 9.6 µs. This is represented by the '@t' at the arc between place "Idle" and transition "TF" (t' equals 9.6 µs in Ethernet and 50 µs in 802.11b). The workstation must wait before it starts transmitting, represented by a token put in place "sdata". In Ethernet the wait "@t" equals to 9.6 µs, while in 802.11 it is equal to random value between $CW_{min}$ and $CW_{max}$ slots time. Place "Backoff/Deferring Time" and transition "FC" is used to decrement the backoff in

wireless LAN, while for Ethernet, it can be left as it is in the figure (no dependence to that transition in the model).

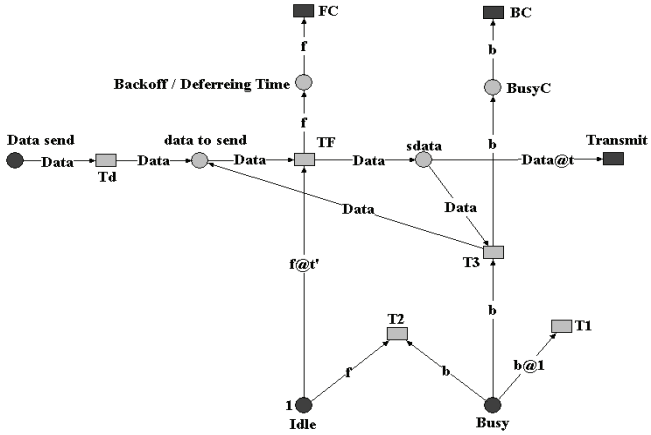



Fig. 17. Channel Check Component

Consequently, if the channel status is changed (a token is put in place "Busy"), the workstation can be in one of the following states:

- It is the transmitter (there is no more tokens in place "sdata"), then nothing is changed and the token in place "Busy" is consumed by transition T1;
- It attempt to send or it has no data to send, then T2 is fired;
- It is in the backoff/deferring phase, then T3 is fired (the workstation rechecks the channel again) and a token is put in place "BusyC" to stop decrementing the backoff. Hence, in wireless LAN, the workstation stops decrementing the backoff, but it keeps its remaining value.

In the three cases the channel status is changed from idle to busy.

Initially, this component has one token with value 1 (representing the free channel) in place Idle. The use of this component is possible in any protocol that demands the sensing the channel before transmitting data. It represents also the status of the channel free or idle. Let us notice here that, for genericity, we use two parameters t' and t to define the delay on the arc Idle-FT and arc Backoff/Deferring Time-Transmit.

2- *Receiving and sending ACK component*

Workstations receive two types of packets: data packet and ACK/JAM frames. In Ethernet network, no acknowledgment is sent after the reception of packet. Therefore, the received packet can be either a data packet or a Jam frame. While in wireless LAN, the received packet is either a data packet or an acknowledgment frame.

Fig. 18 shows the receiving and sending acknowledgment component. One assumes that a token is put in place "Receive". The fields of the token represents: the source address "Sr", the destination address "Dr", the received data "rdara" and the last field represents the lengths of the packet. The workstation checks at first the destination address "Dr" of the packet. The guard condition on transition "Address" checks if the received packet belongs to this workstation, a token is put in place "Data?". Otherwise, the token in place "Receive" is eliminated by transition "Drop". Hence, for simplicity, "Dr==1" is considered as the own address of the workstation, while "Dr==0" is used to represent the multicast or JAM frame reception.

Next, the guard condition of transition "ACK/JAM" is used to check if the received frame is an ACK frame or a JAM frame (for Ethernet only). The "abc" in the guard can be modified according to the needs of the designer and the type of network. However, if the received packet is a data packet, transition "DA" is enabled. This transition is fired after a time equals to the time needed to receive the packet modeled by the "@time(Lr)" at the outgoing arc. This "@time(Lr)" is a function that returns the time corresponding to the length "Lr" of the packet.

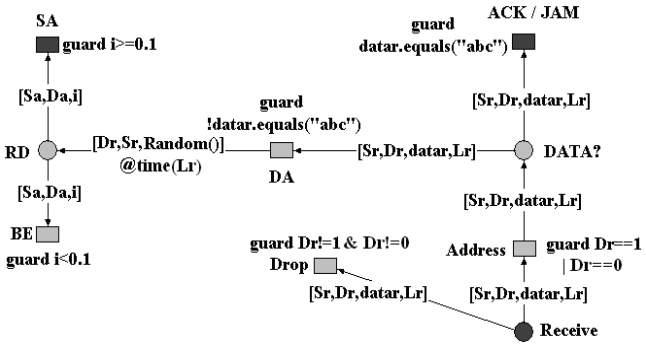



Fig. 18. Receiving and Sending ACK Component

Let us notice here, the functions dynamicity can be used to model mobility of a wireless networks nodes. This can be done since the bit rate is a function of the signal strength and that the signal strength is a function of distance. This means if the source knows the location of the destination, then the distance can be computed, and hence the time needed to send a packet is determined.

The last step is to represent the bit rate or receiving errors. The random function *Random()* is used to generate a random variable i. Assuming that the bit rate error is less than or equal to 10% of the transmitted/received packets. So, if the value of i is less than 0.1, then the packet is discarded (the token in place RD is consumed by transition "BE"). Else, the packet is received correctly and then an acknowledgment is sent, by firing transition "SA". This interface can be left unconnected in Ethernet. As we can see in Fig. 14, the modification of tuples can be done easily, just by modifying the arc inscriptions according to our needs.

As one can see, this component has an important functionality since it is used to identify the received data (own or not), the type of the received data (JAM, ACK, data frame) and the process of sending an acknowledgment after a successful reception. Thus the use of this component is possible for the protocols demanding the identification of data and the send/receive process.

3- *Backoff / BEB component*

The third component is the backoff / BEB component shown in Fig. 19. As we can see in the figure, retransmitting the packet depends on the value of n, (transitions T6 and T7). If the packet is correctly sent/received (a token is put in place "Done"), then n is reset to z (0 for Ethernet and 1 for wireless), for the next attempt to transmit, place N. However, the component inscriptions depend on the type of the network. As an example, Table II shows the differences between Ethernet and IEEE 802.11b networks.

In addition to Table 1, in Ethernet, places "FreeC" and "BusyCh" are not used (they can be left as it is), since the backoff decrement in Ethernet does not depend on the status of the channel. While in 802.11b, this interface is very important in decrementing the backoff each

time the channel is free for a slot time or the backoff is conserved if the channel status is changed to busy.

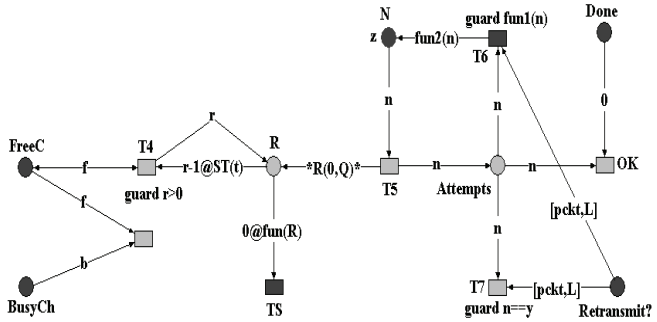



Fig. 19. Backoff / BEB Component

The firing of transition TS represents the (re)transmission allowance of a packet (backoff equals to 0). The backoff component is useful for the protocols that may need a specific timing procedure since it can be related to another components (which the case of wireless: by checking channel always) or just for standalone use.

| Variable Value | Ethernet | IEEE 802.11b |
|---|---|---|
| **fun1(n)** | n<15 | n<33 |
| **fun2(n)** | n=n+1 | n=n*2 |
| **y** | 16 | 64 |
| **z** | 0 | 1 |
| **R(0, Q)** | random(0, $2^X$), X depends on n | random(0, CW) |
| **Fun(R)** | R*51.2μs | 0 |
| **ST(t)** | 0 | Time slot (20μs) |

Table 1. Differences between Ethernet and IEEE 802.11b networks

### 5.3 Application protocols

In this subsection, we will illustrate our modeling approach through two examples: IEEE 802.3 Ethernet MAC protocol and IEEE 802.11 MAC protocol because both protocols are based on CSMA. One of the objectives is to illustrate the advantage of having generic components and the hierarchical composition that allows building composite-components.

1) Modeling an Ethernet workstation

Ethernet is the most widely used LAN technology in the world. Ethernet was designed at its beginning at the Xerox Palo Alto Research Center PARC, in 1973. The used protocol differs from the classical protocols like token control, where a station cannot send before it receives an authorization signal, the token. With Ethernet, before transmitting, a workstation must check the channel to ensure that there is no communication in progress, which is known as the CSMA/CD Protocol.

Fig. 20 shows the detailed and complete module for the Ethernet workstation. As one can see in the figure, the three components: Backoff component, Channel Check component and Receive/Send component are reused to build the workstation. To complete the model and to bind the used components together, some additional places and transitions (in white) are used to answer the specification of an Ethernet workstation.

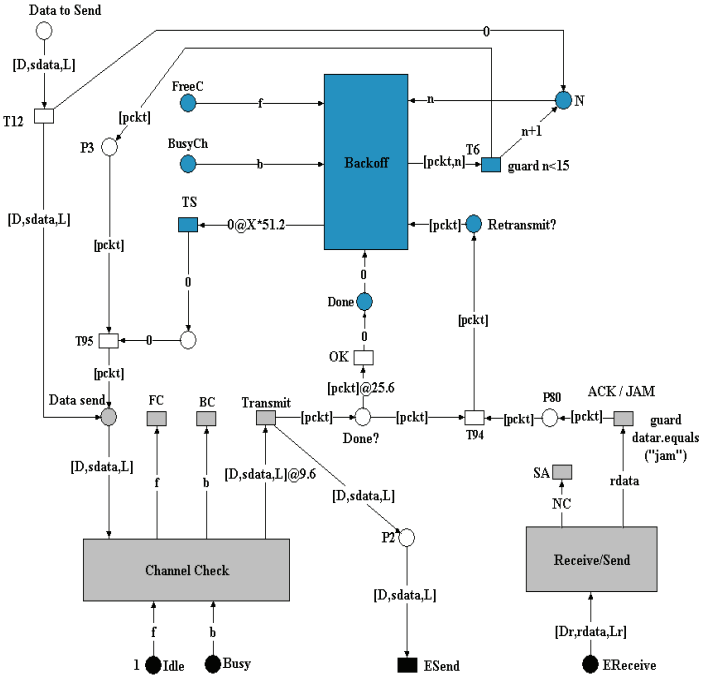



Fig. 20. Composite design of an Ethernet Workstation Component based on Generic Basic Components

In the figure, one can see that five interfaces were not connected:
An important notice is that the whole component can be reused as one component for the Ethernet workstation to build a complete Ethernet network. In other words, this new component is seen as a composite-component with the black places and transitions as the interfaces of this new component.

2) Modeling a 802.11b DCF workstation

Fig. 21 shows the detailed and complete module for the DCF IEEE 802.11b workstation model by the reuse of ready-to-use components designed from the previous sections. The workstation sets the value of N to 1 (place "N"), sense the channel (transition "TF"), sends its data (place and transition "Send") and waits for an acknowledgment (place "Wait"). If no acknowledgment is received during the SIFS period or 10μs, Transition T11 will fire putting a token in place "Retransmit?" to check if the packet can be retransmitted (transition T6) or not (transition T7).
As one can see in this figure, all the components are reused to compose the workstation module. All the interfaces were also used in this module.

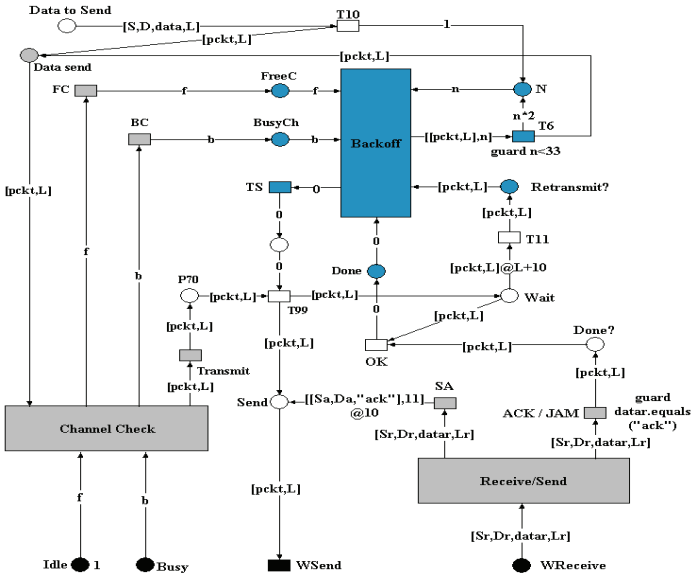



Fig. 21. Hierarchical Design of a DCF IEEE 802.11b Workstation Component based on Generic Basic Components

## 6. Experimental Validation

In the previous sections, we have modeled several components (basic and composite components). In this section, we will validate and evaluate the quality and accuracy of our model by means of simulation. The obtained results will be compared with the data given by other studies about IEEE 802.11b network and also the results of NS-2 simulations performed in the same conditions.

### 6.1 Simulations and Results

To perform the simulations, many tools and extensions of Petri nets exist such as PROD, Renew, ALPHA/Sim, CPN Tools, Artifex and other tools (Petri Nets World, 20090. However, the development of most of these tools has been stopped for a long time, they do not support our needs or they are commercial. Two main, free of charge tools were possible to cover the previous features "CPN Tools" (CPN, 2007) and "Renew 2.1.1" (Renew, 2008).
However, during simulation, "CPN Tools" has shown an important problem that does not apply to our timing needs. We have chosen "Renew" since it is a Java-based high-level Petri nets discrete-event simulator. This combination has permitted modeling all the selection criteria defined previously and more, since it allows the use of nearly all the functions offered by Java.
Our simulations are based on full-mesh dense networks with different numbers of workstations:

1- The simulations were performed for different number of workstations sharing the medium.
2- For each case, the simulations were repeated 100 times to get average measures.

3- Each simulation assumes that all nodes transmit at 11Mbps.
4- All nodes attempt to send data as soon as possible.
5- Each node has 1000 packets (to get the average possible measures) with average packet length of 1150 bytes (packet length varied from 800 byte to 1500byte).
6- All simulations were accomplished on Intel® Core™ 2 Duo Processor T2300, 2G of RAM.

*1- Average bandwidth per node*

The first result is the average bandwidth per workstation. Fig. 22 shows the throughput of 802.11b nodes sharing a bandwidth of 11Mbps. As illustrated by the figure, the bandwidth per node decreases logically with the increase of nodes number. When the number of nodes is small each workstation has more bandwidth from the shared effective bandwidth. However, when the number of the nodes on the network increases, the bandwidth is decreasing exponentially. This is due to the increased number of collisions on the network, and so more bandwidth will be lost.

The other factor is that CSMA gives fair timing to the machines to access the channel. Thus, workstations must wait longer time to have access to the channel. Another factor is after a collision, the workstations must double their contention window which means longer backoff time. So, more time is spent to decrement the backoff or less total bandwidth.

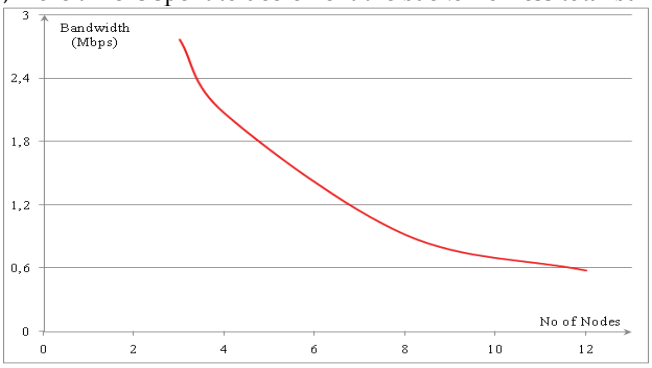



Fig. 22. Bandwidth Variation with Number of Nodes

*2- Collisions rate percentage*

The next step is to compute the collision rate percentage or errors versus the network utilization. Fig. 23 shows how the collision rate increases when the number of workstations increases. As we can see in the figure, when three workstations are sharing the medium, the collision rate is nearly 8%. However, when there are 12 workstations sharing the medium, the collision rate reaches 23.2%. These results confirm the results obtained in the previous section and our explanation.

As one can see, the collision rate is increasing linearly until certain point (8 workstations). The reason is when more workstations attempt to send, more packets are on the shared channel and hence the probability that a collision occurs increases. However, when the number increases more, the collision rate increase becomes slower. The explanation for this *evolution* is the *backoff procedure*. With more workstations, the number of collisions increases, and the value of CW also increases (backoff time). On the other hand, this increment of backoff time decreases the probability of a collision, since workstations in collision must

wait for longer time before attempting to send again. So, the collision rate increment becomes slower.

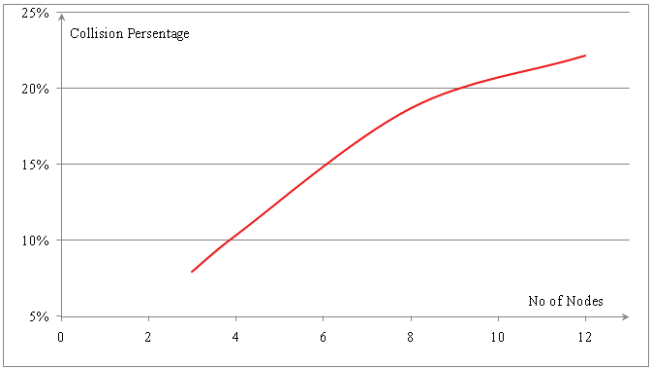



Fig. 23. Collisions Rate Percentage

*3- Transmission Time per Packet*

The next test is to measure the overall time needed to send a packet over Ethernet or DCF protocols (from sender side to receiver side). Fig. 24 shows the time required to transmit one packet versus the number of nodes on the network. The transmission time increases linearly due to the increased number of sent packets on the network and collision rate.

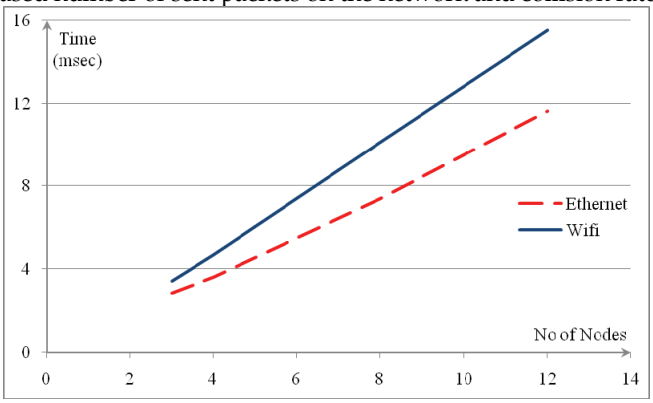



Fig. 24. Transmission Time per Packet

However, sending a packet over Ethernet requires less time than sending it over DCF. The figure shows that with three nodes on the network, DCF seems to be the same as Ethernet. However, with the increase of nodes the difference becomes obvious. This is due to:

1- A workstation attempting to use the channel in wireless networks needs to ensure that the channel is idle during a DIFS period or 50µs, while in Ethernet it only needs 9.6 µs.
2- From the first attempt to transmit, wireless nodes starts a backoff procedure (Bavg = 8 * 20 µs) decremented only if the channel is idle, while in Ethernet, workstations defers only for 9.6 µs.
3- After a collision, in wireless networks, the channel status becomes idle only when all the workstations finish their transmissions (no collision detection process),

while in Ethernet the channel becomes idle after 51.2 µs (channel acquisition slot time).

4- The backoff procedure used after each collision in wireless networks doubles the contention window value which is already 8 times greater than the one used in Ethernet. This makes the backoff in wireless greater than Ethernet BEB even with slot time (20µs) less than the 51.2 µs used in Ethernet.

### 6.3 Comparison with NS-2 simulator and other studies

To evaluate the quality and accuracy of our model, we have used the network simulator NS-2 (NS2, 2008) as a comparative tool since it is widely used to model communication protocols. The NS-2 simulator is a discrete-event network simulator that allows simulating many scenarios defined by the user. It is commonly used in the research due to its extensibility, since it is an open source model. NS2 is widely used in the simulation of routing, multicast protocols and ad-hoc network.

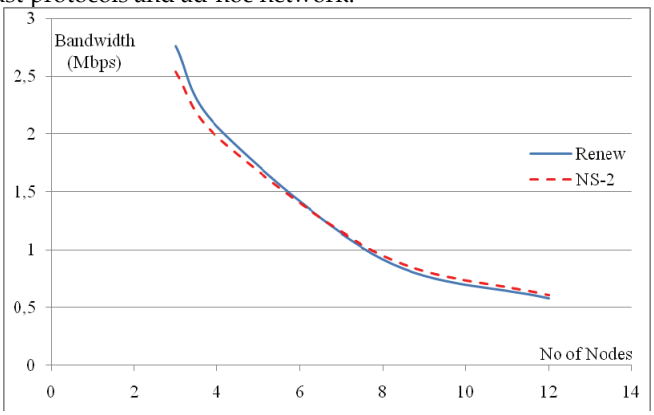



Fig. 25. Comparison between our model and NS-2

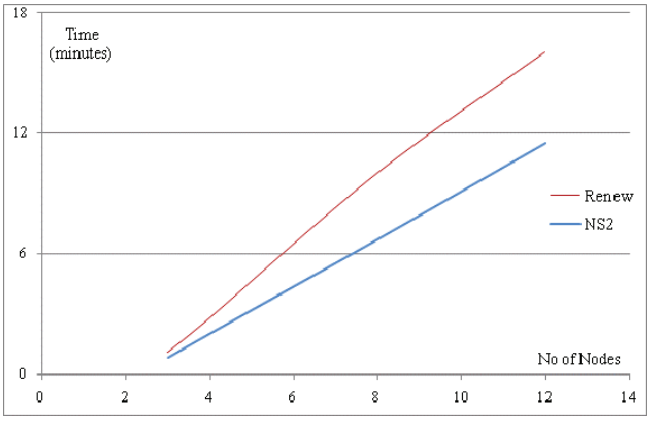



Fig. 26. Effective Simulation Time versus number of nodes

Fig. 25 shows the results obtained from NS-2 and those from our model, (Fig. 22). As we can see the results of both simulations Renew and NS-2, are nearly identical which confirms the

correctness of our model. Moreover, if we compare our obtained results with those in (Anastasi, 2005) and (Heusse, 2003), we can get also the same results from both the simulation technique and the equation we obtained from the results.
The other comparison is the effective simulation time. As we can see in Fig. 26, the simulation time increases in a linear way when the number of nodes increases. The figure shows that NS2 needs less time to perform the same simulation. However, NS2 does not support the step-by-step simulation to verify the system event by event. The second important issue is that it is not possible to model distributed services with NS2 (no supporting package). However, with "Renew" as Petri nets editor and simulator, it is possible to combine both services and protocols in one global model.

## 7. Case Study: Evaluating Performance of ADistributed Manufacturing System

In the last sections, we have shown the modeling part of the communication protocols. In this section we will show the modeling part that concerns the services. An illustrative example, Fig. 27, will be used to model the services offered by a production system. The used modeling technique will be the same as the communication protocols, i.e. component-based methodology, where each part of the system is modeled in hierarchical composition: "*service-workstation*", i.e. each service is modeled over a workstation.

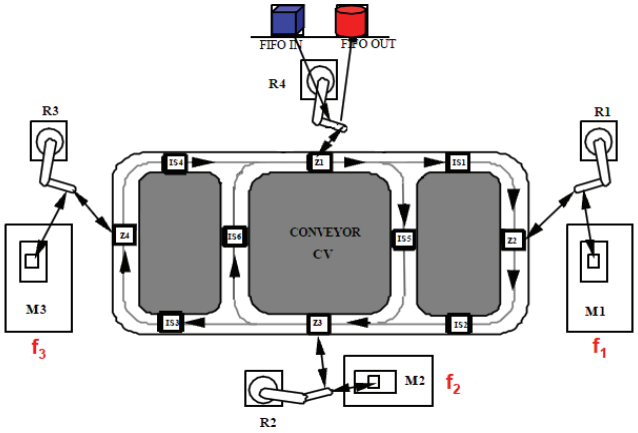



Fig. 27. Manufacturing Plant with Flexibilities

Fig. 28 shows the complete messages exchanged to transfer a product from S to D areas. Each process plays a different role with regard to the client/server mechanism. S is always a client and D is always a server. The role of S→D varies depending on the message. At first, the source area *S* (workstation) sends a request message to the transfer workstation, *S*→*D* ($T_i$ or $R_i$), containing the destination workstation *D*. $T_i$ (or $R_i$) sends a request to D, requesting a free place (*Cons-D*). If there is a free place, *D* will send a positive acknowledgment to $T_i$ (or $R_i$), otherwise S and $T_i$ (or $R_i$) will stay in a waiting period. Once $T_i$ (or $R_i$) receives the acknowledgment, it sends two messages to S containing a positive acknowledgment and a request to release the product. When the product is released S sends an acknowledgment to $T_i$ (or $R_i$) to start the transfer. When $T_i$ (or $R_i$) takes the product, it sends an end message to S

to free one its places (Cons-S). Finally, it sends a message to D asking the arrival of the product to its side. Once the product arrives to D, it sends an acknowledgement to $T_i$ (or $R_i$) informing the end of the transfer.

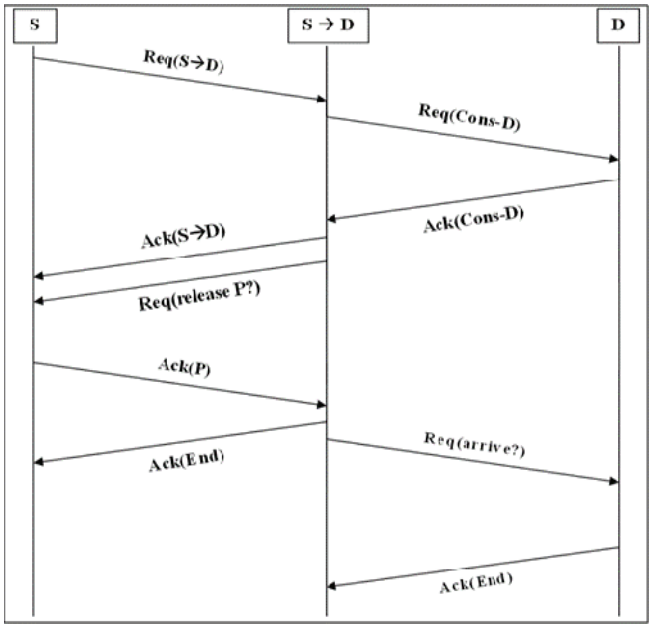



Fig. 28. Exchanged Messages over the Network for the Transfer

### 7.1 Simulqting the complete system

The simulation was performed on the same PCs used in the above sections. The system is assumed to perform 100 different products. The simulation aims to see the impact of using different type of products and different protocols over the system. The transfer time is supposed to be 50 msec and the machining time to be 100 msec. These values have been chosen in milliseconds to really verify the impact of the underlying network on the system. Otherwise, if we use the real values in minutes, the impact of the underlying network would not be obvious with the example we have used. The number of simultaneous products per type is varied from 2 to 5 products.

The type of services on the system affects the number of exchanged messages and transactions on the network. For example, to perform the service f2, the number of transactions is 72 exchanged messages per product. However, to complete service f1 or f2, the number of exchanged messages is 90 messages per product. This is in the case of one product only on the system. However, when there are several products on the system, this number increases due to collisions. So, this number may reach 90~100 messages per product for service f2, and 110~120 messages per product for service f1 or f3.

***1- One Product***

The first simulation is performed to get an idea about the time needed to machine one product over the system. Table 2 shows the impact of changing the communication protocol in the system over the time needed to finish one product. An important difference appears

between Ethernet at 10Mbps and 100Mbps. However, the 1Gbps does not create a big difference, since the machining and transfer times are the dominant in this case.
The other interesting result is the time difference when the required service is f2, or f1 or f3. Since the path to finish the product is longer, the time needed to make the product is clearly longer. In this part, 11M 802.11b *seems to be* better than 10Mbps Ethernet.

| **Service** | **802.11b** | **E-10Mbps** | **E-100Mbps** | **E-1Gbps** |
|---|---|---|---|---|
| **f2** | 564.5 ms | 567.6 ms | 506.7 ms | 500.7 ms |
| **f1 or f3** | 680.2 ms | 684.5 ms | 608.5 ms | 600.9 ms |

Table 2. Time to Machine a Product

***2- Same Products; Different Protocols***
The second results are the most important, since they show the impact of changing the communication protocol over the system.

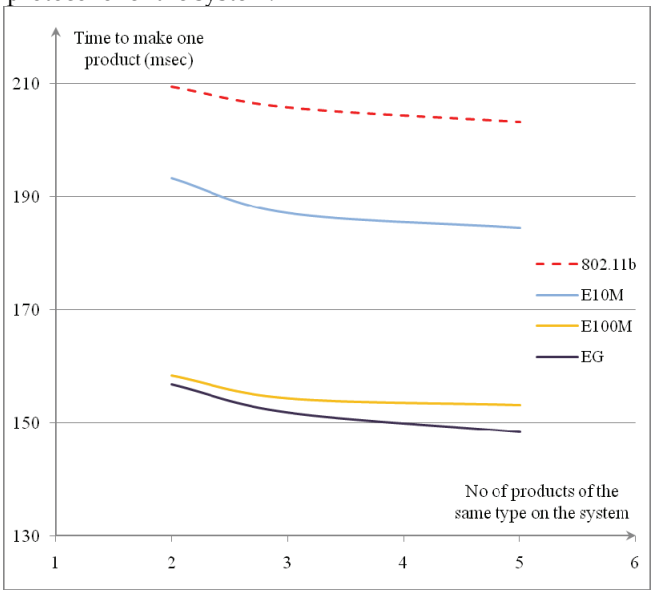



Fig. 29. Impact of changing the communication protocol in the system

Different remarks can be done from the Fig. 29:

1- 802.11b protocol does not present a good choice. This result is conforming with the results of Fig. 20. This becomes clear when the number of simultaneous products increases (the number of exchanged messages increase also).
2- A big time difference is noticed when using 100Mbps Ethernet (compared to 10Mbps Ethernet and 802.11b). The number of messages is important. With 2 simultaneous products of each type, the number of exchanged messages reaches 500 to 600 exchanged messages. With 3 simultaneous products of each type, the number of exchanged messages reaches 900 to 1000 exchanged messages. While with 5 simultaneous products of each type, there are nearly 1400 to 1500 exchanged messages on the network.

The type and speed of protocols is very important since to exchange this huge number of messages on the network, the bit rate is very important and decreases obviously the time needed to exchange these messages between the different resource/workstation on the system.

3- The use of 1Gbps Ethernet did not show a big difference with respect to 100Mbps Ethernet. However, this conclusion is not really correct. The impact of using Giga Ethernet can appear if the modeled system is larger (more machines, stock areas, resources, etc.).
In that case, the number of exchanged messages over the network will be greater. Thus, the impact of using Giga Ethernet will become obvious since the time needed to send these messages will be shorter (for example, as the time difference between 10 and 100Mbps).
However, in our model the number of modeled components is still medium (3 machines, 4 resource areas and 6 stock areas). So, the machining and transfer times are dominant here when using Giga Ethernet compared to 100Mbps Ethernet.

## 8. Conclusion

Distributed systems are more and more present in our daily life. These systems are complex and can be distributed in one place or even everywhere in the world. The use of distributed systems allows sharing different and expensive resources by a several clients. Thus, the need to control the distributed systems is very important. Manufacturing systems are one kind of these systems.
The need to model these systems before their implementation is important. The design stage allows verifying some of their properties. A well-designed model that takes into accounts all the requirements and constraints of a system can save cost and time.
In this work, we have presented the problem of modeling manufacturing systems and the underlying communication protocols. However, modeling a huge and complex system implies to have also a big and complex model. So, we have proposed in this work a component-based modeling approach based on High-Level Petri Nets.
This approach can meet the challenges of modeling the distributed systems and the communication networks. Genericity, modularity and reusability are the main and important characteristics of this approach since it allows reusing ready-to-use components and easily fitting them to new system-models depending on the requirements of clients and applications. These advantages and more allow building complex system-models in an easier way.

## 9. References

A. Masri, T. Bourdeaud'huy, and A. Toguyeni, "A Component Modular Modeling Approach Based on Object Oriented Petri Nets for the Performance Analysis of Distributed Discrete Event Systems," Fifth International Conference on Networking and Services ICNS, pp.222-227, Spain, 2009.

A. Masri, T. Bourdeaud'huy, and A. Toguyeni, "Performance Analysis of IEEE 802.11b Wireless Networks with Object Oriented Petri Nets", Electronic Notes in Theoretical Computer Science, Proceedings of First International Workshop on Formal Methods for Wireless Systems FMWS'08/CONCUR'08, Vol. 242, No 2, pp. 73-85, Canada, 2008.

A. Masri, T. Bourdeaud'huy, and A. Toguyeni, "Network Protocol Modeling: A Time Petri Net Modular Approach." 16th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2008, pp. 274-278, Croatia, 2008.

A. Tanenbaum, "Distributed Operating Systems". Prentice Hall, 1995.

A. Toguyeni, "Design of Modular and Hierarchical Controllers for Reconfigurable Manufacturing Systems." IMACS Multiconference on Computational Engineering in Systems Applications, Vol. 1, pp. 1004-1011, 2006.

C. A. Petri, "Communication with Automata." Technical Report RADC-TR-65-377 Rome Air Dev. Center, New York, 1966.

Computer Tool for Coloured Petri Nets . CPN Tools: http://wiki.daimi.au.dk/ cpntools/cpntools.wiki - 2007.

D. Carney and F. Long, "What Do You Mean by COTS? Finally, a Useful Answer." IEEE Software, 2000.

G. Anastasi, E. Borgia, M. Conti, and E. Gregori, "IEEE 802.11b Ad Hoc Networks: Performance Measurements." Cluster Computing, Vol. 8, No. 2-3, 2005.

G. Coulouris, J. Dollimore, and T. Kindberg, "Distributed Systems: Concepts and Design", 3rd ed. Pearson Education, 2001.

G. Gössler, S. Graf, M. Majster-Cederbaum, M. Martens, and J. Sifakis, "An Approach to Modelling and Verification of Component Based Systems." Lecture Notes in Computer Science, SOFSEM 2007: Theory and Practice of Computer Science, Vol. 4362, pp. 295-308, 2007.

H. Sarjoughian, W. Wang, K. Kempf, and H. Mittelmann, "Hybrid discrete event simulation with model predictive control for semiconductor supply-chain manufacturing." Proceedings of the 37th Conference on Winter Simulation, pp. 256 – 266, 2005.

H. Zimmermann, "OS1 Reference Model-The IS0 Model of Architecture for Open Systems Interconnection." IEEE Transactions on Communications, Vol. COM-28, No. 4, 1980.IEEE 802.3 Ethernet Working Group: http://www.ieee802.org/3/ - 2009.IEEE Computer Society, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." IEEE Std. 802.11™, 2007.

IEEE Std 802.3™, "Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications." 2002.

K. Jensen, "Coloured Petri nets: A high level language for system design and analysis", Lecture Notes in Computer Science, Vol. 483/1991, pp. 342-416, 1991.

N. Mir, "Computer and Communication Networks". Prentice Hall, Inc, 2007.NS2 Official Website: http://nsnam.isi.edu/nsnam/index.php/Main_Page - 2008.

M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11 b." INFOCOM, 2003.

P. Berruet, J. Lallican, A. Rossi, and J-L. Philippe, "A component based approach for the design of FMS control and supervision." IEEE International Conference on Systems, Man and Cybernetics, Vol. 4, pp. 3005-3011, 2005.

W. Stallings, "Data and Computer Communications", 8th ed. Prentice-Hall, Inc, 2007.

P. Brereton and D. Budgen, "Component-Based Systems: A Classification of Issues." IEEE Computer, Vol. 33, No. 11, pp. 54-62, 2000.

Petri Nets World: http://www.petrinetz.de/ - 2009.

R. Bastide and E. Barboni, “Component-Based Behavioural Modelling with High-Level Petri Nets.” In MOCA’04 Aahrus, Denmark, DAIMI, pp. 37-46, 2004.
The Reference Net Workshop. Renew : http://www.renew.de/ - 2008.

# Using Petri Nets to Model and Simulation Production Systems in Process Reengineering (case study)

Pawlewski Pawel
*Poznan University of Technology*
*Poland*

## 1. Introduction

Research presented in this chapter was performed in a marine diesel engines factory. The production process characterizes very long production cycle of at least 6 to 9 months. The products and production processes are unique, and based on specific customer orders. One of the main issues in the production process is occurrence of various disturbances, such as, for example, machine breakdowns, parts inconsistent with specifications, changes of customers requirements, etc. The manufacturer has identified the need to reengineer the production process of a marine diesel engine crankcase in order to increase responsiveness and final product quality. Using simulation in the reengineering process planning was extremely beneficial, as any changes in the actual production process induce considerable costs, primarily due to physical product size and use of the highly specialized processing tools. The aim of the simulation was to provide the guarantee that the solution chosen would be suitable for the company. The Petri Net methodology was applied in simulation development. Due to the fact that the outputs obtained from simulation process were not sufficient to make final decision, additional information was collected with Rapid Re methodology. In the paper conclusions from these approaches concerning modeling and simulation are presented in context of reengineering methodologies.
The chapter is organized as follows: in section 2 production process of a marine diesel engine crankcase is identified and theoretical research framework regarding modeling methods is discussed. The methodology based on Petri Net with discussion about other methods is described in section 3. Section 4 presents comparison of approaches to reengineering using a cycle of organized actions defined by Le Chatelier as the point for analysis. In section 5 additional research based on methodology Rapid Re is presented and hybrid solution based on Petri Net and Rapid Re including activities classification applied in ASME (American Society for Mechanical Engineers) methodology is proposed in section 6.

## 2. Manufacturing process identification

HCP is a factory established in 1857 in Poznań (Poland), and named by its founder Hipolit Cegielski. Nowadays it is the biggest ship engines producer in Europe. HCP produces slow speed-rotation engines for transport ships. The ship engines are built to special orders of customers. Even two engines of same type may have some differences depending on a customer's wishes. HCP builds about 25 – 35 engines per year.

The engines are built under the license of Sulzer Brothers (Wartsila) and Burmeister & Wain. These two license-providing companies are the largest and most significant ship engine constructors. In the late 80's, power plants using combustion engines produced by HCP were built in several Greek islands. Since its establishment till 2008, HCP company produced approximately 1600 diesel ships and stationary engines. The dimensions of such engines are impressive: over 4 meters wide, over 20 meter long, almost 16 meters high. HCP company is the only in Poland and the largest in Europe manufacturer of engines of this kind.

In HCP company, the ship engines are produced to specific orders of customers. Most orders are from the Polish shipyards in Gdynia, Gdansk and Szczecin. The engines are also produced for export, mainly to Germany. A ship engine is a product of a very high capital intensity, and that is why financing its production must be supported by guarantees and bank loans. Therefore, in the company there are two sale schedules: customer sales schedule and optional sales schedule. The customer sales schedule takes into account those engines which are secured by bank guarantees. The optional sales schedule includes the engines which do not have the guarantees yet. Due to the length of the engine production cycle, the customer sales schedule is prepared two years before it is introduced, and the optional sales schedules even three years before the introduction. Obviously, in the meantime the schedules are modified, since owing to the instability of the shipyard industry frequent modifications are necessary.

At the moment, HCP company produces approximately 10 types of engines. An average length of the production process runs at the level of nine months. The manufacturing process in the enterprise takes place in four divisions: welding shop, processing, assembly and packing department. A particular stage of production is assigned to each of the divisions.

In the welding shop the most important and the largest parts of an engine, i.e. motor base and crankcase, are welded. Each of these parts weighs even up to 120 tones. The welding cycle for each of the parts lasts about three months, and therefore, at the same time a few motor bases and crankcases are being welded.

The next stage of a ship engine production is processing. This stage of production lasts about 2 to 3 months. At this stage uncomplicated parts such as e.g. head screws are produced, and crankcases, motor bases, sleeves and cylinder blocks are processed. The processing division is equipped with simple, commonly-used turning lathes as well as planer mills rarely found in industry, due to their size. The planer mills are used for the processing of motor bases, crankcase and cylinder blocks.

The assembly stage can be divided into two smaller stages, i.e. assembling of particular components and final assembly. At the first stage the components of an engine are assembled, and at the final assembly stage the ready-made engine is assembled. The primary assembly stage finishes with an acceptance test. Assembling of components lasts for a month and the primary assembly takes a month and a half.

The last stage of an engine production consists of taking an engine into pieces and dispatching it to a customer. Due to the large size of an engine it is not possible to dispatch it as a whole. The disassembly and dispatch stage lasts from two to four weeks depending on the engine size.
At the moment there is a distinct trend to order smaller ship engines purposed mainly for ships used to transport cars. This tendency brings certain organization problems to the company. The difference in labor consumption in the production of small and large engines runs up to 30 per cent; as for processing in the strategic machines the difference runs at the level of 10 to 15 per cent. In such circumstances, the company enters more contacts in order to maintain the steady level of employment. It results in an increase in the load of strategic machines. An average total cycle of processing a crankcase, cylinder blocks and a motor base lasts for 25 – 30 days. It can be easily calculated that in a year scale the company is able to manufacture twenty such sets. Therefore, it is necessary to optimally use the production potential of the strategic machines.

Fig. 1. Framework of relation of the basic functions involved in marine diesel engine factory (source: HCP)

Flow of the materials presented in figure 1, is an integral part of a ship engine production process. Initiating the research the method of modeling the production process must be choose.
The choice of the modeling language was made taking into consideration the results of projects implemented in the years 2002-2006 as a part of 6th EU Framework Project. One of the reports (Lucas et al., 2005) applied to modeling language comparison. 70 languages were selected for the comparison: ADELE-TEMPO, ALF, AMBER, APPEL, APPL/A, ARIS, Articulator, BAM, BPEL4WS, BPML, Chou-UML, CIMOSA, Converstion Builder, CSP, CSPL,E3, EAI, ebXML, EDOC, EEML, ENVI12204, EPC, EPOS, EVPL, FUNSOFT, GEM,

GRAI, GRAPPLE, Hakoniwa, HFSP, IDEF, IEM, ITM, JIL, LATIN, LOTOS, LSPL, MARVEL, Melmac, Merlin, MVP-L, OIKOS, OORAM, PADM, PEACE+, Petri Net, PMDB+, Process Weaver, Promenade, PSL, RAD, REA, Rosetta Net, SDL, SLANG, Socca, SPADE, SPELL, SPM, STATEMENT, System Dynamics, TEMPO, UEML, UML, UML2, UPM,Woflan, WPDL, XPDL, YAWL . Eighteen criteria were defined in the Project – table 1.

| Code | General requirements | Inclusion/exclusion criteria – Rule |
|---|---|---|
| L01 | Formality | Semi-formal |
| L02 | Expressiveness-Information | Attributes and structure |
| L03 | Expressiveness-Actors | Actors as external entities (environment) and entities implied in the process |
| L04 | Expressiveness-Dynamics | Representation of behavior of entities |
| L05 | Expressiveness-Process | Representation of process features |
| L06 | Expressiveness-Real Time | Do not needed: exclude |
| L07 | Graphical | Visual views of models. Visual interface |
| L08 | Textual | Textual descriptions of model elements. Semantics |
| L09 | Abstraction and modularization | Different levels of abstraction. Modules containing logical set of model elements |
| L10 | Extendibility | Modeling elements to change function or semantics of the model basic elements obtaining new representation elements |
| L11 | Executability | Do not needed: exclude |
| L12 | Analyzability | Do not needed: exclude |
| L13 | Evolution ability | Do not needed: exclude |
| L14 | Multiple conceptual perspectives/views | Different kinds of views of the same process under the corresponding modeling interests |
| L15 | Computer support | At least one computer tool to support the language |
| L16 | Availability | Free specifications and tools |
| L17 | Maintainability | At least a permanent nonprofit organization (Open Source). Commercial corporations would be the second choice |
| L18 | Standardization | The language is a standard de facto or is a OMG, ISO standard |

Table 1. Inclusion/exclusion criteria for modeling language comparison (Lucas et al., 2005)

The languages were compared on the basis of these criteria. The language which met the criteria best was IDEF0. As a result of the reviewers' suggestion, another comparison was made (with the UML language) and the results of this work were presented in a report

(Galan et al., 2005). The basic argument in favor of the choice of IDEF0 was the clarity of the model structure and the facts that IDEF0 models are easy to comprehend. It should be noted that the authors of the report (Galan et al., 2005, pp. 18) conclude that UML definitely is not the best solution unless the software development is the main purpose. These conclusions correspond to the works which were presented at MOSIM2008 (Pawlewski & Fertsch, 2008) conference and by others authors (Romero & Agost, 2008).
The description of the IDEF0 language can be found in (Zakarian & Kusiak, 2001) and its full definition is included in the standard (Processing 183, 1993), (U.S.Air Force, 1981). One of its features is hierarchical absorption and decomposition capacity depending on the point of view. This feature is illustrated by Fig. 2.

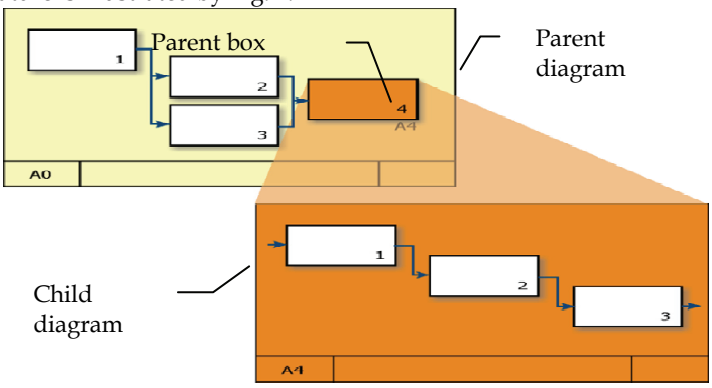



Fig. 2. Hierarchical decomposition in the IDEF0 language (Processing 183, 1993)

The capacity of hierarchical absorption in the IDEF0 language is a great advantage of the language because it can correspond (Pawlewski et al., 2008) to the complexity of the production units expressed by their classification. According to the complexity level, the following production units can be distinguished:
Zero level complexity production units – workstations. These are machines, storage and handing tools area, equipment for handling materials and/or containers with materials within a workplace, area where workers and operators stay to perform their job, the staff operating at a workstation.
First level complexity production units – in the American and European model of production organization there are numerous forms of production units of the first complexity degree (job shops). They consist of production units of zero level of complexity, transport ways, area for storing materials. The typical forms are: lines, seats, workshop, work crew.
Second level complexity production units – divisions. They consist of production units of the first level of complexity and supportive and service units, such as tool distribution shops, material warehouses, tool handling warehouses.
Third level complexity production units – departments. They comprise of all the above-mentioned production units as well as administration units, supervising units, i.e. planning section, repairs-maintenance section
The project was based on 6RTA62U ship crankcase manufacturing process. The engine is manufactured by HCP (Hipolit Cegielski Poznan) and licensed by Warstsila. It is a slow-rotation two-stroke engine for cargo ships. The engine works with the speed of 92 to 115

rpm. The engine has 6 cylinders, diameter 620 mm. The horsepower of the engine is 15 550. Its size is 10.63 m of height, 5.25 m of width and 7.5 of length. The manufacturing cycle is about 7-8 months. Figure 3 presents the crankcase element of a ship engine.

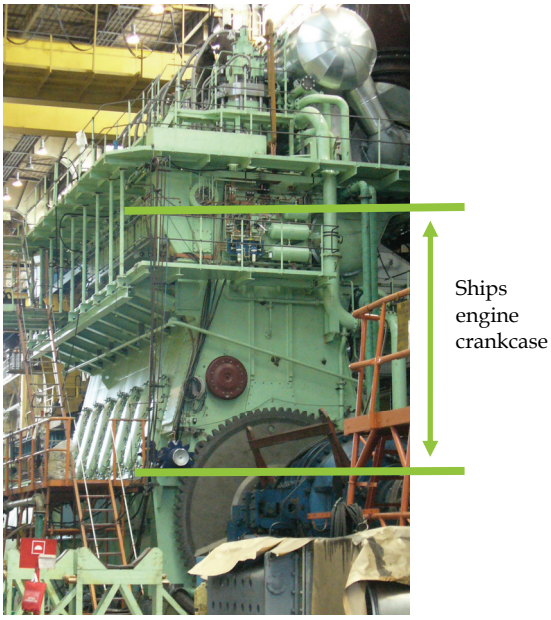



Fig. 3. Crankcase element of ships engine. (source: HCP)

At the stage of identification a process card is designed. It is a fundamental template for data collection, as it includes activities carried out within the ship engine crankcase manufacturing process. Based on the information presented in the form of process cards, a process map can be developed. IDEF0 methodology is a tool for map drawing up, using software tool AIOWin from KBSI Company – www.kbsi.com.
Figure 4 presents the main sub-processes of ship crankcase construction process: burning, execution of prow part, execution of stern part, joining prow and stern parts, treatment.
The whole process is composed of 58 operations - figure 5 shows the part of the process structure in a form of a tree graph.

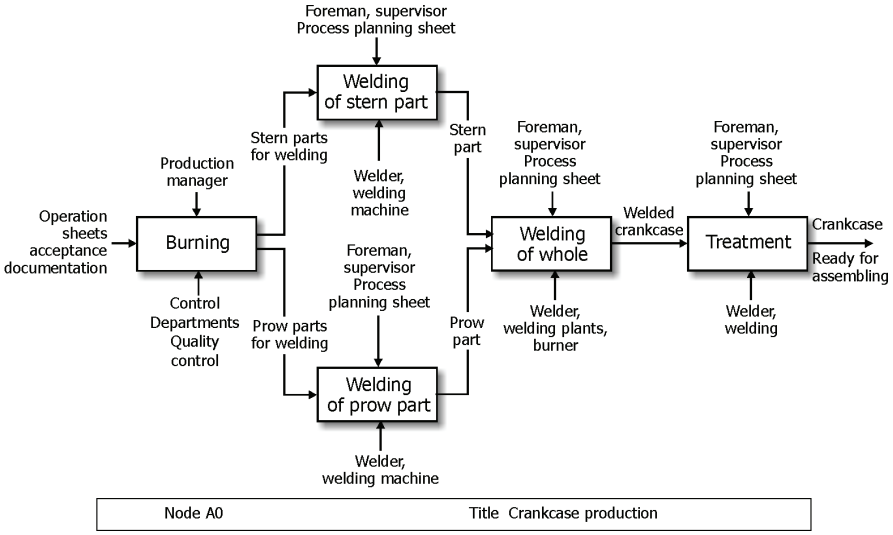



Fig. 4. Graph (IDEF0) of five main sub-processes in ship crankcase manufacturing process (source HCP).

.

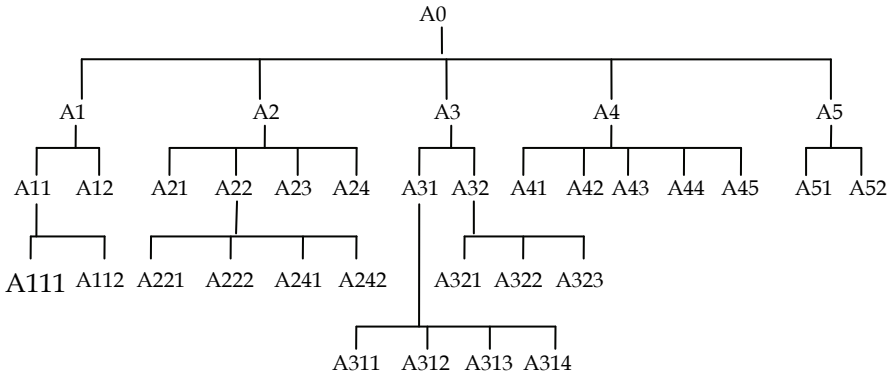



Fig. 5. The part of tree graph of ship crankcase manufacturing activities (source HCP).

## 3. Methodology based on Petri Net

Based on the paper (Jansen-Vullers & Netjes, 2006) standard modeling and simulation procedure is described as follows: "Regarding the simulation of business processes a number of steps can be distinguished. First the business process is mapped onto a process model, possibly supplemented with process documentation facilities. Then the sub processes and activities are identified. The control flow definition is created by identifying the entities that flow through the system and describing the connectors that link the different parts of the process. Lastly, the resources are identified and assigned to the activities where they are necessary. The process model should be verified to ensure that the model does not contain errors. Before simulation of a business process, the performance

characteristics, such as throughput time and resource utilization, need to be included. For statistically valid simulation results a simulation run should consists of multiple sub runs and each of these sub runs should have a sufficient run length. During the simulation, the simulation clock advances. The simulation tool may show an animated picture of the process flow or real-time fluctuations in the key performance measures. When the simulation has been finished, the simulation results can be analyzed. To draw useful and correct conclusions from these results, statistical input and output data analysis is performed."

In case when the simulation is used to confirm the choice of improvements for process reengineering, modeling and simulation procedure presented is insufficient. Applying simulation to reengineering process was beneficial as all the changes in real production process are connected with high costs, due to big size of product and application of the highly specialized processing tools. Based on performed researches definition of following requirements for simulation is possible:

- precise process definition ,
- multi process simulation where use of many processes is possible,
- observation of the situation where the processes "chase one another",
- interactions between processes,
- simulation of disturbances.
- formal verification of simulation model – reduced role of heuristic methods.

Based on literature review (Jansen-Vullers & Netjes, 2006) first decision – choice of Petri Net methods was correct. These methods provide formal semantics which enable precise and unambiguous description of the behavior of the modeled process and verification. Conclusion that many modeling techniques lack for formal semantics and thus powerful analysis methods and tools is in (Aalst & Hee, 1996). Conclusion that there are three good reasons for using Petri Net based methods which appear to be critical in large Business Process Management projects is in (Aalst, 1996). These reasons are:

- the existence of formal semantics despite the graphical nature,
- the state based diagrams instead of event based diagrams (as can be encountered in many workflow products)
- the abundance of analysis techniques.

Discussion and evaluation of tools for process simulation is presented in (Jansen-Vullers & Netjes, 2006). From many tools like ARIS, Arena, CPN Tools etc. based on results of evaluation, the tool CPN Tools was selected. Authors of this papers benefit a lot from the formal verification techniques.

One of presented requirements for simulation is simulation of disturbances. It is problem hard to solve using Petri Net especially with respect to problem of dynamic changes in process structure. It is reason why research in area of reconfigurable manufacturing systems was performed. In detail the logic control design methodologies were reviewed. Today a vast of industrial logic controllers are performed under computer named Programmable Logic Controllers (PLCs). The PLCs are specially designed to respond to use as controllers on industrial processes. A recent research work (Johnson, 2002) published on Control Engineering Journal shows that 96% of those polled programs are using leader diagrams. Ones of principal inconvenient of leader diagrams is shown in the complexity of manufacturing systems. Several alternatives have been developed to lead diagrams to PLCs programming. In particular the standard IEC61131-3 publication and IEC61499 is directed to

resolve some above problems (Lewis, 2001). The two formalisms more used for control manufacturing system are the Finite States Machine (FSM) and the Petri Net (Genc & Lafortune, 2003), (Park et al., 1999). However the real complex controller cases design though Petri Net are few (Gollapudi & Tillbury, 2001).
The state charts are an alternative framework that allows to describe the behavioral of a complex system in a compact form (Harel & Naamad, 1996), (Harel et al., 1987). Similarly to a Petri Nets they have a good concurrence. The complexity of semantic of execution does that the verification of control systems modeled with state charts be a hard task (Gruer et al., 1998). On the other hand the supervisory control theory (Ramadge & Wonham, 1987), (Ramadge & Wonham, 1989) resolves problem in specific cases, when a supervisor creates exactly the desire behavior in a close loop, even when the controller cannot control or observe all events (Cassandras & Lafortune, 1999), (Kumar & Garg, 1995). Supervisor control theory could be not adequate to be used in developing of complex manufacturing control systems reading (Charbonnier et al., 1999). The principal reason is that the controllers of supervisor control theory are designed to prevent the effects by already events despite of to try controlling the reasons that they create. The work (Trujillo, 2004) proposes solution for studying system by behavioral description. A principal reason has been the behavioral description of logic control in complex manufacturing plants. The FSM framework is unavailable to represent adequately complex systems, since its required representation millions of states (Endsley & Tilbury, 2004). Proposed solution (Trujillo, 2004) has properties that allow developing a method thought Virtual Supervisor tree, which identify all possible sequences in a control process. These sequences rapidly can be verified outline, and composed in a safe and fast form obtaining feasible pattern sequences (Trujillo, 2004). Pattern sequences are capable to compose for a new control reconfiguration.
A Petri Net (Aalst et al,. 2000), (Chen, 1990), (Peterson, 1981) is one of the several mathematical representations of discrete distributed systems. As a modeling language, it graphically depicts the structure of a distributed system as a directed bipartite graph with annotations. As such, a Petri Net has place nodes, transition nodes, and directed arcs connecting places with transitions. Petri Nets were presented in 1962 by Carl Adam Petri in his Ph.D. thesis.
Nets enable a survey of system features and they are applied for a description and study of information processing systems. Their theory is becoming one of the basic research directions. They are mostly applied in data analysis, software engineering, work organization, parallel programming. Lately, a large number of research and theoretical works concerning the application of a Petri Net in the business process modeling has been published (Aalst et al., 2000). These publications provided a strong impetus for the project presented.
For the purpose of modeling the ships crankcase manufacturing process with a Petri Net, the following procedure has been drawn up:
STEP I - Choice of the process that is to be modeled
STEP II - Definition of the initial stage – In this case, initial stages are technological processes and process maps
STEP III - Definition of the place - places represent such factors as: communication methods, conditions or states. In the analyzed process, the following places are distinguished: finished sheets, burning process

STEP IV - Definition of transitions – Transitions define such variables as shifts, events, transformations e. g. burning process, control
STEP V - Definition of tokens - Tokens represent such objects as: human resources, machines, goods, states of objects, conditions, information, state indicators (e.g. indicator of the state in which a process or object is)
STEP VI - Modeling of relations between places, transitions and tokens with tree graphs. It consists of a division of crankcase manufacturing process into successive production stages, which are parts of the ship crankcase manufacturing process. They are connected by means of arrows
STEP VII - Definition of attainable states – an attainable state is a state which can be achieved from the current state, arising because of starting the sequence of possible shifts, i. e. shifts between tokens and transitions. In the analyzed case, the attainable states are: burning process, manufacturing process of the crankcase stern part, manufacturing process of the crankcase prow part, process of joining prow and stern parts
STEP VIII - Definition of dead states – a dead state is a state in which no shift is possible. Such states are not distinguished in the conducted research
STEP IX - The model is transferred to Visual Object Net software
STEP X - Conclusions and evaluations

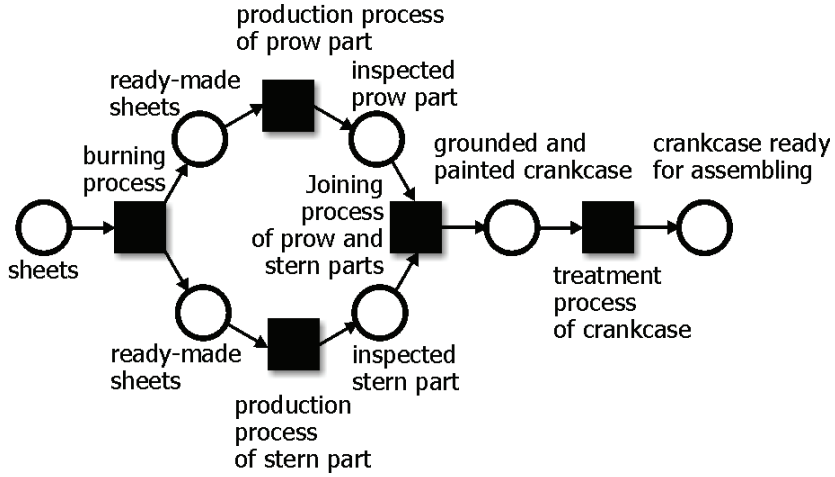



Fig. 6. Crankcase manufacturing process (source: HCP).

Figure 6 and figure 7 present models for main crankcase manufacturing and burning processes.

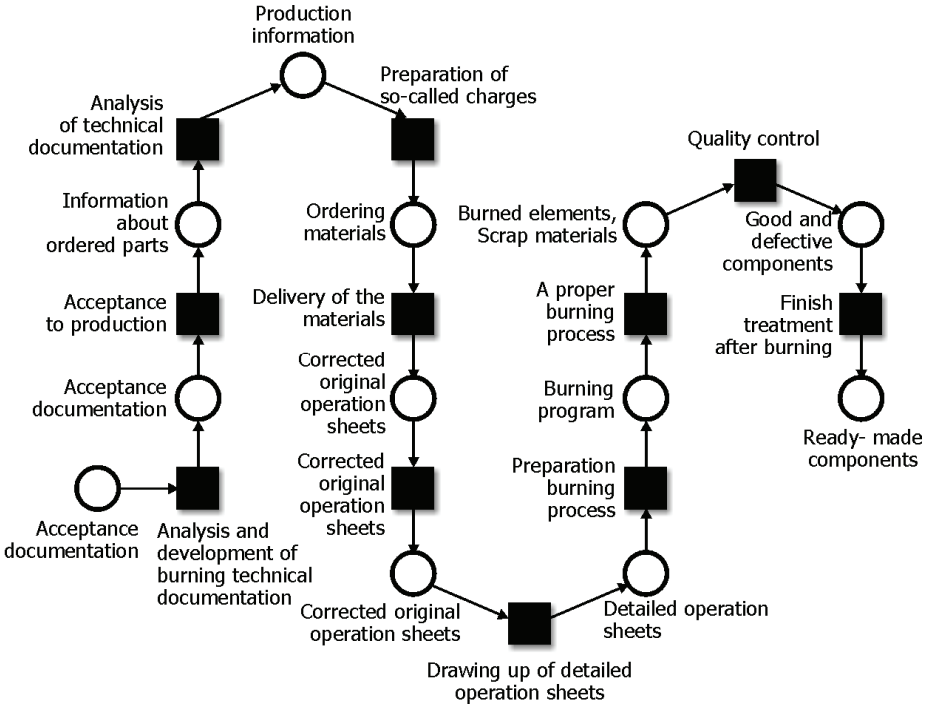



Fig. 7. Burning process (source: HCP).

The application of Petri Nets made it possible to collect valuable information about the structure of production process and provided suitable basis for the simulation. However, the obtained output was not sufficient to make a final decision about real process reengineering. Therefore, an additional analysis with the help of another reengineering methodology was required.

## 4. Discussion of approaches to reengineering

On the basis of literature study five different approaches to reengineering can be identified (Cempel, 2005), (Pacholski et al., 2009):

- M. Hammer and J. Champy approach (Hammer & Champy, 1993)
- R.L. Manganelli and M.M. Klein approach (Manganelli & Klein, 1998)
- N.M. Tichy and S. Sherman approach (Tichy & Sherman, 1993)
- T.H. Davenport approach (Davenport, 1993)
- J. Durlik approach (Durlik, 1998)

The first two approaches can be classified as of a consulting type, the third approach is purely managerial. The remaining two approaches can be classified as mainly academic (Cempel, 2005).

Hammer and Champy (Hammer & Champy, 1993). present an approach, according to which reengineering is rejecting procedures used before and looping at work needed to manufacture a product or perform a service customers require from a different point of

view. In this approach information technology is crucial, as it is a factor enabling changes. According to Hammer and Champy their approach cannot be applied in reference to business processes that have already been performed. Therefore, instead of looking for an answer "how can we use technology to perform our processes faster or better?", it is more reasonable to answer the question "what can we do with new technology – what we never tried before?". The most difficult in this method is finding new possibilities and opportunities technology gives. Such approach requires changing deductive way of thinking into induction. In practice, it is simply finding a very good solution and then searching for problems that can be solved with methodology or tool already developed. Though the authors have never defined such methodology, analyzing their work, the following steps of reengineering project can be distinct (table 2).

| Stage | Description |
|---|---|
| Introduction | Generate reasons for changes; define vision and goals and define and appoint a project team |
| Process identification | Map basic processes |
| Process selection for reengineering | Chose processes to be reengineered at first and define work teams to describe sub processes |
| Understanding of selected processes | At his stage, it is more important to understand how processes work than to analyze them in great detail, comparing processes actually are performed with their description in procedures |
| Clean state design of selected process | His stage requires creativeness, lateral thinking is used, imagination is employed, theoretical optimal processes are defined and then adjusted to fit reality |
| Implementation | Implementation of new solutions |

Table 2. Stages of radical approach – Hammer/Champy (Pacholski et al., 2009).

R.L. Manganelli i M.M. Klein in Reengineering (Manganelli & Klein, 1998). introduced step by step organization improvement procedure. Their work presents unique systematic detailed approach to reengineering idea. The authors wanted to provide a practical tool that can be used in organization.

Rapid Re methodology consists of 54 tasks integrated into five stages (table 3). Finishing each stage is a milestone of a project. Each of the stages in the methodology is illustrated with an example of unreal company ABC Toy Company Ltd. The first results usually appear after six months, up to one year after implementation. It is a consequence of demotivative influence of time-consuming projects on employees, especially those lower levels, what is more managers generally want the results to come as soon as possible. The authors called their method Rapid Re. It is to improve processes of operational level, it is not supposed to be used for tactical or strategically processes such as market choice or New product development. To make the method complete software Rapid-Re: Reengineering Software for Microsoft Windows was developed.

| Stage | Description |
|---|---|
| Preparation | At his stage goals to be achieved in reengineering process are generated by managers, scope of project is defined, schedule, risk and costs acceptable, members of reengineering team are appointed and trained |
| Identification | Customer-focused organization model is developed, strategic and value-adding processes are identified, models of processes are developed, organization and resources are mapped, processes to be reengineered (providing best results) are chosen |
| Developing a vision | Process vision is developed, developed vision is to provide radical change of effectiveness by identification of organization, systems, information flows, current problems, ratios to assess and compare effectiveness are developer, goals and improvement opportunities are identified, as well as changes necessary to achieve them |
| Solution design – technical aspect | During his stage technical aspects of changes are planned, preliminary plans are defined: procedures and systems developing, hardware, software and services purchases, technical changes, testing and modules allocation |
| Solution design – social aspect | Organization, personnel, workplaces, career and motivation system of reengineered process are described, preliminary plans of recruitment process, training, reorganization and personnel movement in organization are also defined |
| Transformation | His stage is simply pilot program and full implementation |

Table 3. Stages of Rapid Re method - Manganelli/Klein (Pacholski et al., 2009)

GE based on leadership methodology was described by N.M. Tichy i S. Sherman in: Control Your Destiny or Someone Else Will (Tichy & Sherman, 1993). The stages of the methodology are presented in the table 4. The book mentioned describes the story of Jack Welch trying (successfully) to save General Electric from falling. The results of his ideas implementation was doubling income and tripling profits, while productivity zoom by 400%. The main idea of GE methodology is revolutionary changes implementation in a continuous way. N. Tichy gives five principles, which should be used when implementing changes in organization:

- Understanding business mechanisms,
- Understanding interpersonal relations,
- Rejecting compromises when striving for golas established,
- being open to changes,
- having a hard head and a kind heart.

| Stage | Description |
|---|---|
| Awakening | At this stage awareness of changes necessity is defined, the next stage is creating urgent need for changes, technical, political and cultural barriers are diagnosed |
| Developing a Vision | During this stage motivating vision of future is created and employees are encouraged to be involved in a project |
| Design and reconstruction | At this stage creative destruction and redesign is performed, and then a new organization is built, it important to motivate people to create, after this stage changes are defined |

Table 4. Stages of metod Tichy/Sherman (Pacholski et al., 2009)

Changing an organization requires defining an idea and vision. The authors suggest three aspects of ideas definition:

- technical – it describes how the company is going to earn money in market competition conditions with resources used,
- political – it describes how power, influence and prizes can be used to stimulate organization,
- cultural – it describes how commonly respected standards and values can keep people together.

The most important technical idea of this methodology is that each company being a part of GE was „the first or the second in the world". Cultural idea was mostly on destroying limitations, political on integrating.

T.H. Davenport (Davenport, 1993) suggests that reengineering teams should focus on several (no more than fifteen) most important processes. In contradiction to radical approach Davenport suggests studying chosen processes to avoid finding old solutions as new ones. The most important is implementation of innovation because it is very important for project success. This stage generally takes longer (minimum a year) than all the other stages. The stages of Davenport methodology are presented in the table 5. It seems that not only information technology is important but employees who are making the change. New work organization should motivate them and make them focused on value adding activities and continuous search for innovation. Innovation should be not a project but continuous process.

| Stage | Description |
|---|---|
| Developing a Vision and objectives | Definition of vision and goals |
| Process Identification | Identification of processes to be redesigned |
| Understanding analysing processes | Testing processes functioning and benchmarking |
| Use of information Technologies | Analyzing opportunities of IT usage in redesigned processes |
| Creating process prototypes | Creating detailed process prototypes, personnel analyses the prototypes, develops further improvements and creates adaptation projects |
| Implementation | Implementation of prototypes tested |

Table 5. Stages of Davenport method (Pacholski et al., 2009)

Davenport also suggests combining reengineering with less revolutionary process approaches f.ex. management through quality (Total Quality Management).
In dynamic business reengineering methodology, a controlling is stressed and (Durlik, 1998) proper steps in reengineering methodology can be performed only after a strategic and economic analysis of the company. After that some decisions concerning product positioning and company's structure are made. The steps of reengineering methodology proposed by Durlik (Durlik, 1998) are presented in the table 6. For each mega process and process goals accepted by managers and executors are defined. Company's departments to be changed are chosen and process to be improved are defined. The range of changes to be made is assessed and potential effects are analyzed. The criteria of projects selection should be profit by it does not have to be defined in traditional way. The profit can be preserving costs or increasing sales potential.

| Stage | Description |
|---|---|
| Setting a Project task | At this stage goals for each process are defined, as well as criteria used to assess them. |
| Preparing a process map and setting the scope of further works | At this stage, except from creating a process map, the order of process solving, project range, executive team and budget are defined. |
| Radical re design of selected processes | At this stage general model of each subprocess, process and subprocess is defined. New solutions variants are developed and changes are designed. Organizational and management structure are adjusted to fit new processes. |
| Simulation and option assessment | At this stage detailed analysis of costs and benefits coming from implementation and use of each new process scenario is performed. The result is recommendation of a process to be implemented. |

| | |
|---|---|
| By selecting a Best option | Reengineering team chooses optimal variant by selecting options and presenting them to top managers. |
| Implementation | Based on project management methodology, includes: planning of project financing, organization of executive teams, negotiations, relations with partners, infrastructure, recruitment, training, mechanical and technological launching, controlling, implementation. |
| Controlling | Implementation of controlling to control execution and supervision on budget defined |
| Continuous improvement | Reengineering team and change manager are obliged to meet in a continous manner |

Table 6. Stages of Durlik method (Durlik, 1998)

To describe processes in organizations process maps and relation diagrams are used. They are developed for each products and for the company as a whole as well. To model processes two types are used:

- technical – including physical parameters of a process (shop floor, machines, energy, resources)
- economic – including two most important parameters – time and money.

Durlik (Durlik, 1998) describes controlling as a tool used to control execution and supervision of a budget plan. Disregarding controlling, according to the author, used to be the reason of overfilling the plans in terms of cost or organizational issues. Changes in project, based on conclusions coming from controlling, are implemented only by reengineering team. The author introduces a term 'dynamic business reengineering' (DRB), which means continuous changes with respect to reengineering principles.

The analysis of these methodologies indicates a number of elements they have in common. A cycle of organized actions defined by Le Chatelier (Cempel, 2005) (Pacholski et al., 2009) was used as the base point for the analysis. This cycle is composed of the following phases:

- Goal choice
- Research of resources and conditions for goal realization
- Resources and conditions preparation
- Goal realization
- Inspection of results

Based on this reasoning, four phases can be distinguished:

- Qualification phase
- Research and optimal solution selection phase
- Realization phase
- Inspection and evaluation chase

Table 7 presents reengineering methods according to the defined phases. The presented order indicates a concentration of activities at the initial stages of the methods. It confirms that the initial stages are the sources of success in 80% of all cases (Vilfredo Pareto principle). However, this order reveals one more problem, i.e. in most methods, the inspection and evaluation phases are not clearly distinguished – only in Durlik's methodology this phase is defined, yet without determining tools or instructions.

| | Qualification phase | Research and optimal solution selection phase | Realization phase | Inspection and evaluation phase |
|---|---|---|---|---|
| M. Hammer / J. Champy | Introduction<br>Process identification<br>Process selection for reengineering<br>Understanding of selected processes | Clean slate design of selected process | Implementation | |
| R.L. Manganelli / M.M. Klein | Preparation<br>Identification<br>Developing a vision | Solution design: technical aspect social aspect | Transformation | |
| N.M. Tichy / S. Sherman | Awakening<br>Developing a vision | Design and reconstruction | ( Implementation is part of phase 3) | |
| T.H. Davenport | Developing a vision and objectives<br>Process identification<br>Understanding and analyzing processes | Use of information technologies<br>Creating process prototypes | Implementation | |
| I. Durlik | Setting a project task<br>Preparing a process map and setting the scope of further work | Radical redesign of selected processes<br>Simulation and option assessment<br>By selecting best option | Implementation | Controlling<br>Continuous improvement |

Table 7. Reengineering methods per individual as phases of an organized activity (Pawlewski et al., 2008b)

Only Durlik's method (Durlik, 1998) shows a need to use simulation to assess individual options; however, on closer analysis, the need for simulation is only indicated, without any hints given on how to proceed with it. There is no description of simulation tools or methodology. On the basis of this analysis, conclusion is that there is a gap in reengineering methodologies since they do not account for industry-based requirements for simulation.

## 5. Additional research based on methodology Rapid Re

In the case study presented, Rapid Re method was applied due to the fact that it has been described precisely and the literature on the subject provides many examples of detailed problem-solving solutions.
Rapid Re is the methodology which was developed by R.L. Manganelli and M.M. Klein in the beginning of the 90's, as a procedure which was described in „The Reengineering Handbook" (Manganelli & Klein, 1998).
The main arguments for this selection are:

- suitability for the improvement of the operation processes, yet not for the tactical or strategic ones
- the most methodological approach - described precisely
- the literature on the subject provides many examples of detailed problem-solving solutions.

This methodology consists of five stages:

- Arrangements – this stage concerns such matters as making the board accept the project, defining purposes of the project, composing the project team, determining skills of the team members, team training, changing the plan of development
- Identification - concerns mostly processes in an organization, their connections to supplier and customer processes, process modeling, preparation of the map of the organization and sources
- Creating a vision - the stage which is an estimation of the existing processes, their influence on general effectiveness, the strategy of the change implementation and the estimation method with the use of benchmarking
- Solution project – technical aspect – the use of technical sources and technology in modifications and – social aspect – the method of human resources transformations
- Transformation – methods of work progress inspection, success estimation, pilot tests

Investigations show that according to Rapid Re methodology, the correcting procedure of a crankcase manufacturing process was elaborated.
Stage 3 – creating a vision – this stage (Manganelli & Klein, 1998) identifies the actions which create added value; these are actions owing to which something is created or appreciated by customers, actions of inspection and others. These actions were compiled in tables for each main sub-process.
Example of a Burning process is presented in Table 8. Based on the tables with classified activities, the actions ratio which generates the added value was enumerated in relation to a general number of actions.

| No | Activity | Type of activity | | |
|---|---|---|---|---|
| | | Value-adding | Inspection | Other |
| | Burning | | | |
| 1 | Acceptance to production | | | X |
| 2 | Technical documentation analysis | | X | |
| 3 | Charge preparation | | | X |
| 4 | Order of materials | | | X |
| 5 | Preparation of detailed operation | | X | |

| | | | | |
|---|---|---|---|---|
| | sheets of details | | | |
| 6 | Developing the burning programme | | X | |
| 7 | Burning process + transport | X | | |
| 8 | Inspection | | X | |
| | ..... | | | |

Table 8. Classification of the activities in a burning process- an excerpt (Pawlewski & Fertsch, 2008)

In the following steps, the factors which influence the effectiveness of the process and potential sources of errors and problems were described. Based on the information collected before, the possibility of process modification was estimated. The modification was evaluated considering the range of modification and difficulties in execution. The expected costs of the modification were assessed as well as profits generated by them. The range of advancement was evaluated as well as the risk which arises from introducing the modification. The estimation of the possibility of reengineering is presented in Table 9.

| Possibility of reengineering | Modification | Difficulty | Advantages | Risk |
|---|---|---|---|---|
| Faults elimination which occurs during order reception and technical documentation analysis | Electronic order reception current bringing up to date | Moderate | Accuracy, less work | Low |
| Fines sentencing for unpunctual orders completing | Modification in agreements signed with subcontractor | Moderate | No delays | May not succeed which results subcontractor change |
| Optimization of COBURG III utilization | Adequate time scale production preparation | High | Cost reduction of equipment operation | Well qualified production planners |
| Faults elimination which appear when appointing a date of executing actions included in the whole process | Making a proper time-scale production | High | Time reduction of crankcase production | Well qualified production planners |

| | | | | |
|---|---|---|---|---|
| Quality inspection carried out adequately early after delivery annealed crankcase to subcontractor | Quality inspector checks up the delivered crankcase | Low | No possibility of receipt the wrong annealed crankcase | Low |

Table 9. List of the possibilities of the ship crankcase production process rationalization – an excerpt (Pawlewski & Fertsch, 2008)

Accomplishment of the up-to-the-present works let to specifying the vision of the "ideal" process, i.e. describing performance of the process when all the parameters are optimal. The execution of basic actions which the process is composed of was defined in order to make them ideal.
Rapid Re methodology is appropriate mainly for business processes, that is why quite a few problems occurred when it was adjusted to reengineering of the production process of the ship crankcase. The method is very responsive to errors connected with compiling data. It is seen particularly in counting the ratio of the actions which bring added value to all actions. In the analyzed case, its high value is caused by time limitations. They resulted in compiled data based mainly on technological documentation instead of being based on direct observation. However, a compact and specified vision of the process was successfully suggested and enabled reengineering definition. It seems that further works should be directed to defining stricter requirements connected with the quality of the compiled data in order to have no doubt when calculating the factor which is the measure of the potential of redesigning the process. On the other hand, it seems to be impossible to build a formal model of the process so that it could be simulated and the results of the redesigning would be observed.

## 6. Hybrid solution based on Petri Net and Rapid Re

Investigations based on Petri Nets and Rapid Re methods presented in the previous sections has shown that none of them entirely fulfills the company requirements for the production process reengineering. The method based on Petri Nets is a suitable tool for identifying the process structure as well as an adequate framework for simulating the analyzed process before and after reengineering. Rapid Re method is not appropriate for simulation; it also lacks the possibility for time analyses of the operations and the classification of activities is not sufficient for a complex production process. The biggest advantage of Rapid Re methodology is the fact that it provides a framework for a reengineering process design and organization. Its procedure is very precisely described in the literature and many examples of detailed problem-solving solutions are given. Rapid Re provides tools and methods for making an assessment of the processes appropriateness, as well as a comparison of the activities in the process.
The research conducted in the analyzed company has shown that a hybrid solution is needed for reengineering a complex production process. The hybrid solution should combine the advantages of both methods. The Rapid Re methodology should be extended by the following elements:

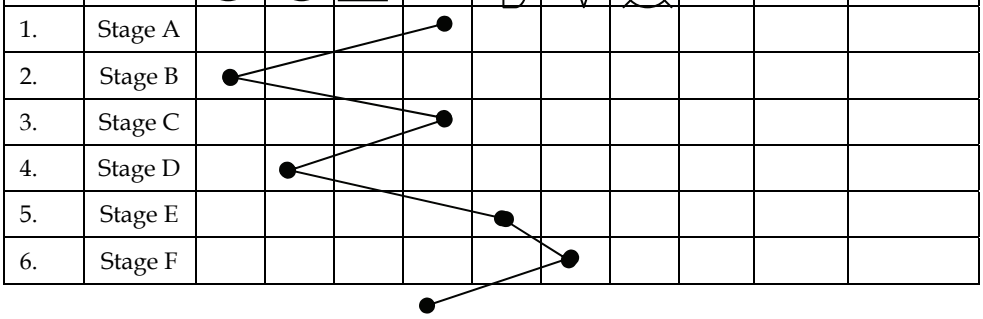
- transition of a process map into a process model based on Petri Nets in order to gain the possibility of analyses and synchronization of parallel activities.
- supplementation of activity-based indicators used in Rapid Re, by the introduction of time-based indicators
- extension of Rapid Re activities classification (value adding, inspection, other) by the classification applied in ASME methodology developed by the American Society for Mechanical Engineers (Cempel 2005) :
    - value adding operations,
    - operations which do not add any value,
    - quality and / or quantity control,
    - transport, flows of people, materials, information, documents, etc.,
    - downtime, temporary storing, delay or –– idle time between operations,
    - storing which is not downtime.

Table 10 presents symbols used in ASME methodology completed by symbol of useless work an table 11 shows the scheme of typical chart of process flow.

| Symbol | Description |
|---|---|
| | Value adding operation |
| | Operation which do not add any value |
| | Quality and / or quantity control |
| | Transport, flows of people, materials, information, documents |
| | Down time, temporary storing, delay or – idle time between operations |
| | Storing which is not down time |
| | Useless work (meetings, double operations, useless review, useless evaluatiuon) |

Table 10. Symbols used in modified ASME methodology (Pacholski et al., 2009)

| Index | Stages | V | ○ | □ | → | D | ▽ | ⊗ | Time | Person | comments |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1. | Stage A | | | | | | | | | | |
| 2. | Stage B | | | | | | | | | | |
| 3. | Stage C | | | | | | | | | | |
| 4. | Stage D | | | | | | | | | | |
| 5. | Stage E | | | | | | | | | | |
| 6. | Stage F | | | | | | | | | | |



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 7. | Stage G | | | | | | | | | | |
| 8. | Stage H | | | | | | | | | | |
| 9. | Stage I | | | | | | | | | | |
| 10. | Stage J | | | | | | | | | | |

Table 11. Scheme of chart process flow according to ASME methodology (Pacholski et al., 2009)

Simulation phase should be introduce into existing phase of Rapid Re methodology – "Solution design" according to fig.8.

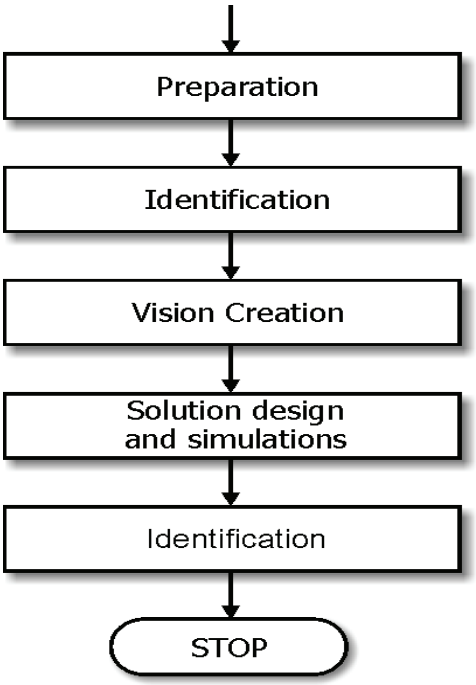



Fig. 8. Rapid Re methodology supplemented with „Simulations" phase.

The presented idea of a hybrid solution establishes the base of a new methodology which will be investigated and described. There are plans to continue this work in the collaboration with the ship engines factory where research was started.

## 7. References

van der Aalst, W.M.P. (1996) Three Good Reasons for Using a Petri-net-based Workflow Management System. In S. Navathe and T. Wakayama, editors, *Proceedings of the*

*International Working Conference on Information and Process Integration in Enterprises (IPIC'96)*, pages 179–201, Camebridge, Massachusetts

van der Aalst, W.M.P. & van Hee, K.M. (1996) Business Process Redesign: A Petri-net based approach. *Computers in Industry*, 29(1-2):15–26

van der Aalst, W.M.P.; Desel, J. & Oberweis, A. (2000) *Business Process Management*. Springer-Verlag Berlin Heidelberg

Cassandras, C.G. & Lafortune, S. (1999) *Introduction to Discrete Event Sistems*. Kluwer Academic Publishers, MA

Cempel, W.A. (2005) *Metodologia reengineering w przedsiebiorstwach przemysłu maszynowego. [Reengineering methodology in engineering industry.]* Doctoral thesis, Poznan University of Technology, Poznan, Poland

Charbonnier, F.; Alla, H. & David, R. (1999) The supervised control of discrete-event dynamic systems. *IEEE Transactions on Control Systems Technology*, 7(2):175–187

Chen, S.M. (1990) Knowledge representation using fuzzy Petri nets. I*EEE Trans. on Knowledge and Data Engineering*

Davenport, T. H. (1993) *Process innovation: Reengineering work through information technology.* Boston, MA: Harvard Business School Press

Durlik, I. (1998) *Restructuring business processes: Reengineering theory and practice. Business process reengineering in High-Technology. [Restrukturyzacja procesów gospodarczych: Reengineering teoria i praktyka. Businesss Process Reengineering w warunkach High-Technology.]* Warsaw, Poland, Placet

Endsley, E.W. & Tilbury, D.M. (2004) Modular verification of modular finite state machines. *In Proceedings of the 43rd IEEE Conference on Decisionand Control*, volume 1-5, Nassau, Bahamas

Galan, J.C.; Marcos, M.; Reif, W.; Balser M. & Schmitt J. (2005) *New model of guideline process. Addendum. Specific Targeted Research Project Information Society Technology*, Universitat Jaume I, September 8, IST-FP6-508794

Genc, S. & Lafortune, S. (2003) Distributed diagnosis of discrete-event systems using petri nets. *In Proceedings of the 24th International Conference on Applications and Theory of Peti Nets*, pages 316–336, Oulu, Finland

Gollapudi, C. & Tilbury, D.M. (2001) Logic control design and implementation for achining line tested using Petri nets. *In Proceedings of the ASME-IMECE Dynamic Systems and Control Division*

Gruer, P.; Koukam, A. & Mazigh, B. (1998) Modeling and quantitative analysis of DES: A statecharts based approach. *Simulation Practice and Theory*, 6:397–411

Hammer, M. & Champy, J. (1993) *Reengineering the corporation*. New York, NY: Harper Business

Harel, D. & Naamad, A. (1996) The statemate semantics of statecharts. *ACM Transaction on Software Engineering and Methodology*, 5(4):239–333

Harel, D.; Pnueli, A.; Schmidt,J.P. & Sherman, R. (1987) On the formal semantics of statecharts. *In Proccedings of the Second Annual Symposium on Logic in Computer Science,* pages 54–64, Ithaca, NY, June, 22-25. Computer Society Press

Jansen-Vullers, H. & Netjes, M. (2006) Business Process Simulation - A Tool Survey. *Seventh Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools,* University of Aarhus, Danmark, October 24-26, ISSN 0105-8517

Johnson, D. (2002) Nano devices lead assault on traditional PLC applications. *Control Engineering*, 49(8):43–44, August

Kumar, R. & Garg, V.K. (1995) *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Publishers, Norwell, MA

Lewis, R.W. (2001) *Modeling control systems using IEC 61499*. The Institution of Electrical Engineers

Lucas, P.; Hommerson, A.; Galan, J.C.; Marcos, M.; Coltell, O.; Mouzon, O.; Polo, C.; Rosenbrand, K.; Wittenberg, J. & van Croonenborg, J. (2005) New model of guideline process. *Specific Targeted Research Project Information Society Technology,* Universitat Jaume I, March 3, IST-FP6-508794

Manganelli, R. L. & Klein, M. M. (1998) *The Reengineering Handbook*. Warsaw, Poland: PWE (polish edition)

Pacholski, L.; Cempel, W. & Pawlewski, P. (2009) *Reengineering – reformowanie procesów biznesowych i produkcyjnych w przedsiebiorstwie. [Reengineering: Reforming business and production processes in company.]* Poznan, Poland: Publishing House of Poznan University of Technology

Park, E.; Tilbury, D.M. &, Khargonekar, P.P. (1999) Modular logic controller for machining systems: Formal representation and performance analysis using Petri nets. *IEEE Transactions on Robotics and Automation*, 15(6):1046–1061

Pawlewski, P. & Fertsch, M. (2008) Using Petri Net To Model Production Process Of Ships Engine Crankcase In Process Reengineering (case study). *Proceedings of 7e Conférence Internationale de MOdélisation et SIMulation* - MOSIM'08 – 31.03-2.04.2008 – Paris- France

Pawlewski P.; Trujillo J.A.; Golinska P.; Pasek Z.J. & Fertsch M. (2008) Process oriented approach versus description of technological routes - their role in production management, *Proceedings of the 18th International Conference on Flexible Automation And Intelligent Manufacturing* FAIM 2008 30.06-2.07.2008 Skovde Sweden

Pawlewski, P.; Golinska, P.; Fertsch, M.; Trujillo, J. & Pasek, Z. (2008) Supportive Role Of The Simulation In The process Of Ship Engine Crankcase Production Process of Reengineering (Case Study). *Proceedings of the 2008 Winter Simulation Conference* S. J. Mason, R. Hill, L. Moench, and O. Rose, eds.

Peterson, J.L. (1981) *Petri Net Theory and the Modeling of Systems*. Prentice-Hall

Processing Standard Publication 183 (1993) *Announcing the Standard for INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0),* Draft Federal Information

Ramadge, P.J.G. & Wonham, W.M. (1987) Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, pages 206–230, January

Ramadge, P.J.G. & Wonham, W.M. (1989) The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, January 1989.

Romero, F. & Agost, M.J. (2008) Activity modeling in a collaborative ceramic tile design chain: an enhanced IDEF0 approach, *Res Eng Design*, Springer-Verlag, 19, pp.1-20

Tichy, N. M. & Sherman, S. (1993) *Control your destiny or someone else will.* New York: Doubleday

Trujillo, J.M. (2004) *Position Machine for Logic Control: Composition of Patterns in Reconfigurable Manufacturing System*. PhD thesis, University of Valladolid, Spain

U.S.Air Force. (1981) *Integrated Computer Aided Manufacturing (ICAM) Architecture Part II, Volume IV - Functional Modeling Manual (IDEF0),* Air Force Materials Laboratory,Wriht-Patterson, AFB, Ohio 45433, AFWAL-tr-81-4023

Zakarian, A. & Kusiak, A. (2001) Process analysis and re engineering, *Computers & Industrial Engineering* 41 pp.135-150

22

# Workflow Diagnosis Using Petri Net Charts

Calin Ciufudean and Constantin Filote
*Stefan cel Mare University of Suceava*
*Romania*

## 1. Introduction

Workflow diagnosis is a crucial and challenging task in the automatic control of complex discrete event systems, e.g. in flexible manufacturing systems (FMSs) as a representative class of discrete event systems (DESs). Our work is focused on controlling the workflows modelled with stochastic Petri nets (SPNs). This goal is achieved by building a new model for Artificial Social Systems (ASSs), and by introducing equivalent transfer functions for SPNs.
ASSs exist in practically every multi-agent system, and play a major role in the performance and effectiveness of the agents. This is the reason why we introduce a more suggestive model for ASSs. To model these systems, a class of Petri nets is adopted, and briefly introduced in the paper. This class allows representing the flow of physical resources and provides analytical approach for the availability evaluation of cellular manufacturing systems, as basic components of flexible manufacturing systems.
An Artificial Social System (ASS) is a set of restrictions on agents' behaviour in a multi-agent environment (Murata, 1989). ASS allows agents to coexist in a shared environment and pursue their respective goals in the presence of other agents. A multi-agent system consists of several agents, where at given point, each agent is in one of several states. In each of its states, an agent can perform several actions. The actions an agent performs at a given point may affect the way that the state of this agent and the state of other agents will change. A system of dependent automata consists of two or more agents, each of which may be in one of a finite number of different local states. We denote the set of local states of an agent *i* by $P_i$. The set ($P_1$, $P_2$, ..., $P_n$) of states of the different agents is called system's configuration. The set of possible actions an agent *i* can perform is a function of the local state. For every state $p \in P_i$ there is a set $A_i(p)$ of actions that *i* can perform when in local state p. The row actions ($a_1$, ..., $a_n$) denote the actions the different agents perform at a given point and is called their joint action there. An agent's next state is a function of the system's current configuration and the joint action performed by the agents. A goal for an agent is identified with one of its states. That is the reason why an agent has plans how to attain its goal.
A plan for agent i in a dependent automata is a function U(p) that associates with every state p of agent *i* a particular action $a \in A_i(p)$. A plan (Baccelli & Liu, 1992) is said to guarantee the attainment of a particular goal starting from an initial state, in a given dependent automata system, if by following this plan the agent will attain the goal, regardless of what the other agent will do, and what are the initial states of the other agents. A dependent automata system is said to be social if, for every initial state $p_o$ and goal state $p_g$, it is

computationally feasible for an agent to devise, on-line, an efficient plan that guarantees to attain the goal $p_g$ state when starting in the initial state $p_0$. For a proper behavior, a dependent automata system is modeled with a social law. Formally, a social law Q for a given dependent automata system consists of functions ($A`_1$, $A`_2$, ..., $A`_N$), satisfying $A`_1(p) \subset A`_i(p)$ for every agent i and state $p \in P_i$. Intuitively, a social law will restrict the set of actions an agent is "allowed" to perform at any given state. Given a dependent automata system S and a social law Q for S, if we replace the functions $A_i$ of S by the restricted functions $A`_i$, we obtain new dependent automata system. We denote this new system by $S^Q$. In $S^Q$ the agents can behave only in a manner compatible with the social law S (Haas & Shendler, 1991). In controlling the actions, or strategies, available to an agent, the social law plays a dual role. By reducing the set of strategies available to a given agent, the social system may limit the number of goals the agent is able to attain. By restricting the behaviors of the other agents, however, the social system may make it possible for the agent to attain more goals and in some cases these goals will be attainable using more efficient plans than in the absence of the social system. A semantic definition of artificial social systems gives us the ability to reason about such systems. For example, the manufacturer of the agents (e.g., robots) that are to function in the social system will need to reason about whether its creation will indeed be equipped with the hardware and the software necessary to follow the rules. In order to be able to reason properly, we need a mathematical model and a description language (Lee et al., 1999). We chose the stochastic Petri nets model in order to model and simulate real conditions encountered in constructions workflow planning. We shall name on further accounts this model as Stochastic Artificial Social System. Petri nets have been recognized as a powerful tool for modeling discrete event systems. State explosion, a typical problem for SPNs, is solved here by introducing the equivalent transfer functions for transitions of SPNs. Data networks, viewed as discrete systems, are analyzed with such models. In the Petri nets theory, mathematical tools are available for analysis of the qualitative properties including deadlock-freeness, boundedness, reversibility, s.a. (Haas & Shendler, 1991). However, simulation remains the effective for performance evaluation. Perturbation (e.g., delays in supply with raw materials, derangements of equipments, etc.) analysis has been developed for evaluating sensitivity measures by using simulations (Fu & Hu, 1992). A generalized semi-Markov process (GSMP) is the usual model for the stochastic processes of discrete-event simulations, and most existing perturbation analysis methods are based on the GSMP framework. Since GSMPs and stochastic Petri nets (SPNs) have been proven to have the same modeling power (Archetti et al., 1993), existing perturbation analysis methods are expected to apply to SPNs. Petri nets models considered here are SPNs with random transition firing times and the sensitivity estimators can be obtained from a simulation run. Our perturbation analysis is based on work of [5] and [6] which provides unbiased gradient estimators for a broad class of GSMPs. In this study, unbiased estimators are applied by using an appropriate SPN representation. Under correct conditioning, the unbiased estimators are easily confirmed by the simulation run of the SPN representation. This confirms the importance of underlying stochastic process. Practical solutions are shown in the paper, in order to give a concrete utilization of the theoretical model realized with SPN. The remainder of this paper is organized as follows. Section 2 presets SPNs under consideration, section 3 gives an approach for diagnosis performed with SPN, and section 4 introduces the analytic support for this diagnosis, respectively the Markov chain diagnose, section 5 presents some basic equivalent transfer function used for simplifying the

complexity of SPNs, section 6 presents unbiased estimators for general stochastic Petri nets, section 7 applies the theoretical approach to a queuing network, respectively to a construction system perturbation analysis, and explicates some practical correlations between theory and practical implementation, and conclusions underline the approaches presented in this paper and establish future work.

## 2. Stochastic Petri Nets

In an ordinary Petri net PN = (P, T, F, $M_0$), where P and T are two disjointed sets of nodes named, respectively, places and transitions. $F \subseteq (PXT) \cup (TXP)$ is a set of directed arcs. $M_0$: P $\rightarrow$ N is the initial marking. Two transitions $t_i$ and $t_j$ are said to be in conflict if they have at least one common input place. A transition t is said to be conflict free if it is not in conflict with any other transition. A transition may fire if it is enabled. A transition $t \in T$ is said to be enable at marking M if for all $p \in {}^*t$, $M(p) \geq 1$. The SPNs considered here are ordinary Petri nets with timed transitions. Timed transitions can be in conflict therefore we say that a marking is stable if no conflict transitions are enabled. In the following we assume that the initial marking is a stable marking. We note by (M, T) a stable marking reachable from M by firing t. The new stable marking M* is obtained from M according to some routing probability. The basic idea is that in order to guarantee that a stable marking can be reached; we must ensure that the respective circuit contains at least one timed transition. A SPN can be defined by the following elements (Lee et al., 1999):

$T_t$ Set of timed transitions

$M_s(M,t)$ Set of stable markings reachable from M by firing transition t

p(M*, M,t) Probability of reaching a stable marking M* from M when t fires. Obviously, we have: p(M*, M, t) = 0 if $M^* \notin Ms(M, t)$.

$F_t(.)$ Distribution function of the firing time of t

The GSMP representation of the SPN can be characterized by the following parameters:

X(t,k) Independent random variables, where $t \in T_t$ , and $k \in N$. Each X(t, k) has distribution $F_t$ and corresponds to the time of the $k^{th}$ firing of transition t.

U(t,k) Random variables on [0, 1]. Each U(t, k) corresponds to the routing indicator at the $k^{th}$ completion of t.

$r_n(t)$ Remaining firing time of transition t at $S_n$

S(t,k) Independent uniform random variables in [0, 1] range, where $t \in T_t$, $k \in N$. Each U(t, k) corresponds to the routing indicator at the $k^{th}$ completion of t.

$t_n$ $n^{th}$ completed timed transition

$M_n$ Stable marking reached at the firing of $t_n$

$S_n$ Completion time of $t_n$

$\tau_n$ Holding time of marking $M_{n-1}$

V(t,n) Number of instances of t among $t_1$ , …, $t_n$.

The dynamic behaviour of an SPN can be explained in the following way: at the initial marking $M_0$, set $r_n(t) = X(t, 1)$, $\forall\ t \in T_t(M_0)$ and set $V(t,0) = 0$, $\forall\ t \in T_t$. All other parameters $t_{n+1}$, $\tau_{n+1}$, $s_{n+1}$, $V(t, n+1)$, $M_{n+1}$, $r_{n+1}$ can be determined recursively as usually done in discrete event simulation. Recursive equations are given in (Chiang et al., 2000). The following routing mechanism is used in GSMP:

$$M_{n+1} = \varnothing(M_n, t_{n+1}, U(t_{n+1}, V(t_{n+1}, n+1))) \quad (1)$$

Where ∅ is a mapping such that P(∅(M, t, U) = M*) = P(M*, M, t).
The flexible manufacturing system to be diagnosed is modelled as a finite state machine of DES's formalism:

$$W = (s, e, T, m_0) \tag{2}$$

Where S is the state space, E is the set of events, t is the partial transition function and $m_0$ is the initial state of system. The model W accounts for the normal and failed behaviour of the system. Let $E_f \leq E$ denote the set of failure events which are to be diagnosed. Our objective is to identify the occurrence of the failure events. Therefore we partition the set of failure events into disjoint sets corresponding to different failure types:

$$E_f = E_{f1} \cup E_{f2} \cup \ldots \cup E_{fn} \tag{3}$$

This partition is motivated by the following considerations (Xie, 1998):
Inadequate instrumentation may render it impossible to diagnose uniquely every possible fault;
It may not be required to identify uniquely the occurrence of every failure event. We may simply be interested in knowing whether failure event has happened as the effect of the same failures in the system.
So, when we say that "a failure of type $F_i$ has occurred", we mean that some event from the set $E_{fi}$ has occurred.
In (Lane & Bradley, 1992) the diagnosability is defined as follows: A prefix-closed and live language L is said to be I-diagnosable with respect to the projection P, the partition $E_f$, and the indicator I if the following holds:

$$\left(\forall i \in E_f\right)\cdot\left(\exists n \in N\right)\cdot\left(\forall s \in E_{f_i}\right)\cdot\left(\forall t \in \frac{L}{s} : st \in I\left(E_{f_i}\right), \left\|t\right\| \geqslant n; \quad D\right) \tag{4}$$

Where the diagnosability condition D is:

$$\omega \in P_L\left[P(st)\right] \quad E_{f_i} \in \omega \tag{5}$$

Note that I ($E_{fi}$) denotes the set of all traces of L that end in an event from the set $E_{fi}$. The behaviour of the system is described by the prefix-closed live language L (A) generated by A (see relation (1)). L is a subset of E*, where E* denotes the keen closure of the set E (Lee et al., 1999). ||s|| denotes the length of trace s∈E. L/s denote the post language of L after s, i.e.

$$\frac{L}{s} = \left\{t \in E^* / st \in L\right\} \tag{6}$$

We define the projection P: E*→ E in the usual manner (Carmen et al., 1991):

$$P(\varepsilon) = \varepsilon, \text{ and } P(s_1, s_2) = P(s_1)\cdot P(s_2), \quad s_1 \in E^* \text{ and } s_2 \in E \tag{7}$$

Where $\varepsilon$ denotes the empty trace.
The above definition, e.g. relations (4) and (5), means the following: Let s be any trace generated by the system that ends in a failure event from the set $E_{fi}$, and let t be any

sufficiently long continuation of s. Condition D then requires that every trace belonging to the language that produces the same record of observable events, and in which the failure event is followed by certain indicator, should contain a failure event from the set Efi. This implies that on some continuation of s one can detect the occurrence of a failure of the type Fi with a finite delay, specifically in at most ni transitions of the system after s. To summarize, here diagnosability requires detection of failures only after the occurrence of an indicator event corresponding to the failure. In this paper we improve this approach by according a gradual importance of failure indicators, in correspondence with the availability of the system. In our assumption the diagnoser is a SCPN where the places are marked with the availability of the correspondent production cell. The availability of a production cell is calculated with a Markov chain, where the transitions reflect the gradual importance of the failures in the cell. We may say that the diagnoser is an extended observer where we append to every state estimate a label. The labels attached to the state estimates carry failure information and failures are diagnosed by checking these labels. We also assume the system W is normal at start.

## 3. The Petri Net Diagnoser of a FMS

In our work we assumed that when a device, sensor, transducer or any other hardware component of the FMS fails, the system reconfiguration (after repairing it) is often less than perfect. The notion of imperfection is called imperfect coverage, and it is defined as probability c that the system successfully reconfigures given that component fault occurs. The imperfect repair of a component implies that when the repair of the failed component is completed it is not "as good as new". A dependability model for diagnosability of flexible manufacturing systems is presented. The meaning of dependability here is twofold:

- System diagnosability and availability
- Dependence of the performance of the FMS on the performance of its individual physical subsystems and components.

The model considers the task-based availability of a FMS, where the system is considered operational as long as its task requirements are satisfied; respectively the system throughput exceeds a given lower bound. We model the FMS with SCPN. We decompose the FMS in productions cells.

In our assumption the availability of a cell $j$ ($j$ = 1, 2, ..., n, where n is the total number of part type cells in the FMS) is calculated with a Markov chain which includes the failure rates, repair rates, and coverability of the respective devices in the production cell $i$. The colour domains of transitions that load cell $i$ include colours that result in a value between 0 and 1, and the biggest value designates the cell (respectively the place in the SCPN model) which ensures the liveness of the net, respectively which will validate and burn its output transition. We assume that the reader is familiar with Petri nets theory and their applications to manufacturing systems or we refer the reader to (Murata, 1989). Each part entering the system is represented by a token. The colour of the token associated with a part has two components (Xie, 1998). The first component is the part identification number and the second component represents the set of possible next operations determined by the process plan of the part. It is the second component that is recognized by the stochastic colours Petri net model, and the first component is used for part tracking and reference purposes. Let $B_i$ be a (1 x m) binary vector representing all the operations needed for the

complete processing of part type *i*. Let $E_i$ be a (m x m) matrix representing the precedence relations among the operations of part type i, where m is the number of operations that are performed in the respective cell *j* (*j*=1.2…., n). For a part to be processed in the cell *j* it requires at least one operation that can be performed in the cell, that implies $B_j > 0$. Also, for a part type, where there is no precedent relationship between required operations, $E_i$ is a matrix of zeros. For a part with identification x and part type *y*, the initial colour of the corresponding token is:

$$V_{yx} = \left[ yx, B_y - \left( B_y \cdot E_y \right) \right] \tag{8}$$

Where $\left( B_y \cdot E_y \right)$ is a matrix of multiplication.

For example consider the process plan of part type $L_1$ and $L_2$ shown in Fig.1

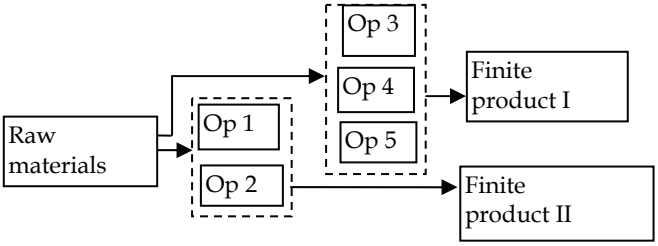



Fig. 1. Process plan of part type $L_1$ and $L_2$

Our process plan first requires operation op 1 and then operation op 2 for complete processing. We assume that our FMS can complete 5 different types of operations (e.g., for simplicity we consider only 5 different types of operations). For part type $L_1$, we have: $B_{L1}$ = [00011].

$$E_{L_1} = \begin{array}{c} \\ op\,5 \\ op\,4 \\ op\,3 \\ op\,2 \\ op\,1 \end{array} \begin{array}{c} \begin{array}{ccccc} op\,5 & op\,4 & op\,3 & op\,2 & op\,1 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & A_2 \\ 0 & 0 & 0 & A_1 & 0 \end{pmatrix} \end{array}$$

Where $A_1$ is the availability of production cell 1 (which performs operation 1), and $A_2$ represents the availability of production cell 2 at time *t*. The availability $A_i$ of cell *i* is calculated, as shown below, with Markov chains (Brehends, 2000). We notice that $A_i$ is re-evaluated at each major change in the process plan of FMS (such as occurrence of events: damages of hardware equipments, changes of process plan, etc).

Assuming that $A_1 > A_2$, then we assign to $A_1$ value 1 and to $A_2$ value 0, so that applying relation (7), the initial color of the token corresponding to a part that belongs to part type $L_1$ with identification mark 1, would be $V_{L1.1}$ = ($L_{1.1}$, 00001). Note that the information carried by the color of the tokens in the SCPN indicates the next operation to be performed by the FMS. Generally, we may say that V is the set of colors that represent all the possible

combinations of operations that can be performed in the FMS. Each member of the set V is a vector with m components, where m is the maximum number of operations to be performed in the cells of the FMS. For example, in a FMS with 5 operations to be performed, we may have V = {00000, 00001, …, 11111}. For simplicity, we assume that operations in FMS are maped to places in the SCPN model, places which are labeled with the operation identification number. The requirement for a production cell *j* (*j* = 1, …, n) which have $N_i$ (i = 1, …, m) devices of type i, is that at least $k_i$ of these devices must be operational for the FMS to be operational. To determine the system availability which includes imperfect coverage and repair, a failure state due to imperfect coverage and repair was introduced (Ciufudean & Popescu, 2004). To explain the impact of imperfect coverage, we consider the system given in Fig.2 which includes two identical manufacturing devices $M_1$ and $M_2$.

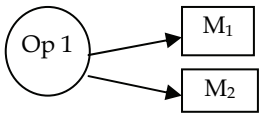



Fig. 2. Example of operation performed by two identical devices

If the coverage of the system is perfect, i.e. *c* = 1, then operation op 1 is performed as long as one of the devices is operational. If the coverage is imperfect, then operation op 1 fails with probability *1-c*, if one of the devices $M_1$ or $M_2$ fails.
We may say that, if operation op 1 has been scheduled on device $M_1$ that has failed, then the system in Fig.2 fails with probability 1 - *c*. The Markov chain for manufacturing cell *i* is shown in Fig.3. In Fig.3 the parameters $\lambda$, $\mu$, *c, r* denote respectively the failure rate, repair rate, coverage factor and the successful failure repair rate of devices in the cell.
The first part of the horizontal transition rate with the term 1 - *c* represents the failure due to imperfect coverage of an alternative equipment.
The second part, with the term 1 - *r* represents imprecise repair of the devices.
The vertical transitions reflect the failure and repair of the equipments. We assume that only one device fails at a time, in a certain operation cell.

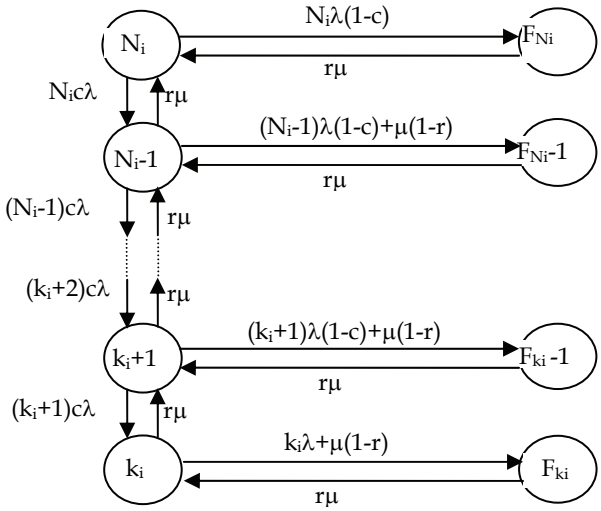



Fig. 3. Markov model for cell i

At state $N_i$ cell *i* is functioning with all $N_i$ devices operational. At state $k_i$ there are only $k_i$ devices oparational. The state of cell *i* changes from working state $w_i$, for $k_i \le w_i N_i$, where $w_i$ is the number of operational devices at a certain moment, to failed state $F_i$, either due to imperfect coverage (1 - *c*) or due to imperfect repair (1 - *r*). If the fault coverage of the system and repair of the components are perfect, the Markov chain in Fig.3 reduces to one-dimension model. The solution of the Markov chain model given in Fig.3 is a probability that at least $k_i$ devices are working at time *t*.
The availability of cell *i* is given by the next relation (Ciufudean & Larionescu, 2002):

$$A_i(t) = \sum_{j=k_i}^{N_i} P_{kj}(t), \quad \text{for } i = 1, 2, ..., n \tag{9}$$

Where $A_i(t)$ = the availability of cell *i* at moment *t*;
$P_{ki}(t)$ = probability of $k_i$ devices being operational in cell *i* at time *t*;
$N_i$ = total number of devices of type *j* in cell *i*;
$K_i$ = required minimum number of operational devices in cell *i*.
After a Markov chain for each cell of the measuring system is constructed and desired probabilities $A_i(t)$, *i* = 1,2,…,n corresponding to each manufacturing cell are determined, the stochastic colored Petri net (SCPN) can be initialized and the simulation process of the flexible manufacturing system (FMS) begins. The status of this graph (e.g., the SCPN) at different moments $t_k$, gives us the diagnosis of the FMS.

## 4. Constructing the Markov Chain Diagnoser of a FMS

Starting from the PN modular structure described in the previous section we shall build a Markov chain diagnoser structure. Let A denote the set of alphabets. A trace over A is a finite sequence of alphabets and is synonymously for Petri net complete sequence of firing transitions (Yee & Ventura, 2000). Given a trace $a \in A$, $|a|$ denotes the length of the trace. Given a trace $a$ and a positive integer $i \leq |a|$, $a_i$ and $a[i]$ denotes the prefix consisting of the first $i$ alphabets, and the $i$-th symbol respectively. The concatenation of two traces $a_1$ and $a_2$ is denoted by $a_1 \cdot a_2$. B={0,1} is the binary alphabet. A diagnoser over the alphabet set A is a function f: A → B. One may read 0 as "good" and 1 as "bad" (Recalde & Silva, 1998). In the context of diagnosis, a 1 placed after reading a trace should be interpreted as an alarm (something anomalous is happening); on the other hand, 0 indicates normal behavior. Diagnosis is a real-time activity, due to Petri nets simulation, where alarms have to be available also in real time. We assume that the set of traces, denoted by T, is defined as a finite table of patterns. We also assume that each state of a trace has an associated identifier; then the set T maps an identifier of a final state to 1 and the rest of the identifiers are mapped to 0. We will use the set T to construct a Markov chain. A transition in the Markov chain is a pair of states, and a state is associated with a trace of length Λ over the alphabet A. The operation shift ($a$, $x$) shifts the trace $a$ left and appends the alphabet $x$ at the end, e.g. shift (*aba, c*) is equal to *bac*. The initial state of the Markov chain, which corresponds to the initial marking in the modular Petri net controller, is associated with a trace of length $|a|$ consisting of all null symbols {0}, e.g. if $|a|=3$, the initial state is associated with the trace [0, 0, 0]. The operation next ($a$) returns the first symbol of the trace α and left shifts $a$ by one position, e.g. next (*abc*) returns $a$ and updates the trace to *bc*. For each trace $a \in T$, we execute the following steps until all the alphabets of $a$ are scanned:

1. Let $c$ = next($a$);
2. Set next-state to shift (current-state, $c$);
3. Increment the counter for the state current-state and transition (current-state, next-state);
4. Update current-state to be next-state;
5. Go to Step 1 if $c \neq 0$, else Stop.

After all the traces in the set T have been processed, each state and transition has a positive integer associated. The probability of transition ($S_i$, $S_j$) is $N(S_i, S_j)/N(S_i)$, where the $N(S_i, S_j)$ and $N(S_i)$ are the counters associated with transition ($S_i$, $S_j$) and state S, respectively. We may say that probability of a transition is the ratio of the frequency of the transition and the frequency of its source (Zuo et al., 2000).

We build a connected graph, where each edge is labeled with the probability of transition between two adjacent vertices that correspond to the states of the associated Markov chain. The Markov chain has a uniform equilibrium distribution; the rate of convergence to this distribution is determined by the largest eigenvalue of the transition matrix (Hopkins, 2002), e.g. we deal with the problem of finding the fastest mixing Markov chain on the graph.

We consider a connected graph $G = (V, E)$ with vertex set $V = [1, ..., n]$ and the set of edges $E \subseteq V \times V$, with $(i, j) \in E$ $(j, i) \in E$. We assume that there are self-loops that connect an edge with itself: $(i, i) \in E$. We define a discrete-event Markov chain on the vertices of the graph as follows. The state at a certain amount of time $t$ will be denoted $X(t) \in V$, for $t$ = 0, 1, …, n, where $n \in N$. Each edge in the graph is associated with a transition in the Petri net model given in the section 1. Firing a transition it evolves a probability assessment so that it

makes a transition between the two adjacent vertices. These edges probabilities must be nonnegative, and the sum of the probabilities of edges connected to each vertex, including the self logs, must be equal to 1. The probability associated with self-logs (*i, i*) is the probability that X(t) stays at vertex *i*. We describe the Markov chain through its transition probability matrix $P \in R^{nxn}$, where:

$$p_{ij} = \text{prob}(x(t+1) = j / x(t) = 1) \quad i, j = 1, ..., n \tag{10}$$

This matrix must satisfy the following condition:

$$P \geq 0, \; P \cdot \mathbf{1} = \mathbf{1}, \; P = P^T \tag{11}$$

Where the inequality $P \geqslant 0$ means element-wise, i.e. $P_{ij} \geqslant 0$ for *i, j*=1,..,n, and **1** denotes the vector of all ones.
We note that condition (2) means that P must be symmetric and doubly stochastic, and:

$$p_{ij} = 0 \text{ , } (i, j) \neq E \tag{12}$$

Relation (3) states that transitions are allowed only between vertices that are linked by an edge. Let $\Pi(t)^T \in R^n$ be the probability distribution of the state at time *t*: $\pi_i(t) = \text{prob}(X(t) = i)$ . The state distribution satisfies the following recursion relation:

$$\Pi(t)^T = \Pi(0)^T \cdot P^T \tag{13}$$

If P is symmetric and $P \cdot \mathbf{1} = \mathbf{1}$, then $\mathbf{1}^T \cdot P = \mathbf{1}^T$, so the uniform distribution $(1/n) \cdot \mathbf{1}$ is an equilibrium distribution for the Markov chain. If the chain is irreducible and non-periodic, then the distribution $\Pi(t)$ converges to the unique equilibrium distribution $(1/n) \cdot \mathbf{1}$ as *t* increases (Karlin & McGregor, 1959). A discrete-time Markov chain is said to be skip-free if the one-tep transition probabilities satisfy $p_{ij} = 0$ for $j \leq i$-2 and $j \geq i$+2. One may call a discrete-time, skip-free Markov chain a random walk (Schrijner, 1995). We denote by:

$$P = \left(p_{ij}\right) = \begin{bmatrix} p_{00} & p_{01} & 0 & 0 & 0 & . & . & . \\ p_{01} & p_{11} & p_{12} & 0 & 0 & . & . & . \\ 0 & p_{21} & p_{22} & p_{23} & 0 & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \end{bmatrix} \tag{14}$$

The matrix of one-step transition probabilities that is stochastic and the random walk X is said to have a reflecting boundary 0 if $p_{ii+1} = 0$ and P is sub-stochastic and the random walk has an absorbing state which can be reached though state 0 only if $p_{ii+1} > 0$. In relation (5) we

have: $p_{ii-1} + p_{ii} + p_{ii+1} = 1$. The state space S constitutes an irreducible class which is non-periodic if $p_{ii} > 0$ for some *i* and periodic with period 2 if $p_{ii} = 0$ for all *i*. Accordingly, X will be called periodic if $p_{ii} > 0$ for some *i* and periodic if $p_{ii} = 0$ for all *i*. One may define:

$$\pi_0 = 1;\ \pi_i = \frac{p_{01} \cdot p_{12} \cdots p_{i-1i}}{p_{10} \cdot p_{21} \cdots p_{ii-1}},\quad i \geqslant 1 \tag{15}$$

And noting $\sum_i \left[\pi_i + \left(p_{ii+1} \cdot \pi_i\right)^{-1}\right] = \infty$, the following theorem is true (Schrijner, 1995):

*Theorem:* If $p_{00} = 0$ then the random walk X is recurrent if $\sum_{i=0}^{\infty} p_i \cdot \pi_i = \infty$, and nonnegative recurrent if $\sum_{i=0}^{\infty} p_i \cdot \pi_i \leq \infty$.

### 4.1 Illustrative example

Consider a product manufactured in two steps, in the FMS given in Fig.1, respectively the part type $L_1$ (Ciufudean et al., 2005). We assume that in matrix $E_{L1}$, we have $A_1 > A_2$, and therefore operation op 1 is performed the first. Two types of machines, $M_i$, *i*=1, 2, …, 5 are used to perform the first operation, and $U_i$, *i* = 1, 2, 3 are used to perform the second operation. The structures of the production cells op 1 and op 2 are given in Fig.4.a and 4.b, respectively. It is required for cell op 1 that at least three machines are operational and for cell op 2 at least two machines must be operational, in order to achieve the desired throughput of port type $L_1$ of FMS.

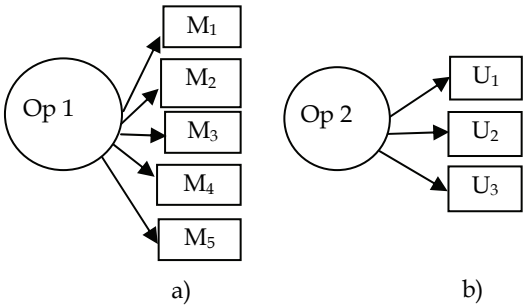



Fig. 4. Structures of production cells of part type $L_1$ FMS from Fig.1

The SCPN model of the part type $L_1$ of the FMS given in Fig.1 is represented in Fig.5.

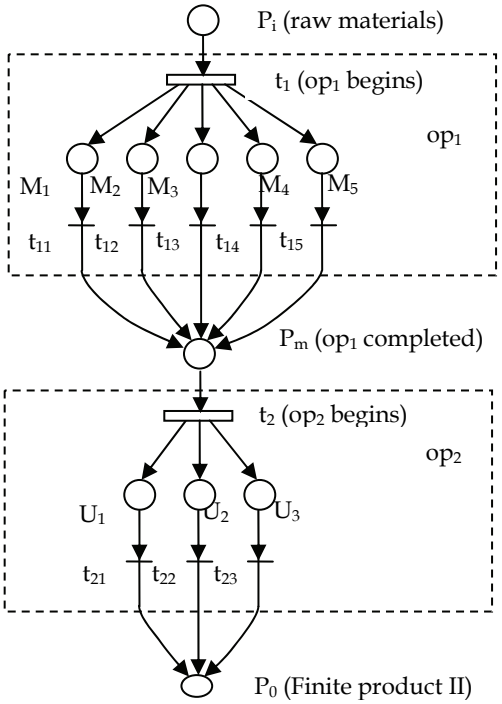



Fig. 5. The SCPN model for part type $L_1$ of the FMS from Fig.1

Six and four state Markov chains and transitions rate matrices (Fig.6 and Fig.7) were constructed for each production cell $op_1$ and $op_2$, respectively. Table 1 presents failure/repair data of the system components. The availabilities of cell 1 and cell 2 at time *t* are represented in Fig.8.

$$\begin{bmatrix} (3\lambda+\mu) & 0{,}8\mu & 0 & 3\lambda+0{,}2\mu & 0 & 0 \\ 3{,}2\lambda & (4\lambda+\mu) & 0{,}8\mu & 0 & 0{,}8\lambda+0{,}2\mu & 0 \\ 0 & 4\lambda & 5\lambda & 0 & 0 & \lambda \\ 0{,}8\mu & 0 & 0 & 0{,}8\mu & 0 & 0 \\ 0 & 0 & 0{,}8\mu & 0 & 0 & 0{,}8\mu \end{bmatrix}$$

Fig. 6. Transition rate matrix for production cell $op_1$

$$\begin{bmatrix} (2\lambda+\mu) & 0{,}8\mu & 2\lambda+02\mu & 0 \\ 2{,}4\lambda & 3\lambda & 0 & 0{,}6\lambda \\ 0{,}8\mu & 0 & 0{,}8\mu & 0 \\ 0 & 0{,}8\mu & 0 & 0{,}8\mu \end{bmatrix}$$

Fig. 7. Transition rate matrix for production cell $op_2$

## 5. Basic Equivalent Transfer Functions for SPNs

In a timed PN let F: T → R is a vector whose component is a firing time delay with an extended distribution function. By extended distribution functions, we mean that exponential distribution functions are allowed for concurrent transitions. Two transitions are said to be concurrent at marking m if and only if firing either does not disable the other. The firing rule for an SPN provides that when two or more transitions are enabled, the transitions whose associated time delays is statistically the minimum fires. According to the transition-firing rule in PN, when a transition $t_k$ has only one input place $p_i$, and $p_i$ is marked with at least one token, $t_k$ is enabled. The enabled transition can fire. The firing of $t_k$ removes one token from the $p_i$ and then deposits one token into each output place $p_j$. Let P(i, k) be a probability that transition $t_k$ can fire. The process from the enabling to the firing of $t_k$ requires a time delay, $\tau_k$. This delay $\tau_k$ of a transition can be either a constant or an extended random variable in SPN. P(i, k) and M(s) depend on $\tau_k$ as well as the current marking and the time delays of other enabled transitions at that marking. M(s) denote the moment generating function, and is defined as follows:

$$M(s) = \int_{-\infty}^{+\infty} e^{st} F_t(.)\, dt \tag{16}$$

Where s is an extended parameter, and $F_t(\cdot)$ is a probability density function of random variable-transition t (i.e. distribution function of the firing time of transition $t \in T_t$). A transfer function of a stochastic Petri net (Fu & Hu 1992) is defined as the product P(i, k)·M(s), and is:

$$W_k(s) = P(i,k) \cdot M(s) \tag{17}$$

Transition $t_k$ characterized by P(i, k) and $\tau_k$ is expressed by a transition characterized by $W_k(s)$. Four fundamental structures can be reduced into a single transition. The reduction rules can be used to simplify some classes of SPN. With these reduction rules we transform PN into finite state machines (in a finite state machine each transition has only one input and output place, and there is one token in such a net). Fig.8, a, b, c, d depict these reduction rules.

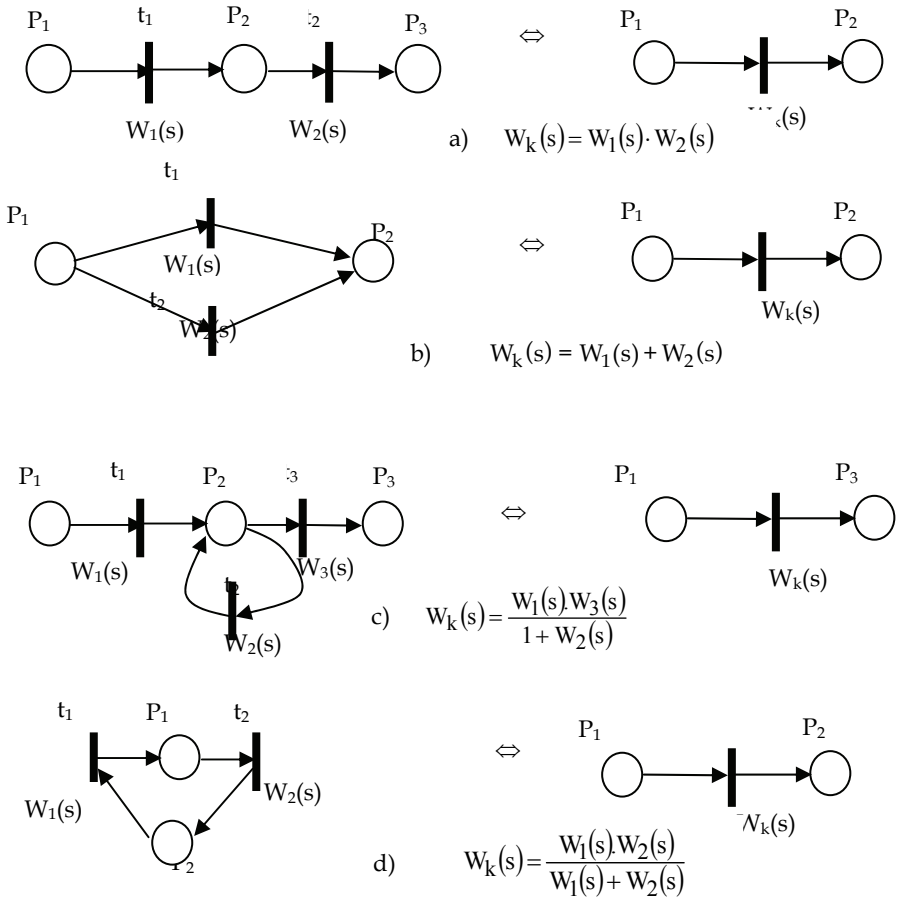



Fig. 8. Equivalent transfer functions for basic structures of SPN

The moment generating functions for the state machine SPN which models the construction systems represent the availability of the cells (subsystems) which form the SPN.

## 6. Perturbation Parameters Modelled with SPNs

Following the approach given in (Chiang et al., 2000), we suppose that the distributions of firing times depend on a parameter Θ. Parameters defined in section 2 are, in the above assumption, functions of Θ. In perturbation analysis the following results hold, where performance measures under consideration are of the form $g(M_1, t_1, \tau_1, \ldots, M_n, t_n, \tau_n)$ and a shorthand notation $g(\Theta)$ is used:

a) For each Θ, g(Θ) is a.s. continuously differentiable at Θ and the infinitesimal perturbation indicator is:

$$\frac{dg(\theta)}{d\theta} = \sum_{i=1}^{n} \frac{g}{\tau_i} \cdot \frac{d\tau_i}{d\theta} \tag{18}$$

b) If $d \in [g(\Theta)]/d\Theta$ exists, the following perturbation estimator is unbiased:

$$\sum_{i=1}^{n} \frac{g}{\tau_i} \cdot \frac{d\tau_i}{d\theta} + \sum_{k=1}^{n} \hat{f}_k(h_k) \cdot G_k \tag{19}$$

$$f_k = \frac{f_{tk+1}(L_k(t_{k+1}))}{F_{tk+1}(L_k(t_{k+1}) + y_k - F_{tk+1}(L_k(t_{k+1})))} \tag{20}$$

$$y_k = \min \{r_k(t) : \forall t \in T(M_k) - \{t_{k+1}\}\} \tag{21}$$

$$\tau_k = \frac{dL_k(t_{k+1})}{d\theta} - \frac{dX(t_{k+1})}{d\theta} \tag{22}$$

$L_k(t)$ is the age of time transition t at $S_k$; $G_k = g_{pp,k} - g_{DNP,k}$. The sample path ($M_1(\Theta)$, $t_1(\Theta)$, $\tau_1(\Theta)$, …,$M_n(\Theta)$, $t_n(\Theta)$, $\tau_n(\Theta)$) is the nominal path denoted by NP. The $g_{DNP,k}$ is the performance measure of the $k^{th}$ degenerated nominal path, denoted by $DNP_k$. It is identical to NP except for the sojourn time of the $(k+1)_{th}$ stable marking in $DNP_k$. $g_{pp,k}$ is the performance measure of a so-called $k^{th}$ perturbed path, denoted by $PP_k$. It is identical to $DNP_k$ up to time $s_k$. At this instant the order of transition $t_k$ and $t_{k+1}$ is reversed, i.e., the firing of $t_{k+1}$ completes just before that of $t_k$ in $PP_k$. We notice that by definition, $DNP_k$ and $PP_k$ are identical up to $s_k$. At $s_k$, the events $t_k$ and $t_{k+1}$ occur almost simultaneously, but $t_k$ occurs first in DNP and $t_{k+1}$ occurs first in $PP_k$. The commuting condition given in (Baccelli & Liu, 1992) guarantees that the two samples paths became identical after the firing of both $t_k$ and $t_{k+1}$. Our goal is to introduce a correction mechanism in the structure of the SPN so that the transition $t_k$ and $t_{k+1}$ fire in the desired order, and the routing mechanism given in relation (1) is re-established. We will exemplify this approach on an example, and we will correlate the theoretical assumption with some practical mechanisms in order to verify the approach.

### 6.1 Application to a Queuing Network

In Fig.9, we represented a workflow queuing network. The servers are $s_1$, $s_2$, and for any of them, if the downstream buffer is full, the customer is blocked until the downstream buffer has one hole. For simplicity of the Petri net model, we consider the perturbation analysis of only one way in the workflow (Ciufudean et al., 2008). In the corresponding SPN of the system in Fig.10, the transitions $t_1$ and $t_4$ model the arrivals. Transitions $t_3$, $t_6$, $t_7$, $t_9$ are used to model the materials departure between constructors.

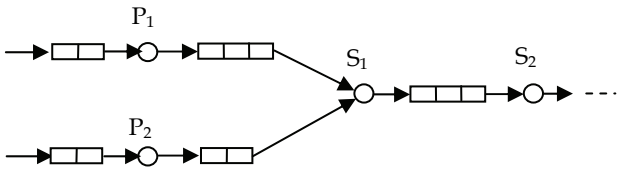



Fig. 9. A data queuing network with finite line capacity

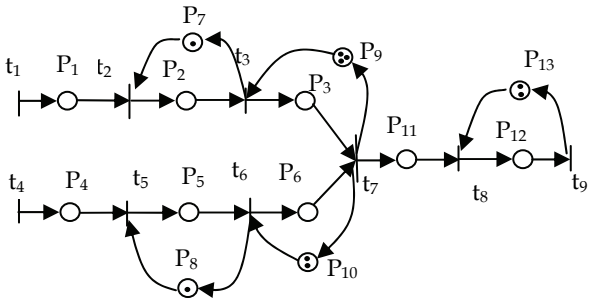



Fig. 10. The SPN model of the queuing network given in Fig. 9

The transitions $t_2$, $t_5$, $t_8$ model the service periods in the network. The holding times of the transitions $t_2$, $t_5$, $t_8$ in the SPN are identical to the service times of computers in the workflow. The information transmitted to $p_{11}$ by firing $t_7$ is determined by u′ (routing indicator defined in section 2, see relation (1)) when $t_2$ fires first and it is determined by u″ when $t_5$ fires first. Since u′ and u″ are independent random variables, the commuting condition given in (Xie, 1998) does not hold (i.e., $\varnothing(\varnothing(M, t_2, u'), t_5, u'')$ can be different from $\varnothing(\varnothing(M, t_5, u''), t_2, u')$. In order to make true the commuting condition we added in Fig.10 the locations $p_7$, $p_8$, $p_9$, $p_{10}$, and $p_{13}$ and corresponding arcs ensure a kanban mechanism in the SPN, in order to achieve the desired order in firing transitions $t_3$ and $t_6$, and, for $p_{13}$, a delay in materials transmission to the output. Locations $p_7$ and $p_8$ ensure the priorities in servicing of the materials flow arrivals (the arrival of the external raw materials).

For the average delay of demands ( $g = \frac{4}{n}\sum_{i=1}^{n} M_i(p)\cdot\tau_i$ ) the perturbation estimator given in (18) is unbiased.

$$\frac{4}{n}\sum_{i=1}^{n}\frac{g}{\tau_i}\cdot\frac{d\tau_i}{d\theta} = \frac{4}{n}\sum_{i=1}^{n} M_{i-1}(p)\cdot\frac{d\tau_i}{d\theta} \tag{23}$$

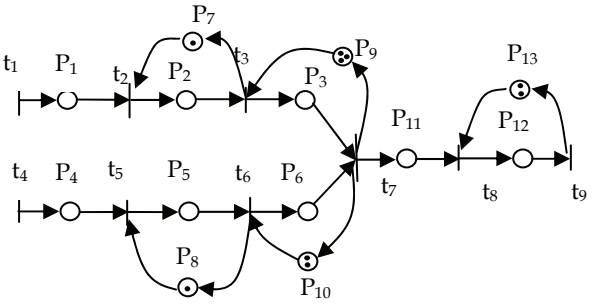
$$g = \frac{4}{n}\sum_{i=1}^{n} L(M_{i-1})\cdot\tau_i \tag{24}$$

Where $L(M_i) = M_i(p_1) + M_i(p_2) + M_i(p_3) + M_i(p_4) + M_i(p_5) + M_i(p_6) + M_i(p_8)$.

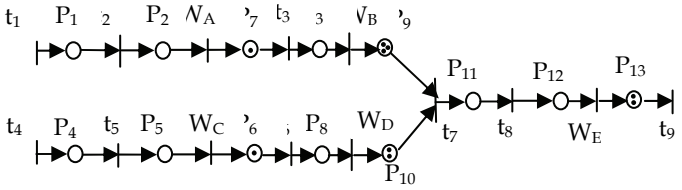
The perturbation estimator is equal to:

$$\frac{4}{n}\sum_{i=1}^{n} L\left(M_{i-1}\right).\frac{d\tau_i}{d\theta} \tag{25}$$

Assuming that firing times are exponentially distributed with mean equal to: Θ for $t_1$, $t_2$, $t_4$, and $t_6$; 1 for $t_3$; 0,86 for $t_5$; 0,75 for $t_8$, 0,9 for $t_9$, we consider the average customer delay (Θ). The mean value of the gradient evaluated at Θ = 1.22 and at Θ = 1.24 is close to the central finite difference: (E[g(1.24) – E[g(1.20)]) / 0,04 = -10.27. This result is acceptable, and we notice that additional values can be obtained by modifying the net structure as discussed before, and as it is drawn in Fig.10, by modifying the marking in the places $p_7$ and $p_8$. We notice that we can simplify the structure of the SPN in Fig.10 using the approach presented in chapter 5 (see Fig.8). This approach is useful when we deal with complex Petri nets, and we want to simplify these structures by reducing them to finite state machine, in order to analyze them properly. For the Petri net in Fig.10, we may have the following equivalent schemes, and correspondingly, the equivalent transfer functions. We notice that depending to the specificity of each modelling processes, or to the operator skills the reduction procedure can be stopped at a desired level of simplicity.




⇔

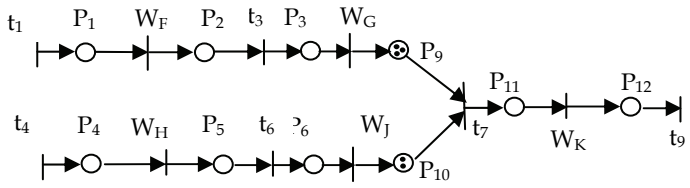



a) Where:

$$W_A(s)=\frac{W_{t_2}(s).W_{t_3}(s)}{W_{t_2}(s)+W_{t_3}(s)},\ W_B(s)=\frac{W_{t_3}(s).W_{t_7}(s)}{W_{t_3}(s)+W_{t_7}(s)},\ W_C(s)=\frac{W_{t_5}(s).W_{t_6}(s)}{W_{t_5}(s)+W_{t_6}(s)},$$

$$W_D(s)=\frac{W_{t_6}(s).W_{t_7}(s)}{W_{t_6}(s)+W_{t_7}(s)},\ W_E(s)=\frac{W_{t_8}(s).W_{t_9}(s)}{W_{t_8}(s)+W_{t_9}(s)}$$

⇔

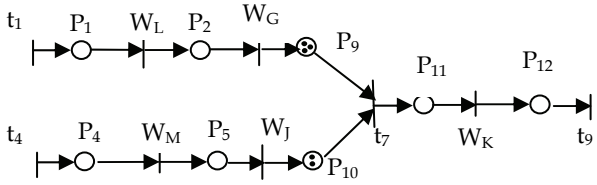



b) Where:

$W_F(s) = W_{t_2}(s).W_A(s)$, $W_G(s) = W_{t_3}(s).W_B(s)$, $W_H(s) = W_{t_5}(s).W_C(s)$,

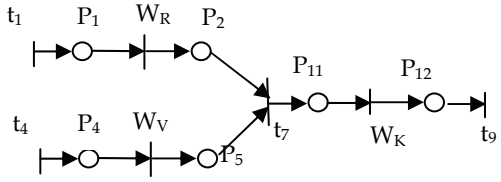
$W_J(s) = W_{t_6}(s)\quad W_D(s)$, $W_K(s) = W_{t_8}(s).W_E(s)$

⇔




c) Where: $W_L(s) = W_{t_3}(s).W_F(s)$, $W_M(s) = W_{t_6}(s).W_H(s)$

⇔




d) Where : $W_R(s) = W_L(s).W_G(s)$, $W_V(s) = W_M(s).W_J(s)$

Fig. 11. An example of aplying the equivalent transfer function for minimizing the size of a SPN

## 7. Application to a Flexible Manufacturing System

### 7.1 The system description

The manufacturing system considered in this paper consists of two cells linked together by a material system composed of two buffers A and B and a conveyor. Each cell consists of a machine to handle within cell part movement. Pieces enter the system at the load/unload station, where they are released from those two buffers, A and B, and then are sorted in cells (pieces of type "a" in one cell, and pieces of type "b" in the other cell).

We notice that in the buffer A are pieces of types "a", "b", where the number of pieces "a" is greater than the number of pieces "b". In the buffer B there are pieces of types "a", "b", where the number of pieces "b" is greater than the number of pieces "a". The conveyor moves pieces between the load/unload station ad the various cells. The sorted piece leaves the system, and an unsorted piece enters in the system, respectively in one of those two

buffers A or B. The conveyor along with the central storage incorporates a sufficiently large buffer space, so that it can be thought of as possessing infinite storage capacity. Thus, if a piece routed to a particular cell finds that the cell is full, it is refused entry and is routed back to the centralized storage area. If a piece routed by conveyor is of a different type of the required types to be sorted, respectively "a", and "b", then that piece is rejected out of the system. We notice that once a piece is blocked from entry in a cell, the conveyor does not stop service; instead it proceed with its operation on the other pieces waiting for transport. At the system level, we assume that the cells are functionally equivalent, so that each cell can provide the necessary processing for a piece. Hence, one cell is sufficient to maintain production throughput.

We say that the manufacturing system is available (or, operational) if the conveyor and at least one of the cells are available.

A cell is available if its machine is available. Over a specified period of operation, owing to the randomly occurring subsystem failures and subsequent repairs, the cellular construction system (CCS) will function in different configurations and exhibit varying levels of performance over the random residence times in these configurations. The logical model of our manufacturing system is showed in Fig.12.

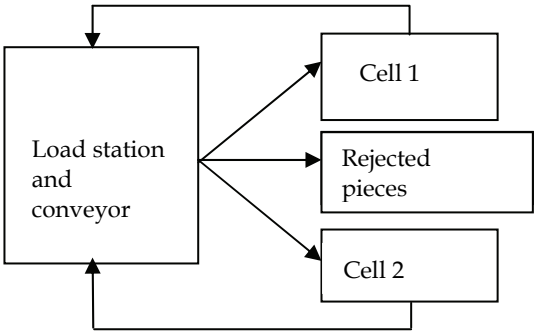
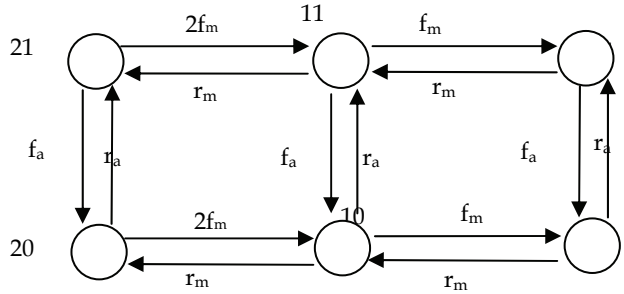



Fig. 12. Logical model for a flexible manufacturing system

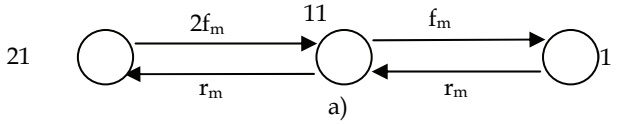
### 7.2 A Markov model for evaluating the availability of the FMSs

For the flexible manufacturing system depicted in Fig.12, we assume that the machines are failure-prone, while the load/unload station and the conveyor are extremely reliable. Assuming the failure times and the repair times to be exponentially distributed, we can formulate the state process as a continuous time Markov chain (CTMC). The state process is given by $\{X(u), u \geq 0\}$ with state space $S = \{(i\ j), i \in \{0, 1, 2\}, j \in \{0, 1\}\}$, where $i$ denotes the number of machine working, and $j$ denotes the status of the material handling system (load station and conveyor): up (1), and down (0). We consider the state independent, or time dependent failure case and the case of time independent, or operation dependent failure case separately. *Time dependent failures:* In this case, the component fails irrespective of whether the system is operational or not. All failure states are recoverable. Let $r_a$ and $r_m$ denote the repair rates of the material handling system, and a machine respectively. The state process is shown in Fig.13.a. Because the failure/repair behavior of the system components are independent, the state process can be decomposed into two CTMCs as shown in Fig.13.b. Analytically, the state process is expressed by relations: $S_0 = \{(21), (11)\}$

and $S_F$ = {(20), (10), (00)}. For each failure state in $S_f$ no production is possible since the Material Handling System (MHS) or both the machines are down.

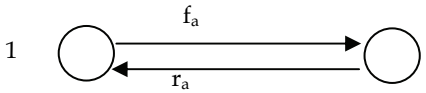



for machines:




a)

for MHS:




b)

Fig. 13. State process of a CCS with time-dependent failures, (a) State process for a state-independent failure model, (b) Decomposed failure/repair process

In Fig.13.b the failure/repair behaviour of each resource type (machines or MHS) is described by a unique Markov chain. Thus, the transient state probabilities, $p_{ij}(t)$ can be obtained from relation (Benson &Zang, 2000):

$$p_{ij}(t) = p_i(t) \cdot p_j(t) \tag{26}$$

Where $p_i(t)$ is the probability that $i$ machines are working at time $t$ for $i$ = 0, 1, 2. The probability $p_i(t)$ is obtained by solving (separately) the failure/repair model of the machines. $P_j(t)$ is the probability that $j$ MHS (load/unload station and conveyor) are working at instant $t$ , for $j$ = 0, 1. Let $f_a$ and $f_m$ denote the failure rates of the MHS and of a machine respectively. *Operation-dependent failures:* Assume that when the system is functional, the resources are all fully utilized. Since failures occur only when the system is operational, the state space is: S = {(21), (11), (20), (10), (01)}, with $S_0$ = {(21), (11)}, $S_f$ = {(20), (10), (01)}. The Markov chain model is shown in Fig.14. Transitions representing failure will be allowed only when the resource is busy. Transitions rates can however be computed as the product of the failure rates and percentage utilization of the resource, and $T_k^{ij}$ represents the average utilization of the $k^{th}$ resource in the state (i j).

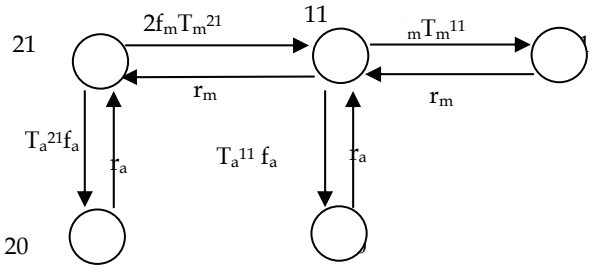



Fig. 14. State process of a CCS with state-dependent failures

### 7.3 Numerical study

For the CCS presented in this paper, in the table 1 are given the failure/repair data of the system components We notice that $T_k^{ij} = 1$ since the utilization in each operational state is 100% for all *i, j, k,* where $i = \{0,1,2\}$, $j = \{0,1\}$, $k = 4$.

The other notations used in table 1 are: $f$ is the exponential failure rate of resources, $r$ is the exponential repair rate of resources, $N_p$ is the required minimum number of operational machines in cell $p$, $p = \{1,2\}$, and $n_p$ is the total number of machines in cell $p$.

| | r | f | $N_p$ | $n_p$ | $T_k^{ij}$ |
|---|---|---|---|---|---|
| Machines | 1 | 0,05 | 1 | 2 | 1 |
| MHS | 0,2 | 0,001 | 1 | 1 | 1 |

Table 1. Data for the numerical study

From Fig.13 and Fig.14 we calculate the corresponding infinitesimal generators and after that, the probability vector of CTMC. With (11) we calculate the availability of CCS.

The computational results are summarized in Table 2 for the state process given in Fig.13 (CCS with time-dependent failures), and respectively in Table 3 for the state process given in Fig.14 (CCS with state-dependent failures). We consider the system operation over an interval of 24 hours (three consecutive shifts).

| Time hour | Machines | MHS | System Availability |
|---|---|---|---|
| 0 | 1.0000 | 1.0000 | 1.0000 |
| 1 | 0.9800 | 0.9548 | 0.9217 |
| 4 | 0.9470 | 0.8645 | 0.7789 |
| 8 | 0.9335 | 0.8061 | 0.7025 |
| 12 | 0.9330 | 0.7810 | 0.6758 |
| 16 | 0.9331 | 0.7701 | 0.6655 |
| 20 | 0.9330 | 0.7654 | 0.6623 |
| 24 | 0.9328 | 0.7648 | 0.6617 |

Table 2. Computational results for the CCS in Fig. 13

| Time hour | Machines | MHS | System Availability |
|---|---|---|---|
| 0 | 1.0000 | 1.0000 | 1.0000 |
| 1 | 0.9780 | 0.9528 | 0.9201 |
| 4 | 0.9450 | 0.8628 | 0.7762 |
| 8 | 0.9315 | 0.8039 | 0.7008 |
| 12 | 0.9310 | 0.7798 | 0.6739 |
| 16 | 0.9320 | 0.7688 | 0.6632 |
| 20 | 0.9318 | 0.7639 | 0.6598 |
| 24 | 0.9320 | 0.7636 | 0.6583 |

Table 3. Computational results for the CCS in Fig. 14

The results of the availability analysis of the construction system are illustrated in Fig.15, which depicts the availability of the system as a function of the time. The numbers $x$ = 1, 2 indicate the system in Fig.13, respectively Fig.14. One can see from Fig.15 that the layout with CCS with time-dependent failures is superior to that with CCS with state-dependent failure.

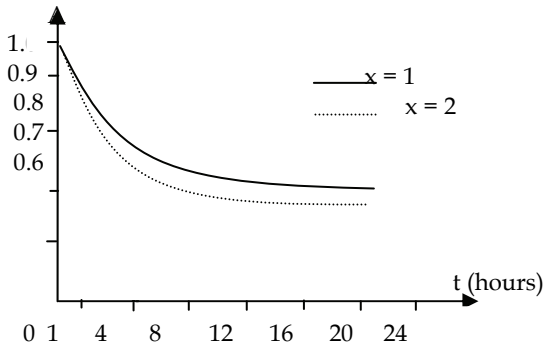



Fig. 15. Availability analysis of the CCS

## 8. Conclusion

In this paper we analysed the control charts estimators in discrete event systems modelled with stochastic Petri nets (SPNs). The approach presented in this paper (e.g. Stochastic Artificial Social Systems) can be used to analyze SPNs that model complex dynamic system interactions. Unbiased gradient estimators proposed in (Haas & Shendler, 1991) were used for the sensitivity analysis of the GSMP representation and some practical solutions for attenuating the perturbations influences were indicated. The proposed procedure was imagined for a data network perturbation analysis. We estimate that this methodology can be applied to modelling and analysis of manufacturing systems, job scheduling in a chain management system, such as flexible manufacturing systems. Future research will focus on differential and fluid Petri nets in order to estimate throughput of complex systems.
An analytical technique for the availability evaluation of the flexible manufacturing systems was presented. The novelty of the approach is that the construction of large Markov chains

is not required. Using a structural decomposition, the construction system is divided into cells. We can simplify the structure of the SPN using the approach presented in chapter 5 (see Fig.8). This approach is useful when we deal with complex Petri nets, and we need to simplify these structures (e.g. graphs) in order to analyze them properly. For each cell a Markov model was derived and the probability was determined of at least $N_i$ working machines in cell *i,* for $i = 1, 2, .., n$ and *j* working material handling system (MHS) at time *t*, where $N_i$ and j satisfy the system production capacity requirements.

We also presented a throughput diagnoser controller for flexible manufacturing system. Our approach Is based on Petri nets and Markov chains formalism. We presented a general framework for constructing controllers using Petri nets and for constructing diagnosers using Markov chains. Performance metrics for the diagnosers were also defined. These diagnosers can be easily incorporated into an controller built with Petri nets as every Petri net has a back-up Markov chain.

The model presented in this paper can be extended to include other components, e.g., tools, control systems, etc. The results reported here can form the basis of several enhancements, such as conducting performance studies of complex systems with multiple part types.

## 9. References


Archetti, F.; Gaivoronski, A.; Sciomachen, A. (1993). Sensitivity analysis and optimisation of stochastic Petri nets, *Journal of Discrete Event Dynamic Systems: Theory Appl*., pp. 5-37, vol. 3.

Baccelli, F.; Liu, Z. (1992). Comparison properties of stochastic decision free Petri nets*, IEEE Trans. on Autom. Contr.,* pp. 1905-1920, vol. 37, no. 12.

Benson, S.; Zang, Ye, X. (2000). Solving large-scale sparse semidefinite programs for combinatorial optimization, *SIAM Journal Optimization*, pp.443-461, no.10.

Brehends, E. (2000). Introduction to Markov Chains with Special Emphasis on Rapid Mixing, *Advances Lectures on Mathematics*, Vieweg, Germany.

Carmen, T.; Leiseron, C. ; Rivest, R. (1991). *Introduction to Algorithms*, The MIT Press.

Chiang, S. I.; Kuo, C.T.; Meerkov, S.M. (2000). DT-Bottlenecks in serial production lines: Theory and Applications, *IEEE Trans. Autom. Contr*., pp. 567-580, vol. 16, no. 5.

Ciufudean C. (2000). Modeling the reliability of the interaction man-machine in railway transport*, The Annals of the "Stefan cel Mare" University of Suceava*, pp. 80-84, no.13.

Ciufudean, C., Larionescu, A. (2002). Safety criteria for production lines modeled with Petri nets*, Advances in Electrical and Computer Engineering*, pp. 15-20, vol. 2(9), no. 2(18), Stefan cel Mare University of Suceava, Romania.

Ciufudean, C.; Popescu, D. (2004). Modelling Digital Signal Perturbation with Stochastic Petri Nets, *Advances in Electrical and Computer Engineering*, pp. 71-75, vol. 4(II), no. 1 (21), Stefan cel Mare University of Suceava, Romania.

Ciufudean, C.; Graur, A. (2005). Availability of Fluid Stochastic Event Graphs, International *Conference on Control and Automation, ICCA 2005*, June 26-29, Budapest, Hungary, pp. 35-39, ISBN 0-7803-9138-1, IEEE Catalog Number: 05EX1076C.

Ciufudean, C.; Petrescu, C.; Filote, C. (2005). Performance Evaluation of Distributed Systems*, International Conference on Control and Automation, ICCA 2005*, June 26-29, Budapest, Hungary, pp. 21-25, ISBN 0-7803-9138-1, IEEE Catalog Number: 05EX1076C.

Ciufudean, C.; Filote, C.; Popescu, D. (2006). Worflows in Constructions Modelled with Stochastic Artificial Petri Nets, *The 23rd International Symposium on Automation and Robotics in Construction, ISARC 2006*, pp. 773-778, October 3-5, Tokyo, Japan.

Ciufudean, C.; Satco, B.; Filote, C. (2007). Reliability Markov Chains for Security Data Transmitter Analysis, *The Second International Conference on Availability, Reliability and Security, ARES 2007*, 10th – 13th of April, pp. 886-892, Vienna University of Technology, Austria.

Ciufudean, C.; Filote, C.; Amarandei, D. (2007). Measuring the Performance of Distributed Systems with Discrete Event Formalisms, *The 2nd IEEE –IAS Seminar for Advanced Industrial Control Applications, SAICA 2007*, pp. 263-267, November 5-6, Madrid, Spain.

Ciufudean, C.; Filote, C.; Amarandei, D. (2008). Control Charts of Workflows, *Advances in Data Mining, 8th Industrial Conference ICDM 2008*, pp. 330-344, Leipzig, Germany, July18-20, 2008, Springer Verlag Berlin-Heidelberg.

Cover, T.; Thomas, J. (1991). *Elements of Information Theory*, John Wiley and Sons.

Fu, M. C.; Hu, J. Q. (1992). Extensions and generalizations of smoothed perturbation analysis in a generalized semi-Markov process framework, *IEEE Trans. Automat. Contr.*, pp. 1483-1500, vol. 37.

Haas, P. J.; Shendler, G. S. (1991). Stochastic Petri nets: Modelling power and limit theorems, *Probability Eng. Inform. Sci.* , pp. 477-498, vol. 5.

Hopkins, M. (2002). Strategies for determining causes of events, *Technical Report R-306*, UCLA Cognitive Systems Laboratory.

Hopkins, M. (2002). A proof of the conjuctive cause conjecture in causes and explanations, *Technical Report R-306*, UCLA Cognitive Systems Laboratory.

Karlin, S.; McGregor, J.L. (1959). Random Walks, *Illinois J. Math.*, pp. 66-81, no. 3.

Lane, T. ; Bradley, C. (1992). Temporal sequence learning and data reduction for anomaly detection, *ACM Transactions on Informatics and System Security*, pp. 295-331, no. 2(3).

Lee, W.; Stoflo, S.; Mak, K. (1999). A data mining framework for building intrusion detection models, *IEEE Symposium on Security and Privacy*, pp. 139-145, vol.15.

Murata, T., (1989). Petri nets - Properties, analysis and applications, *Proc. IEEE*, vol. 77, pp. 541-580.

Recalde, L.; Teruel, E.; Silva, M. (1998). Modelling and analysis of sequential processes that cooperate through buffers, *IEEE Transaction on Robotics and Automation.*, pp. 267-277, vol. 14, no. 2.

Schrijner, P. (1995). *Quasi-Stationarity of Discrete-Time Markov Chains*, University of Twente, The Netherlands.

Yee, Sh. T.; Ventura, J. A. (2000). Phase-type approximation of Petri nets for analysis of manufacturing systems, *IEEE Trans. on Rob. and Autom.*, pp. 318-322, no. 3.

Xie, X. (1998). Perturbation analysis of stochastic Petri nets, *IEEE Trans. on Autom. Control*, pp. 76-80, vol. 43, no. 1.

Zuo, M.J.; Liu, B.; Murthy, D.N.P. (2000). Replacement-repair policy for multi-state deteriorating products under warranty, *European Journal of Operational Research*, pp. 519-530, no.123.

23

# Evaluation of Power System Security with Petri Nets

Jose L. Sanchez, Mario A. Ríos and Gustavo Ramos
*Universidad de los Andes*
*Colombia*

## 1. Introduction

Electrical power systems provide electricity supply to a considerable amount of people around the world. There are three main processes in order to deliver that electricity to end users, those are: generation, transmission, and distribution, they work as a chain process, where each one requires to bring a reliable, secure, and stable service. Therefore, there is an increased interest for developing tools that allow the security evaluation of power systems, and also to ensure high levels of quality, reliability, and availability. However, these levels are affected by some factors that have been recognized as contributing elements in order to lead to catastrophic and cascading events, such as: uselessness and hidden failures of protections in Power Systems. Nevertheless, those can be modelled as probabilistic events, which can be calculated taking into account the operating sequences, considering the following operating states: normal, alert, emergency, extreme emergency and restorative (Lester & Carlsen, 1978).

According to the NERC 2008 annual report (NERC, 2009), equipment failures are involved in about the 23% of major disturbances, and protection misoperations are involved in about 42% of major disturbances. Thus, the protective system plays an important role in power system operation, and a very important role in causing cascading events. However, there are still major blackouts in spite of technological advances and huge investments in system reliability, adequacy, and security.

For that reason, it is important, besides to reinforce the protection systems, to develop new tools to analyze, study, and measure the impact of determined protection system failures in terms of adequacy, security, and reliability. Thus, researchers have proposed several models for reliability and security evaluation of substations (Dong et al, 2003) and (Dobson & McCalley, 2008). However, the main weakness is to forget the impact of a substation fault over neighbors substations, as well as, the uncertainty in the appropriate response of the protective systems.

In the set of protection devices, circuit breakers are very critical components because they are the last barrier to protect other devices of a Power System against faults. Thus, a detailed study of these devices allows to find the root causes and dynamics of cascade events in Power Systems. As well, it is important to quantify the probability of failure of the whole

system to assess the risk to lead a voltage collapse, taking into account the performance and, furthermore, the unreadiness of the breakers (Anderson, 1999).
Petri Nets theory allows the evaluation of the Power Systems Security considering the system response to sudden disturbances produced by short circuits and component outages, and the computation of operating state probabilities considering the probability of the appropriate operation of each protection device.
This Chapter provides a comprehensive review of the application of Petri nets in security and reliability analysis of Power Systems, such as, electrical industrial systems, meshed power systems and substations. Additionally, special emphasis is done about modelling and interpretation related to the conventional definition of operating states in power systems, i.e. normal state, alert state, emergency, extreme emergency state, and restorative state.
Some applications of Petri Nets are:

- Modelling of cascade phenomenon after operation of protection devices.
- Computing of possible electrical failures in power system protections.
- Risk evaluation of power system unavailability.

Some particular properties that will be covered include:

- Coverability trees, which help to select the operating state of the system.
- All states can be reached in protection schemes of power systems, which represent the liveness of the system. Additionally, in power systems there are no deadlocks, which allow using Petri Networks in order to simulate the behaviour of the system.
- Because there is a restoration state in the system, the property of reversibility is applied.

The proposed analysis looks at the events: how, when, and in what order they occur. Each answer provides detailed information in order to analyse the substation behaviour and the security of the system. Such analysis is summarized in tables, which show the different sequences after a failure of a protective system, and how a single failure leads to a cascade event. All these outcomes allow planning and designing better, more reliable, and more secure systems.

## 2. General Overview of Power System Security

Security is defined as the ability of the power system to respond to sudden disturbances without supply interruption. Therefore, security analysis must evaluate non-appropriate response of the system, unnecessary operation of any device (such as protection devices) and/or bad operation of some subsystems when a sudden disturbance occurs. Any of these events affect the power quality and/or the reliability of the electrical system and, consequently, the electrical infrastructure and the associated productive industrial processes lead to a risky scenario.
Operating states of power systems were proposed in (Lester & Carlsen, 1978). They divided the operating states of the power systems into five stages, those are: Normal, Alert, Emergency, In Extremis and Restorative, their interactions are showed in Fig. 1. These states were defined as follows:

- Normal: The system operates satisfying all the constraints of the system, so that all substations can supply the load demand that is required to ensure the proper functioning of the system. None of the protective equipment or lines is being overloaded.

– Alert: The system is still operating, but some operating constraints are not met within the system as a result of the overloading of some protective system. In this state corrective actions should be carried out to avoid a blackout in the system and thus return to normal state. For non-controlled transitions, there is a reduction in the security level. Therefore, the system is susceptible and vulnerable to subsequent interruptions. Possibly due to either unexpected increases in loads, not boot-generating machines, loss of generating units, loss of transmission lines, or increased levels of risk due to storm or natural disasters.
– Emergency: System constraints have been exceeded; these constraints are related to some of the following variables: voltage levels, system frequency, and angles of machines or buses. The security level is low, therefore control measures should be undertaken to bring the system into the alert state. However, the system is still intact.
– In Extremis: In this state the constraints of the system have been violated, and the system has lost significant loads. Thus, the system is not still intact. Actions must be undertaken to restore the supply to all loads making reconnections.
– Restorative: After taking control action in the system. It is reconnected, and the loads are returned.

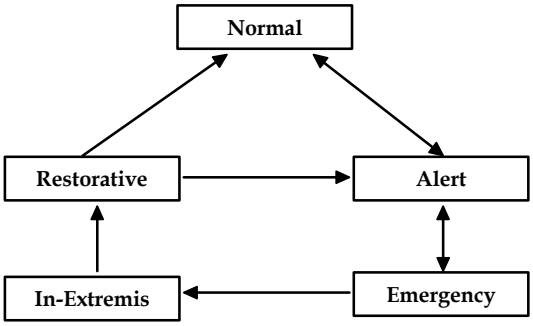



Fig. 1. Operating States of Power Systems

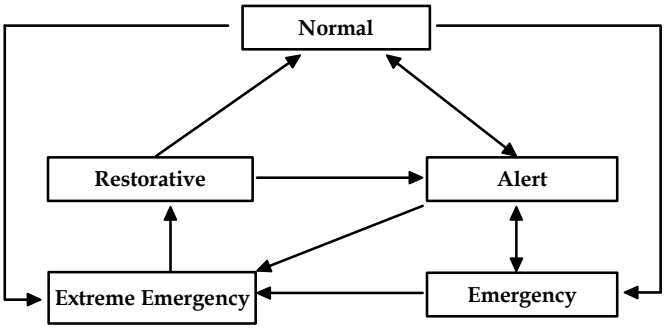



Fig. 2. Improved Operating States of Power Systems

However, the analysis shown above was improved (EPRI, 1987) and (Kundur, 1994). Basically, they added new transitions between the different states. Thus, one of the most

complete diagrams is found in (Billinton et al., 1997), which associates the different states as shown in Fig. 2.
This diagram has transitions from normal state to the state of emergency, and from normal state to extreme emergency state. Likewise, it adds a transition between alert state and extreme emergency state. It is important to mention that the fourth state changed its name, from In-Extremis State to Extreme Emergency State.

## 3. Petri Net Modelling

A Petri Network is a graphical and mathematical tool to model synchronization process, asynchronous events, sequential operations, concurrent operations, conflicts and resources management (Wang, 1998). PN models can be readily used to describe the system behavior by means of causal relationships between conditions and events in a sequential way. For that reason, PN's are very useful for the analysis of various industrial processes such as production facilities, modeling of electrical systems, and computational systems.
The main components to define the operating states of the Power Systems according Fig. 2, which are taken into account for the formulation of the PN, are the protective devices, such as breakers. So the main events used in the PN formulation are: short circuits, interruption of energy supply, and power quality problems.
Thus, taking into account the sequence operation of protective devices when a sudden disturbance occurs, the unreadiness probability, or the probability of non response of the protections when they are needed, is equivalent to the conditional probability of non-operation when the disturbance is present (Jenkins & Khincha, 2006).
As in power systems, the non-secure probability is computed from the probabilities that the system reaches an emergency or extreme emergency state, when a sudden disturbance occurs (such as a short circuit) as function of the operation of main and back-up protections. Therefore Petri Networks allows to study power systems, considering unreadiness probabilities and operating states of protections.
Then, the security assessment consists in the computation of the probability of those operating states when a fault occurs in the Power System. Hence, the security assessment methodology using PN is as follow, according to Fig. 3:

1. Definition of the event to be analyzed. It consists in selecting the initiating event. Such event could be short circuits and component outages.
2. System definition: Identification of protective devices such as circuit breakers and relays, including protection zones.
3. Study of possible failures: Main failures are selected in order to simulate their effects on the system. All possible operative states for each device and event are established.
4. Create Petri Net structure. For instance, for each fault in a Power System the PN model is built according to Fig. 4.
5. Assign an operating state for each place in the Petri Net. Fig. 4 shows that a place corresponds to alert state, another place to emergency state and the last one corresponds to extreme emergency state. This step might be done for each place of the PN.
6. Simulation and validation of the Petri Net model, in the Petri Net toolbox V 2.3. (Matcovschi, 2005). This software allows also to study certain properties of the Petri

Net that gives important information about the PN, some of the properties are: liveness, reachability, among others.

7. Generation of the Coverability graph and identification of the operating state of the system.
8. Security index assessment. After simulating the PN, we know how many times the tokens passed through each place, which allows to compute the occurrence probability of each operating state.

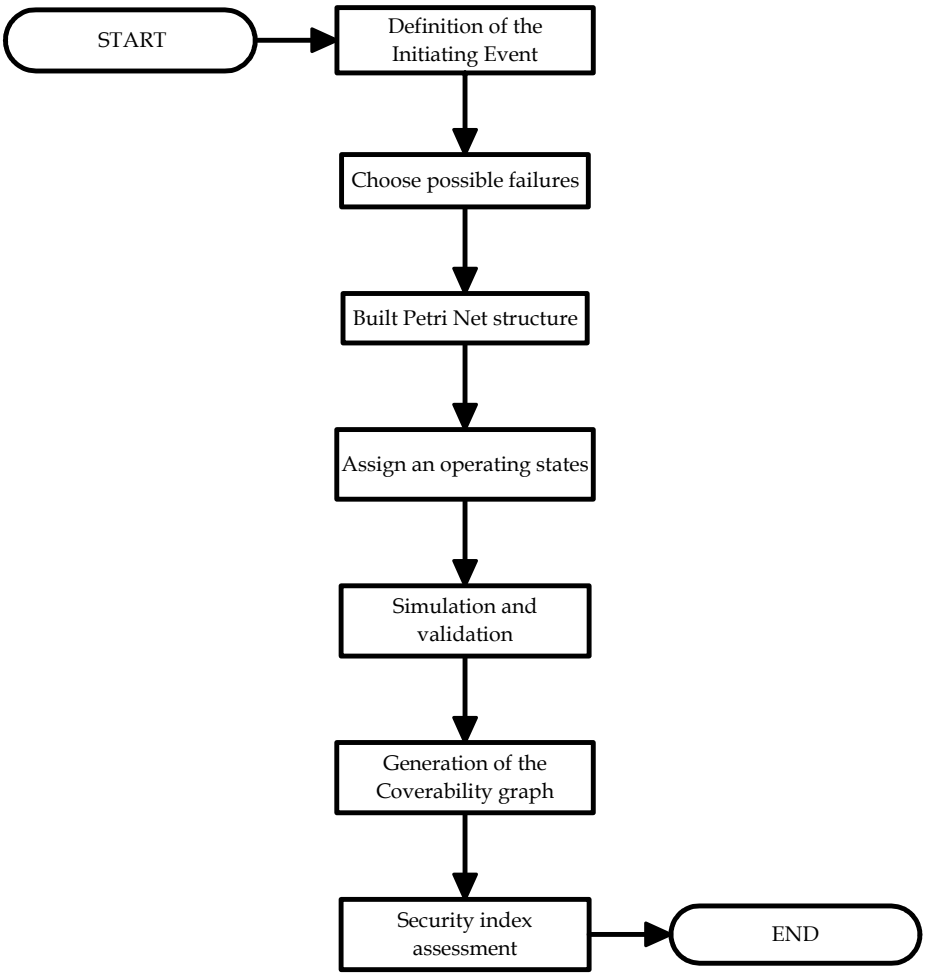



Fig. 3. Security assessment methodology

In order to apply the new methodology for modeling protection sequences with Petri Nets, Fig. 5 presents a power system with 2 buses. Considering two possible failures with the same probability of occurrence, the first one is located close to bus L, and the second one

close to bus M. The Petri Net for this system is presented in Fig. 6. As well, places and transitions descriptions are in Table 1 and Table 2.

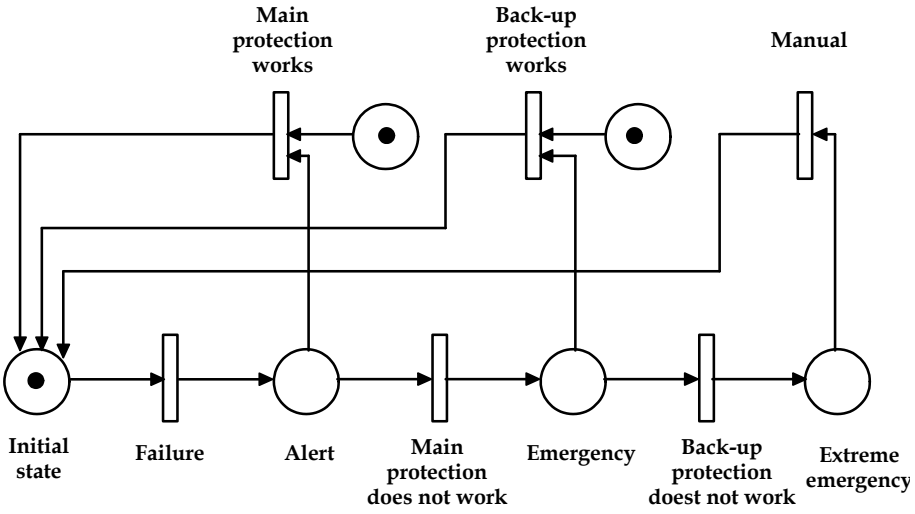



Fig. 4. Proposed Methodology for PN Models of Protection Systems (Sánchez et al. 2008)

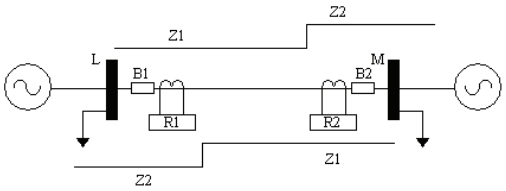



Fig. 5. Two Buses System

| Place | Description |
|---|---|
| p1 | Initial state |
| p2 | Fault close to bus L state |
| p3 | Availability of main protection |
| p4 | Isolation of fault by main protection |
| p5 | Emergency state |
| p6 | Availability of back-up protection |
| p7 | Isolation of fault by back-up protection |
| p8 | Extreme emergency state |
| p9 | Isolation of fault by manual operation on main protection |
| p10 | Isolation of fault by manual operation on back-up protection |
| p11 | Fault close to bus M state |
| p12 | Isolation of fault by main protection |
| p13 | Emergency state |
| p14 | Isolation of fault by back-up protection |

Table 1. Petri Net Places for 2 Buses System

For the 2 buses system, the Petri Net was simulated with PetriNet Toolbox, and the results - using a failure probability of 5%. Because all protections are taken with the same configuration and failure probabilities, it is expected that almost 5% of cases correspond to emergency state.

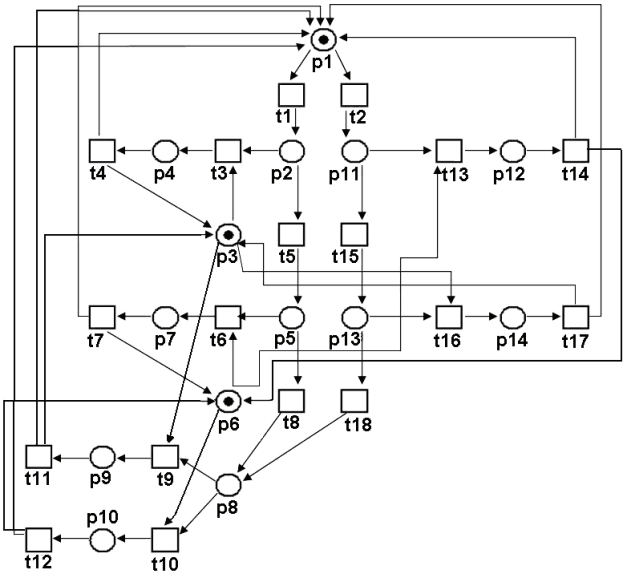



Fig. 6. Petri Net for 2 buses System

| Transition | Description |
|---|---|
| t1 | Fault close to bus L |
| t2 | Fault close to bus M |
| t3 | Main protection fired |
| t4 | System restoration |
| t5 | Main protection failed |
| t6 | Back-up protection fired |
| t7 | System restoration |
| t8 | Back-up protection failed |
| t9 | Manual operation |
| t10 | Manual operation |
| t11 | System restoration |
| t12 | System restoration |
| t13 | Main protection fired |
| t14 | System restoration |
| t15 | Main protection failed |
| t16 | Back-up protection fired |
| t17 | System restoration |
| t18 | Back-up protection failed |

Table 2. Petri net Transitions for 2 Buses System

## 4. Applications

### 4.1 Electrical Industrial Systems (EIS)

#### 4.1.1 Small Radial Systems

The PN modeling is applied to evaluate the sequence operation of a typical protection scheme, as Fig. 7 shows. The system is composed by a local primary protection (B2) and for a remote (backup) protection (B1). The PN model of Fig. 8 shows the transitions scheme between states and specifies the alert, emergency and extreme emergency states. This PN can be used to evaluate the security of the EIS when a fault F1 take place downstream B2.

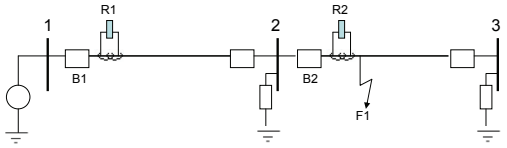



Fig. 7. Typical protection System

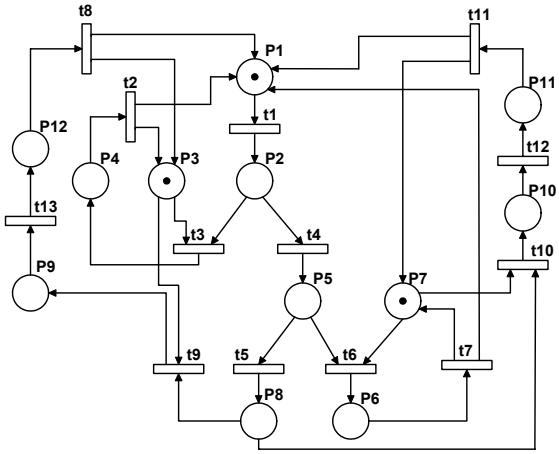



Fig. 8. Petri Net for the EIS basic protection System

The normal state is represented by P1 and the system could reach the P2 state (faulted system) when the transition t1 occurs (short circuit at F). From P2, the system could reach P4 or p5 states by action of conflicting transitions t3 (action of the main protection B1 happens) and t4 (non-action of the main protection B1 happens), respectively. So, in P2 a logical decision is taken between action and non-action of the main protection modeled as a probability (Anderson & Agarwal, 1992) of appropriate operation when it is required.

If the system has reached the p4 state, the transition t2 that represents the restoration of the system, it moves to the P1 state, i.e. to the normal operation state. By contrast, if the system reaches the p5 state (non-operation of the main protection); the same analysis is made for the operation of the backup protection device.

Table 3 lists the places or system states that can be reached by the system when a disturbance occurs; while Table 4 presents the states' transitions. So, from Table 3 and Table

4 a direct relationship is established to the operational states of Fig. 2. Hence, the normal operational state is P1 in the Petri model, alert state is P4, emergency state is P5, extreme emergency state is P6 and restorative state is the addition of P9 and P10.

| Place | Description |
|---|---|
| P1 | Normal State |
| P2 | Faulted System between nodes 2 and 3 |
| P3 | Main protection available |
| P4 | Alert State - Fault isolated by main protection |
| P5 | Emergency State by non-operation of main relay |
| P6 | Emergency State – Fault isolated by backup protection |
| P7 | Backup protection available |
| P8 | Extreme Emergency State by non-operation of backup relay |
| P9 | Restorative State – system in reparation main→backup |
| P10 | Restorative State – system in reparation backup→main |
| P11 | Restorative State – system repaired |
| P12 | Restorative State – system repaired |

Table 3. Places for the Petri Net representation of Basic Protection System

| Transition | Description |
|---|---|
| T1 | Fault F1 occurs |
| T2 | Main protection (R2) closes breaker B2 |
| T3 | Fault clearance by main protection (R2) (B2 opens) |
| T4 | Main protection (R2) doesn't operate→ B2 is closed |
| T5 | Backup protection (R1) doesn't operate→ B1 is closed |
| T6 | Fault clearance by backup protection (R1)(B1 opens) |
| T7 | Backup protection (R1) closes breaker B1 |
| T8 | Manual operation main protection (B2 closes) |
| T9 | Manual operation main protection (B2 opens) |
| T10 | Manual operation main protection (B1 opens) |
| T11 | Manual operation main protection (B1 closes) |
| T12 | Fault in line 2-3 is eliminated |
| T13 | Fault in line 2-3 is eliminated |

Table 4. Transitions for the Petri Net Representation of a Basic Protection System

The security indicators are computed by simulations on the Petri Net. Thus, a 10000 probabilistic trials simulation has been made on the PN of Fig. 5 assuming a probability of 95% of appropriate operation of the main and backup protection devices. Thus, for each trial the token is moved through the system states by activation of transitions. The activation of transitions takes into account when a decision between conflicting transitions must be taken. As final result, Fig. 9 shows the conditional probabilities to reach the normal, emergency and extreme emergency states when a fault at F1 happens.

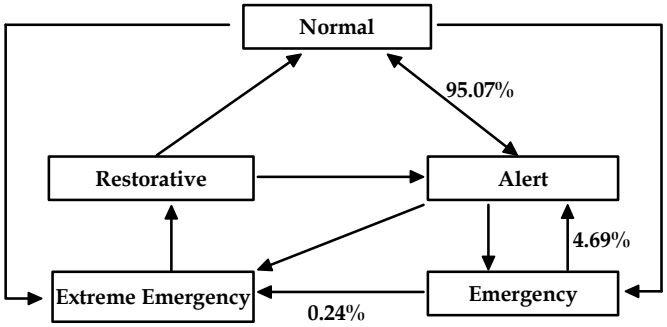



Fig. 9. Probabilities of Operational States Transitions

### 4.1.2 System IEEE 493

*Test System:* The electric supply of the IEEE 493 system (Koval et al., 2003) is employed as test system, developed to test methodologies of reliability evaluation in EIS. The IEEE 493 system is a dual utility source system with standby generation in configuration to many mission-critical electric systems, serving both military and commercial facilities. Service transformers are supplied by two independent 15-kV primary distribution feeders. There are four diesel engine generators in the facility, where two of four generators are required to meet the network load demand at all time (Koval et al., 2002).

The complete PN for the supply analysis of the IEEE 493 system is built in two phases: PN for the automatic operation of the main switchgear (generation and utility supply) and the PN for the alimentation of loads from the main switchgear.

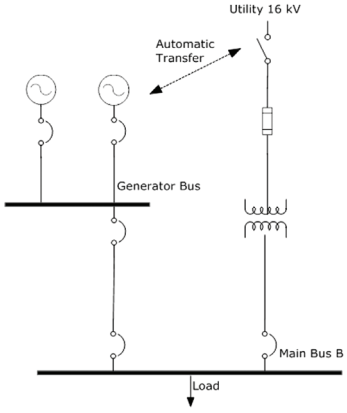



Fig. 10. Automatic Transfer System Scheme IEEE 493 Trest System

*Automatic Transfer System:* Automatic transfer switches are an integral part of the power generation process, shown in Fig. 10. If the power supply from the utility is interrupted, the transfer switch sends a start signal to the generator and then transfers the load. When the utility power returns, the transfer switch stops the generator and transfers the load. Fig. 11

shows a PN model for automatic transfer system with power utility and generator systems (Ramos et al., 2009).
Table 5 lists the places or system states that can be reached by the system when an outage occurs and Table 6 presents the transitions of the states. So, from Table 5 and Table 6 a direct relationship is established to the operating states of Fig. 12. Hence, the normal operating states are P1 and P8 in the Petri Model, alert state is P3, emergency states are P5 and P11, extreme emergency states are P6, P7, P9 and P10, and restorative states are equivalent to extreme emergency states.
Conflicting transitions are present when: a supply outage is present (T2-T3), the generator is starting (T6-T7), and the generator fails after starting (T8-T10).

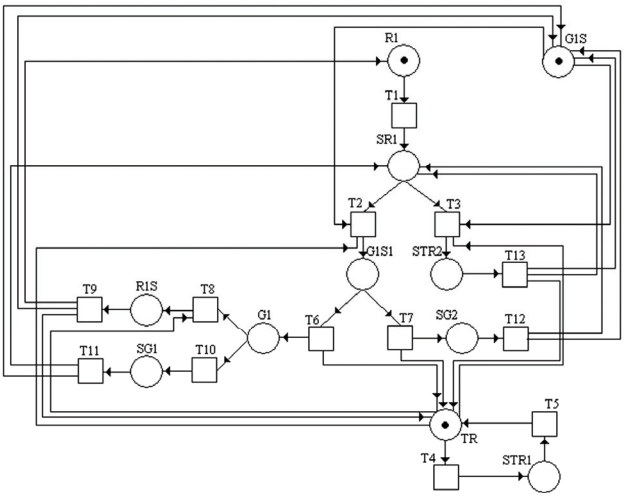



Fig. 11. Petri Net for Transfer System

Fig. 12 shows the conditional probabilities to reach the normal, emergency, and extreme emergency status when an outage happens assuming a probability of 95% of appropriate operation of the generator and transfer switch.
*Load System:* The load system is composed by a main distribution circuit and two branch circuits, Bus A and Mech Bus A. The main distribution circuit is protected for Main Bus A protection (4000A), the branch circuits are protected for Bus A protection (1600A) and Mech Bus A protection (800A), shown in Fig. 13. Each circuit has primary and secondary protections; e.g. Bus A protection is the primary protection and Main Bus A protection is the back-up protection for the Bus A circuit. Fig. 14 shows the PN for the load system which is similar to the typical protections system with two branch circuits. The conditional probabilities to reach state when a supply outage happens, assuming a probability of 95% of appropriate operation of the protection system of loads are: from alert to normal state 62.17%, from emergency to normal state 35.96%, and from emergency to extreme emergency 1.87%.

| Place | Description |
|---|---|
| P1-R1 | Normal State. System on R1 |
| P2-G1S | Generator standby available |
| P3-SR1 | Alert State - Outage R1 |
| P4-TR | Transfer system available |
| P5-G1S1 | Emergency State. G1starting |
| P6-STR1 | Extreme Emergency. Transfer system unavailable |
| P7-STR2 | Extreme Emergency. Transfer system don't work |
| P8-G1 | Normal State. System on G1 |
| P9-SG1 | Extreme Emergency. G1 don't start |
| P10-SG2 | Extreme emergency. Fault on G1 |
| P11-R1S | Emergency State. R1 starting |

Table 5. Places for the PN Representation of the Transfer System

| Transition | Description |
|---|---|
| T1 | Power in R1 is interrupted |
| T2 | Transfer works on demand |
| T3 | Transfer don't work on demand |
| T4 | Transfer out of service. Maintenance |
| T5, T13 | Transfer repaired |
| T6 | G1 starts |
| T7 | G1 don't start |
| T8 | R1 returns |
| T9 | Transfer returns on Normal state |
| T10 | G1 don't work by fault |
| T11, T12 | G1 repaired |

Table 6. Transitions for the PN Representation of the Transfer System

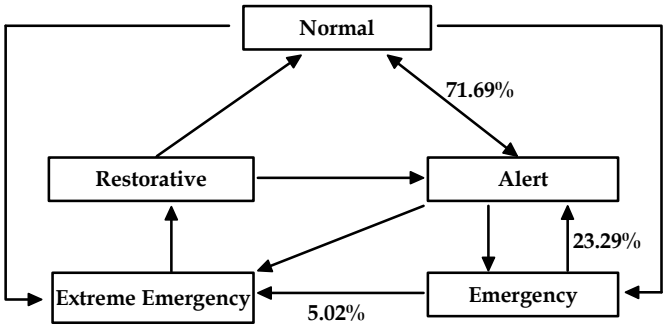



Fig. 12. Probabilities of Operatin States for Transfer System

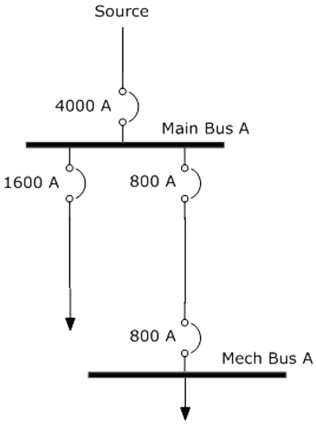



Fig. 13. Load System Scheme IEEE 493 Test Systen

Table 7 lists the places or system states that can be reached by the system when an outage occurs and Table 8 presents the states' transitions. Table 7 and Table 8show the operating states transitions. Hence, the normal operating states are P1 and P8 in the Petri model, alert state is P3, emergency states are P5 and P11, extreme emergency states are P6, P7, P9 and P10, and restorative states are equivalent to extreme emergency states.

| Place | Description |
|---|---|
| P1 | Normal State |
| P2 | Faulted System between Main Bus A and Bus A |
| P3 | Faulted System between Main Bus A and Mech Bus A |
| P4-Emerg | Faulted System between Utility system and Main bus A |
| | Emergency state by non-operation of Bus A or mesh |
| | bus relay |
| P5 | Bus A protection available |
| P6 | Alert State - Fault isolated by Bus A protection |
| P7 | Mech Bus A protection available |
| P8 | Alert State - Fault isolated by Mech Bus A protection |
| P9 | Main protection available |
| P10-extreme | Extreme emergency State by non-operation of main relay |
| P11 | Emergency State–Fault isolated by main protection |
| P12 | Restorative State–system in reparation for Bus A |
| P13 | Restorative State–system in reparation for Main Bus A |
| P14 | Restorative State–system in reparation for Mech Bus A |

Table 7. Places for the PN Representation of the Load System

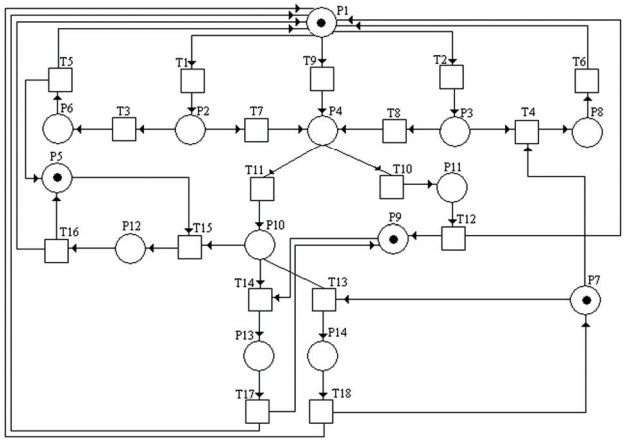



Fig. 14. Petri Net for Load System

| Transition | Description |
|---|---|
| T1 | Fault in Bus A occurs |
| T2 | Fault in Mech Bus A occurs |
| T3 | Fault clearance by Bus A protection |
| T4 | Fault clearance by Mech Bus A protection |
| T5 | Bus A protection closes breaker |
| T6 | Mech Bus A protection closes breaker |
| T7 | Bus A protection doesn't operate |
| T8 | Mech Bus A protection doesn't operate |
| T9 | Fault in Main Bus A occurs |
| T10 | Fault clearance by Main protection |
| T11 | Main Bus A protection doesn't operate |
| T12 | Fault clearance by Main Bus A protection |
| T13 | Manual operation Mech Bus A protection |
| T14 | Manual operation Main Bus A protection |
| T15 | Manual operation Bus A protection |
| T16 | Fault between Main Bus A and Bus A is eliminated |
| T17 | Fault between Utility and Main Bus A is eliminated |
| T18 | Fault between Main Bus A and Mech Bus A is eliminated |

Table 8. Trasitions for the PN Representation of the Load System

Fig. 15 shows the complete PN for the supply analysis of the IEEE 493 system, which is elaborated from Fig. 11 and Fig. 14 with an appropriate renumbering of these states.
The connection between these PN's is established by means of modeling and analysis of contingencies and the protection coordination among two subsystems, and the transfer system response. Then, each place is classified as one operating state: normal, alert, emergency, extreme emergency, and restorative. In this way, the simulation computes the probability of occurrence of each place when a fault in the power system occurs, and in consequence, the security indicators are computed. A 10000 probabilistic trials simulation has been made on the PN assuming a probability of 95% of appropriate operation of the main and backup protection devices, and for the automatic transfer between generators.

That number of trials satisfies an error lower than 5% with a 95% of confidence level. Then, for each trial, the token is moved through the system states after transitions fire. The activation of transitions is taken into account when a decision between conflicting transitions must take place. The conditional probabilities to reach the normal, emergency and extreme emergency status are: from alert to normal state 62.17%, from emergency to normal state 35.96%, and from emergency to extreme emergency 1.87%. The system will be in secure states in 87.43% when a fault (short circuit) occurs in the system.

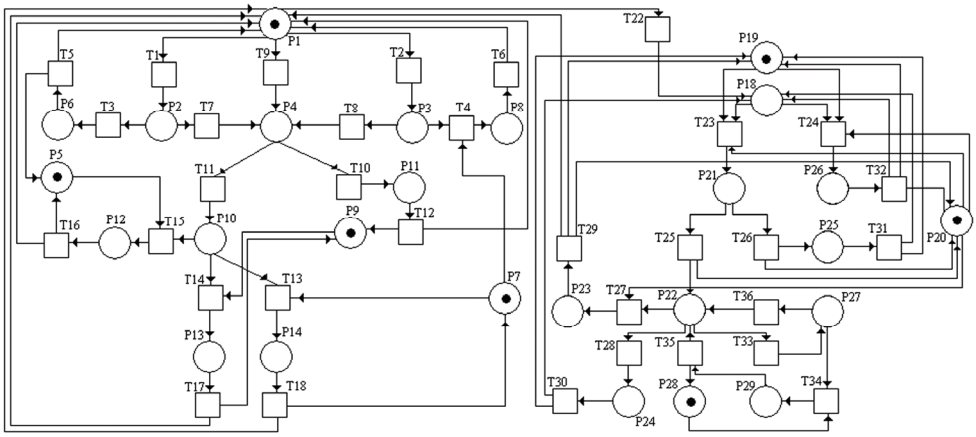


Fig. 15. Petri Net for Total System

### 4.2 Looped Power Systems

A looped power system with 4 buses, with 8 possible faults, each one at the begin and end of each line. Also, this system has 8 sets of protection. The system is shown in Fig. 16 and the Sub-Petri Net modeled is presented in Fig. 17, which represents a fault in line 1. This model is similar to the other line faults, and all Sub-PN are connected.

After simulating the complete Petri Net, 95.2% of failures were cleared by the main protections and 4.56% of failures were isolated by back-up protections. It was expected that almost 5% of cases correspond to the emergency state, because all protections are taken with the same configuration and failure probabilities.

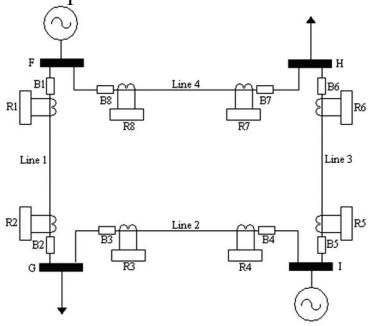


Fig. 16. Four Buses-system

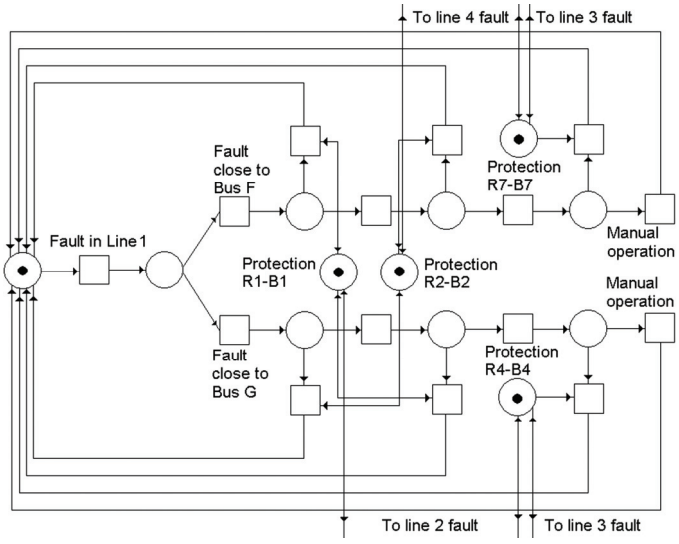



Fig. 17. Sub-Petri net for 4 Buses System

### 4.3 Substations

Substations have always been modeled as a single bus bar (Caro & Rios, 2008) and (Ramos et al., 2009). Furthermore, all the circuits connected to the substation trigger when a system fault is simulated. Nevertheless, there are many different configurations in substations, e.g. single bus, ring bus and breaker-and-a-half bus (Mc Donald, 2007). So, when a fault occurs in the system, not all the protections trigger. Therefore, cascade events studies are incomplete without considering different configurations of substations. Therefore, it is necessary to develop a tool that can be able to evaluate and analyze in detail the effect of protections in substations with different configurations, and also able to measure the impact of the protections over the Power System.

#### 4.3.1 Single Bus

The first configuration studied corresponds to the Single Bus Substation, shown in Fig. 18. In this configuration, the circuits are connected to the Bus through a single switch. The main disadvantage of this configuration is the lack of reliability, security, and flexibility (Mc Donald, 2007). The faults evaluated are: Fault on Bus and Fault on Line 4, and the Line 1 was selected to evaluate the probability of occurrence of a Cascade Event.

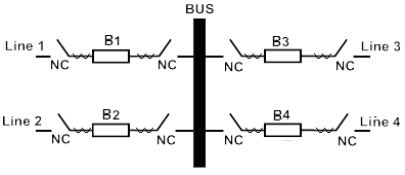



Fig. 18. Single Bus Substation Configuration

Fig. 19 presents the Petri Net for the Fault on Bus in the Single Bus Substation configuration. Table 9 and Table 10 show the places and transitions of the PN, respectively.

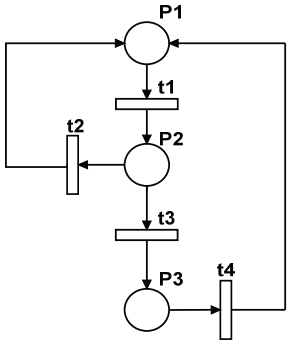



Fig. 19. Petri Net Representation for a Bus Fault on Single Bus Configuration

| Place | Description |
|---|---|
| P1 | Normal operation |
| P2 | Fault on Bus |
| P3 | Fault on Line 1 |

Table 9. Places for a Bus Fault on Single Bus Configuration

| Transition | Description |
|---|---|
| t1 | Fault on Bus |
| t2 | R1 triggers correctly |
| t3 | R1 failure |

Table 10. Transitions for a Bus Fault on Single Bus Configuration

After simulate the PN presented in Fig. 19, the fault leads a cascade event on Line 1 the 5% of the cases. It can be seen that because there is only one bus, the fault has only one way in order to lead an outage in line 1. This confirms the lack of flexibility, and reliability of the single-bus configuration.

#### 4.3.2 Breaker-and-a-half Bus

The breaker-and-a-half bus configuration offers a flexible operation, and high reliability. Also, this configuration allows to isolate any breaker for maintenance without service disruption (Mc Donald, 2007). However, it presents a more complicate relaying, because the center breaker has to act on faults for either of the two circuits . Fig. 20 shows a diagram of this configuration.

The PN for a Line 4 fault in the Breaker-and-a-half bus configuration is shown in Fig. 21. Table 11 and Table 12 show the places and transitions of the PN, respectively.

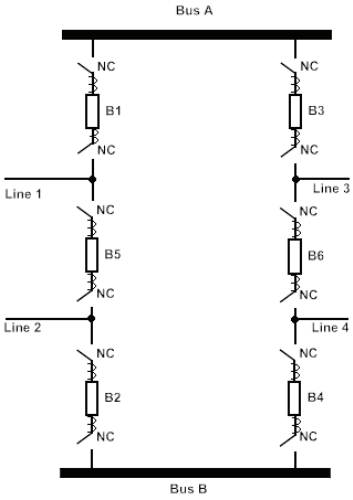



Fig. 20. Breaker-and-a-half Configuration

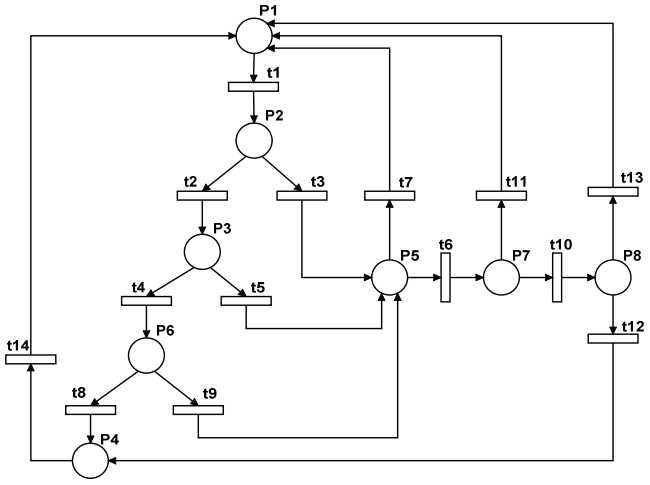



Fig. 21. Petri net Representation for Line 4 Fault on Breaker-and-a-half Configuration

After simulate the PN presented in Fig. 21, the fault leads a cascade event on Line 1 the 0.04% of the cases. According to this result, it is clear that this configuration has a high reliability, as well as high security and adequacy.

### 4.4 Large Power Systems

For large power Systems, it is proposed the use of building blocks theory, which is basically to break up large systems into smaller units that are easier to handle (Dale et al., 2004) . For instance, if the main goal is to model a system composed by two subsystems, the problem

solution should be to model each subsystem separately. Then, combine the solutions together and model the complete system.

| Place | Description |
|---|---|
| P1 | Normal operation |
| P2 | Fault on Line 4 |
| P3 | Fault on Bus B |
| P4 | Fault on Line 1 |
| P5 | Fault on Line 4 |
| P6 | Fault on Line 2 |
| P7 | Fault on Line 3 |
| P8 | Fault on Bus A |

Table 11. Places for Line 4 Fault on Breaker-and-a-half Bus Configuration

| Transition | Description |
|---|---|
| t1 | Fault on Line 4 |
| t2 | R4 failure |
| t3 | R4 triggers correctly |
| t4 | R2 failure |
| t5 | R2 triggers correctly |
| t6 | R6 failure |
| t7 | R6 triggers correctly |
| t8 | R5 failure |
| t9 | R5 triggers correctly |
| t10 | R3 failure |
| t11 | R3 triggers correctly |
| t12 | R1 failure |
| t13 | R1 triggers correctly |
| t14 | Restorative |

Table 12. Transitions for Line 4 Fault on Breaker-and-a-haf Bus Configuration

So, in order to model the connection of substations with different configurations. (Ramos et al., 2009) and (Sánchez et al., 2008) suggest the use of High-level Petri Nets in order to model larger Power Systems. However, the software that simulates High-Level Petri Nets with uncertainties does not exist. Thus, we proposes the use of building blocks, which is basically to break up large systems into smaller units that are easier to handle. For instance, if the main goal is to model a system composed by two substations, the problem solution should be to model each substation separately. Then, combine the solutions together and model the complete system.

#### 4.4.1 Modelling two interconnected substations

Fig. 22 shows a Petri Net to model the effects of a substation on another substation. Transition 1 (t1) fires when there is a fault in the substation 1, e.g. fault on main bus. Transitions 2 and 3 (t2 and t3) are related to the security index for the substation 1, assessed in the first step, that is, the probability of occurrence of a cascade event through the line that connects both substations. Likewise, transitions 4 and 5 (t4 and t5) are related to the security index for the substation 2 assessed in the first step.

A similar diagram can be obtained for multiple connections and/or multiple substations, according to the size of the system to model.
Table 13 shows the results for two connected substations, according to Fig. 22, taking into account that the first event is a fault on the main bus of the substation. On the other hand, Table 14 shows the probabilities of operating states of two connected substations, with fault on a Line as the initiating event.

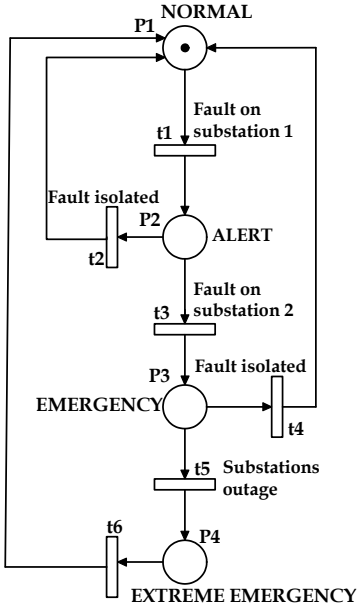



Fig. 22. Petri Net for modeling two connected substations

| Configuration | Alert | Emergency | Extreme Emergency |
|---|---|---|---|
| Single Bus with Tie-Breaker | 0.9500 | 0.0498 | 0.0002 |
| Main and Transfer Buses | 0.9970 | 0.0025 | 0.0002 |
| Single Breaker - Double Bus | 0.9500 | 0.0498 | 0.0002 |
| Ring Bus | - | - | |
| Breaker-and-a-Half bus | 0.9974 | 0.0025 | 0.0001 |
| Double Breaker - Double Bus | 0.9496 | 0.0478 | 0.0026 |

Table 13. Operating State Probabilities of Two Connected Substations Fault on Main Bus

| Configuration | Alert | Emergency | Extreme Emergency |
|---|---|---|---|
| Single Bus with Tie-Breaker | 0.9974 | 0.0024 | 0.00020 |
| Main and Transfer Buses | 0.9974 | 0.0024 | 0.00020 |
| Single Breaker - Double Bus | 0.9974 | 0.0024 | 0.00020 |
| Ring Bus | 0.9498 | 0.0477 | 0.00025 |
| Breaker-and-a-Half bus | 0.9996 | 0.0003 | 0.00010 |
| Double Breaker - Double Bus | 0.9490 | 0.0484 | 0.00026 |

Table 14. Operating State Probabilities of Two Connected Substations - Fault on Line

## 5. Conclusions

This chapter has proposed a methodology of security assessment of Power Systems based on Petri Nets Theory. The methodology not only proposes to model the operating sequence of protection devices, but also proposes the modeling of uncertainty in the operation of protection devices using GSPN, and it proposes to measure its impact on the security assessment. Thus, it is proposed the establishment of a relationship between the operating states of Power Systems and the Petri Net models' places, in order to determine the Petri Net Places that model the non-secure operating states.

This chapter has shown that Petri Networks theory is useful tool for assessing security indexes for Substations. So, the proposed technique allows the security analysis of substations in Power Systems, taking into account hidden failures, unreadiness, and the sequence of operation in protective systems.

The proposed method of building blocks is very effective when it is required to study large Power Systems. Furthermore, the combination of Petri Networks, building blocks, and operating states develop a strong tool to analyze and study the impact of protective systems on the Power System, in terms of adequacy, security, and reliability.

It was demonstrated that the cascade events studies are incomplete without considering different configurations of substations. Additionally, it was demonstrated the importance of selecting the fault to be analyzed, because each fault brings out different failures probabilities.

The Generalized Stochastic Petri Networks (GSPN) is a rigorous mathematical technique for the electrical systems security analysis that allows the modeling and simulations of system with a large number of states. On the other hand, the modeling capabilities of GSPN allow the inclusion of models of time transitions between states, different to the exponential distributions.

Simulation results on the GSPN of power systems give an easy and intuitive interpretation of the probability of keeping each operating state of the system. Therefore, these state probabilities complement traditional reliability system indicators, assisting to understand the effect of a fault element on system operation.

The numerical results are consistent with expectations, since for all protections it is assumed equal failure probabilities. For this reason, the results would change if there are used different failure probabilities for each protection.

The future application of Petri Nets for assessment of larger Power Systems requires the use of High-level Petri Nets, e.g. Colored Petri Nets. It is proposed as further work, to develop a graphical software to model High-level Petri Nets with uncertainties. Likewise, it is proposed to apply this methodology to larger Power Systems, e.g. IEEE 118 nodes.

## 6. References

Anderson, P. (1999). *Power System Protection*. John Wiley and Sons.

Anderson, P., & Agarwal, S. (1992). An improved model for protective system reliability. *IEEE Transactions on Reliability* , No. 41, pp. 422-426.

Billinton, R., Fotuhi-Firuzabad, M., & Aboreshaid, S. (1997). Power System Health Analusis. *Reliability Engineering and System Safety* , Vol. 55, No.1, pp. 1-8.

Caro, M. A., & Rios, M. A. (2008). Super components contingencies for Severity Analysis of Power Systems. *IEEE/PES Transmission and Distribution Conference and Exposition, Latin America*, (p. 6). Bogota.

Dale, N. B., Weems, C., & Headington, M. R. (2004). *Programming and problem solving with C++* (4th Edition ed.). Jones & Bartlett Publishers.

Dong, Z., Koval, D., & Propst, J. (2003). Reliability of Various Industrial Substations. *IEEE Transactions on Industry Applications* , Vol. 40, No. 4, pp. 989-994.

EPRI. (1987). *Composite-System Reliability Evaluation: Phase I - Scoping Study EL-5290*. EPRI.

Jenkins, L., & Khincha, H. P. (2006). Deterministic and Stochastic Petri Nets models of protection Schemes. *IEEE Transactions on Power Delivery, Vol. 25*, No. 1. , Vol. 7, No.1, pp. 84-90.

Koval, D. O., Jiao, L., & Arno, R. G. (2002). Zone-branch reliability methodology applied to Gold Book standard network. *IEEE Transactions on Industry Applications* , Vol. 38, No. 4, pp. 990-995.

Koval, D. O., Xinlie, Z., & Propst, J. (2003). Reliability methods applied to The IEEE Gold Book Standard Network. *IEEE Industry Applications Magazine* , Vol. 9, No. 1, pp. 32-41.

Kundur, P. (1994). *Power System Stability and Control*. New York: McGraw-Hill Professional.

Lester, F., & Carlsen, K. (1978). Operating under stress and strain. *IEEE Spectrum* , Vol. 15, No. 3, pp. 48-53.

Matcovschi, M.-H., Mahuela, C. & Pastranavu, O. (2005) Learning about Petri Net Toolbox. Iasi, Romania: Technical University "Gh Asachi", Vol. 2.1.

McDonald, J. D. (2007). Electric power substations engineering. In L. L. Grigsby, *Electric power engineering handbook* . Boca Raton: CRC Press.

Murata T. (1989) Petri nets: Properties, analysis and applications. *Proceedings of the IEEE.* Vol. 77, No. 4. pp. 541-580.

NERC. (2009). 2008 Annual Report.

Dobson I. & McCalley J. (2008). *Risk of Cascading Outages*. Iowa State University, PSERC.

Ramos, G., Sánchez, J. L., Ríos, M. A., & Torres, Á. (2009). Power Systems Security Evaluation using Petri Nets. *IEEE Transactions on Power Delivery.*

Sánchez, J. L., Ramos, G., & Ríos, M. A. (2008). Modeling of operative sequences of protections in Power Transmission systems using Petri Nets. *IEEE/PES Transmission and Distribution Conference and Exposition: Latin America* .

Wang, J. (1998). *Timed Petri Nets, theory and applications* (Vol. 1st Edition). Ed. Wiley.

# Fault Diagnosis on Electric Power Systems based on Petri Net Approach[1]

Alejandra Santoyo-Sanchez, José Alberto Gutiérrez-Robles,
Elvia Ruiz-Beltrán, Carlos Alberto De Jesús–Velasquez,
Luis Isidro Aguirre-Salas and Víctor Ortiz-Muro
*Universidad de Guadalajara*
*Instituto Tecnológico de Aguascalientes, INTEL*
*México*

## 1. Introduction

Fault diagnosis is a process that identifies and locates faults occurrence in systems using their inputs, outputs, and their structures. In the context of electric power systems fault diagnosis is realized using the voltages and currents measurements at each node and the physical connections. In this process, there are two conventional important problems that must be solved: 1) how to determine whether a system is diagnosable and 2) how to design a diagnoser. The problem of fault diagnosis has been addressed through various approaches and application methods, for example based on artificial intelligence techniques, and based on Discrete Event Systems. In the last approach, Finite Automata and Petri nets have been mainly used as modelling formalism.

Although the Finite Automata are suitable for Discrete Event Systems, its application is limited to small systems, since the models should explicitly taken in account all the possible states of the system, resulting in very large models which are very difficult to work with them. In order to cope with the state explosion problem, the structural models, represented as vector Discrete Event Systems or Petri nets allow a compact representation of a Discrete Event Systems, avoiding a large set of states as Finite Automata does. Also, with this formalism, the discrete event systems analysis can be realized using linear algebra methods. Using the advantage of these features, fault diagnosis has been addressed using Petri nets.

Several methods have been proposed recently, for example in (Sheng-Luen et al., 2003) and (Genc & Lafortune, 2003) the Petri nets approach is used, but the diagnosability test has been based on the reachability graph; thus the method is limited to small size systems (Sampath et al., 1995) and (Sampath et al., 1996). In (Lefebvre & Delherm, 2007) a diagnoser based on the Petri nets paths and causality relationships is proposed to determine the presence of faults in a system. In (Hadjicostis & Verghese, 1999a), (Hadjicostis & Verghese, 1999b) and (Hadjicostis & Verghese, 2000) Petri nets model is used in order to introduce

[1] Supported by project PROMEP 103.5/09/1158

redundancy into the system that allow to detect and isolate faults. Unfortunately, all those solutions have the state explosion problem.
In order to cope with the state explosion problem, in (Ramírez-Treviño et al., 2004), (Ramírez-Treviño et al., 2007) and (Ruiz-Beltrán et al., 2007) techniques based on structural characterizations for determining the diagnosability property are proposed; in (Ruiz-Beltrán et al., 2007) is extended the characterizations and the diagnoser structures of (Ramírez-Treviño et al., 2004) and (Ramírez-Treviño et al., 2007), but that result is limited for binary Interpreted Petri nets. In these approaches, the diagnosability property is tested as a linear programming problem with polynomial complexity. In (Sheng-Luen et al., 2003), (Ulerich & Prowers, 1998), and (Yang et al., 2004) some applications of fault diagnosis using Petri nets are proposed, but they do not consider the diagnosability analysis of the system, which do not guarantee that a fault can be detected in a finite number of events when this fault occur. However, this condition is fulfilled in (Ramírez-Treviño et al., 2004), (Ramírez-Treviño et al., 2007), and (Ruiz-Beltrán et al., 2007).
Recently, the model-based approach for fault diagnosis in Discrete Event Systems is widely been worked by the research community of Discrete Event Systems. Several works on the matter use Petri nets for performing the fault diagnosis in electric power systems (Hadjicostis & Verghese, 1999a), (Hadjicostis & Verghese, 2000), (Ren et al., 2004), and (Ren & Zengqiang, 2006). In (Hadjicostis & Verghese, 1999a) and (Hadjicostis & Verghese, 2000) the authors propose coding theory techniques for detecting and locating failures in Discrete Event Systems in the field of electric power systems, they build a monitoring Petri net, which operates concurrently with the power network. However, for some failures, their techniques can not give a right explanation, and they do not determine how to construct the monitoring matrix. In (Ren et al., 2004), (Ren & Zengqiang, 2006) and (Proth et al., 1993) the same method is further studied. All possible failures are analyzed and remapped to the embedded Petri net models and a method of how to construct the monitoring matrix to encode the former Petri net model is proposed. However, the previous works do not determine the diagnosability of the electrical systems, i.e., it can not be determined off-line if a fault can be detected and located in a finite number of steps.
Based on the voltages and currents measurements and its digital processing with a relay to maintain the operation of the power electric system; in this chapter we will address: 1) how the operational behaviour of power electrical system with Interpreted Petri Net is represented, where the Interpreted Petri Net captures the normal and fault condition of the power electrical system. 2) How it is determined whether the model of the power electric system with Interpreted Petri Net is diagnosable and 3) how it is designed a diagnoser that has the same operative characteristics (security and velocity) that a relay in a real case. Thus, the results in (Ramírez-Treviño et al., 2004), (Ramírez-Treviño et al., 2007), (Ruiz-Beltrán et al., 2007) and (Santoyo-Sanchez et al., 2008) will be extended.
This chapter will be as follows. Section 2 will introduce fault features in electric power systems, how the electric power systems are modelled, and how the fault diagnosis is executed into the context of electric power systems. In the next section will be introduced a review Petri net notation and concepts used in the chapter. Section 4 will introduce the diagnostic from the Petri net point of view; in this case it is proposed a necessary condition to determine if a power electrical system is diagnosable. Moreover, it will be presented a method for designing a diagnoser, and how the diagnostic can be done. Next, in section 5

the results will be illustrated with a case study. Finally, in section 6 conclusions and future research will be given.

## 2. Fault modeling conditions in electric power systems

The analysis of the power electrical systems involve three stages namely, pre-fault condition (steady-state operation), fault condition (transient condition) and post-fault condition (how the network arrives to a new steady-state operation condition). If the relay is the equipment to protect the entire system, so it is necessary to know the exact relay behaviour under each stage. The tests of the relay performance are doing by off-line simulations; this reason renders necessary the modelling of the network components, the digital processing of the electrical signals and the procedure to take a logical decision: order a trip in a fault condition, do nothing in a safe condition. This decision is processed sample by sample. The relay behaviour, the used models to simulate the power electrical system network and the signal processing are described in order to clarify why and how Petri Nets are used to increment or to give redundancy to the protection schemes.

### 2.1 Relay behavior

The distance elements are the most widely used to protect transmission lines. Usually, they shared a communication channel to implement pilot schemes which increases the overall performance of the line protection. Each distance element has a boundary of the zone of protection in the locations of the currents transformers. The other boundary is extended in the forward direction and is determined by measurements of the system quantities that may vary with generation and configuration schemes. One of the most important design considerations of relaying is security. It is a measure of the relay's ability to avoid operation for all other conditions for which tripping is not desired. It is hard to achieve, and almost infinite variety of tests would be needed to simulate all possible conditions to which a relay may be exposed. Another important design consideration is protection velocity. The fault clearance must be as fast as possible for conditions within the transmission line. For distance elements is not a problem in their primary zone, but it is used to be a delay within an extended backup zone. This is shown in figure 1.

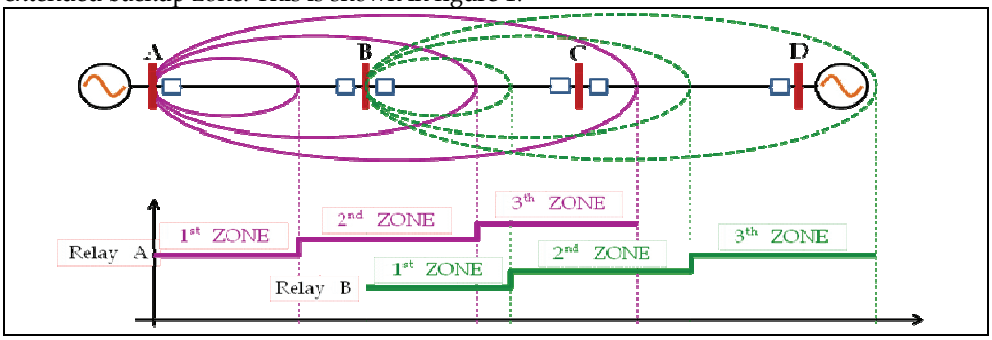



Fig. 1. Relay protection zones

Figure 2a shows the functional diagram of the distance relay. The process begins when the voltages and currents flow to the relay through the potential transformers and currents transformer installed in the power electric system network. These signals are passed through the relay's internal transducers, and then the signals are filtered before the analogical-digital conversion. The analogical filtering has the function of limit the bandwidth, so it is possible to use a low-frequency rate sampling without introduce a big error due to the aliasing phenomenon. The next step is the use of a digital filter to eliminate the superior harmonics and the no-periodic exponential component still remaining; the useful filters are the finite impulse response ones like the Fourier or cosine types.
Phasors of fundamental frequency (60 Hz) are computed by the discrete convolution between the digitalized, voltages (V) and currents (I), measured samples and the coefficients of the digital filter. It is demonstrated that independently of the frequency used rate in the analogical-digital conversion, the most convenient window length of the digital filter is equal to one period of fundamental frequency which means a delay of the trip, to fulfill the window with the fault signal, of one cycle the normal rate of relays is of 32 samples by cycle. The adjust parameters, to model short-circuits, are based in phasors which are used to represent the steady-state operation. Under ideal conditions, the estimate impedance is equal to the line section from the relay to the fault location. Faults provoke transient operation state in the network, so the impedance estimation is affected by the transient state components (fault resistance), this affects the relay response related with the fault detection in the limit zone operation (zone discrimination); for this reason the first zone of the relay is adjust to 80-90% of the protected line. Figure 2b shows the stages of the distance relay procedure and the normal consuming time each stage. Times are normalized with the fundamental frequency. In this figure, it could be notice that the execution of the digital filtering to compute the phasors of fundamental frequency is the most demanding time in the process of the trip signal. For typical filters the delay is one cycle of 60 Hz, the corresponding time is around 16.66 milliseconds. The delay of the digital filter is then a limitation that could not be avoided.

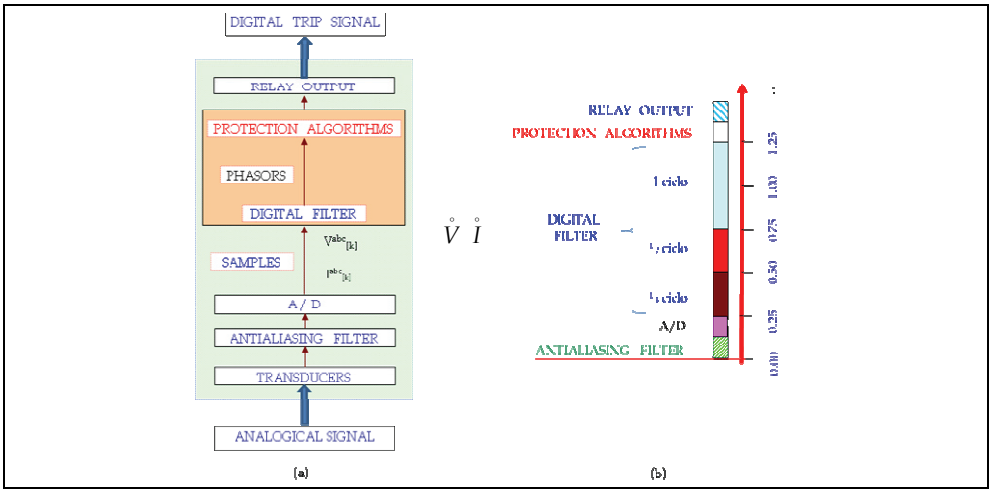


Fig. 2.a) Relay signal processing, b) Relay consuming times

The algorithms of the relay use both, phasors and the modal transformation described in section 2.2 of this document, here called symmetrical components; with this information the relay elaborate its response, the instantaneous one and the off-line one. The first involves the fault detection and the fault zone discrimination. The second is referred to the fault classification and localization.

- The relay detect a fault in its zone protection by comparing an adjust parameter with a function related with the voltages and currents phasors.
- The capacity to discriminate faults in front or behind near to the fault zone, is difficult because the estimate voltage phasor is almost zero; so, the effective and useful method is the polarization of the relay with the positive sequence network of the symmetrical components.
- The fault classification is performing by the symmetrical components as it is shown in table 1 described in section 2.2 of this document.
- The fault localization is computed by comparing the voltage-current relation with the total impedance of the line, so it is expressed in % of the line length.

National Electrical System is characterized by the large distance between the generation plants and the consuming points (charge). For that reason, the National Electrical System is divided into eight strategic located regional centers for operating and coordinating the production and distribution of the electric energy. Figure 3 shows the limits of each control area, also it is showed in this figure the electrical connectivity of the western control area which will be taken like an example, in this work, to implement the develop methodology. The fundamental criteria of reliability and operative security must be satisfied, so it is essential that must exists a good coordination among the operative entities in all operative stages. In this context these systems of energy generation have diverse exploration braches of study, among them, the maintenance area that as important factor as for costs, where innovating techniques of fault diagnosis for a later plan of reliable maintenance are required.

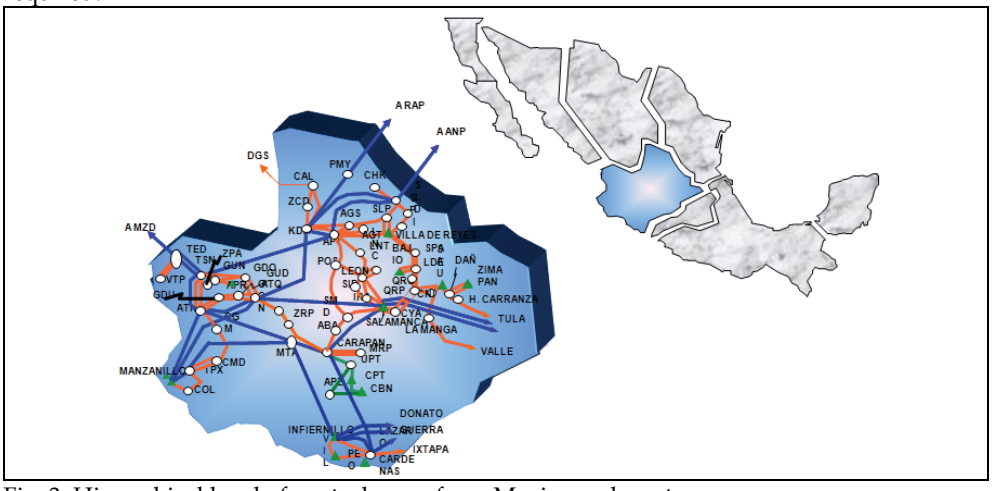



Fig. 3. Hierarchical level of control areas from Mexico and western area

### 2.2 Modeling the electrical system

The electrical systems are composed by generators that are interconnected through transmission lines to loads, sensors, capacitors, switches, substations, and transformers with the objective to produce and to distribute electrical energy. In order to simulate and analyze the electrical system behavior, the minimum models of elements of the system are: A) lines, B) sources and C) charges; which are brief described below. Figure 4 shows a piece of three phase line connected with a three phase source and a single charge each phase.

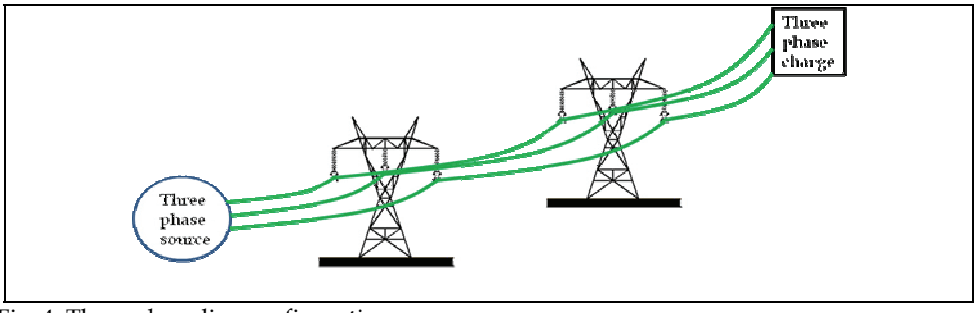



Fig. 4. Three phase line configuration

A) LINES

A rigorous analysis of the transmission line with losses is obtained by the Maxwell equations after some simplifications concerning to the electromagnetic field distribution. These simplifications conduced to the quasi-transversal electromagnetic fields. So, the line electromagnetic response of a transmission line is modeled by the Transmission Line Telegrapher's Equations as:

$$-\frac{\partial \mathbf{V}(x,t)}{\partial x} = \mathbf{R}(x)\mathbf{I}(x,t) + \mathbf{L}(x)\frac{\partial \mathbf{I}(x,t)}{\partial t} \tag{1}$$

$$-\frac{\partial \mathbf{I}(x,t)}{\partial x} = \mathbf{G}(x)\mathbf{V}(x,t) + \mathbf{C}(x)\frac{\partial \mathbf{V}(x,t)}{\partial t} \tag{2}$$

where $\mathbf{L}(x)$, $\mathbf{R}(x)$, $\mathbf{C}(x)$ and $\mathbf{G}(x)$ are the line per unit-length inductance, resistance, capacitance and conductance matrices, respectively.

The solution of these equations in frequency domain proceeds as follows; one has firstly the solution of the voltage and current like function of time and space as:

$$\mathbf{V}(x,t) = \mathbf{V'}(x,t)e^{j(\omega t+\varphi)} \qquad \mathbf{I}(x,t) = \mathbf{I'}(x,t)e^{j(\omega t+\varphi)} \tag{3}$$

If one has, $\mathbf{V}(x) = \mathbf{V'}(x)e^{j\varphi}$ and $\mathbf{I}(x) = \mathbf{I'}(x)e^{j\varphi}$ then;

$$\mathbf{V}(x,t) = \mathrm{Re}\left(\mathbf{V}(x)e^{j\omega t}\right) \qquad \mathbf{I}(x,t) = \mathrm{Re}\left(\mathbf{I}(x)e^{j\omega t}\right) \tag{4}$$

Substituting equation (4) into equation (1) and (2) yields to:

$$\frac{\partial\left(\mathrm{Re}\left(\mathbf{V}(x)e^{j\omega t}\right)\right)}{\partial x} + \mathbf{L}(x)\frac{\partial\left(\mathrm{Re}\left(\mathbf{I}(x)e^{j\omega t}\right)\right)}{\partial t} + \mathbf{R}(x)\left(\mathrm{Re}\left(\mathbf{I}(x)e^{j\omega t}\right)\right) = 0 \tag{5}$$

$$\frac{\partial\left(\mathrm{Re}\left(\mathbf{I}(x)e^{j\omega t}\right)\right)}{\partial x} + \mathbf{C}(x)\frac{\partial\left(\mathrm{Re}\left(\mathbf{V}(x)e^{j\omega t}\right)\right)}{\partial t} + \mathbf{G}(x)\left(\mathrm{Re}\left(\mathbf{V}(x)e^{j\omega t}\right)\right) = 0 \tag{6}$$

Solving these equations, one has:

$$\mathrm{Re}\left( e^{j\omega t} \frac{\partial \mathbf{V}(x)}{\partial x} \right) + \mathrm{Re}\left( \mathbf{L}(x)\mathbf{I}(x) \frac{\partial \left(e^{j\omega t}\right)}{\partial t} \right) + \mathrm{Re}\left(\mathbf{R}(x)\mathbf{I}(x)e^{j\omega t}\right) = 0 \tag{7}$$

$$\mathrm{Re}\left( e^{j\omega t} \frac{\partial \mathbf{I}(x)}{\partial x} \right) + \mathrm{Re}\left( \mathbf{C}(x)\mathbf{V}(x) \frac{\partial \left(e^{j\omega t}\right)}{\partial t} \right) + \mathrm{Re}\left(\mathbf{G}(x)\mathbf{V}(x)e^{j\omega t}\right) = 0 \tag{8}$$

Re-arranging equation (7) and (8) one obtains:

$$\mathrm{Re}\left( \left( \frac{\partial \mathbf{V}(x)}{\partial x} + \mathbf{L}\mathbf{I}(x) j\omega + \mathbf{R}\mathbf{I}(x) \right) e^{j\omega t} \right) = 0 \tag{9}$$

$$\mathrm{Re}\left( \left( \frac{\partial \mathbf{I}(x)}{\partial x} + \mathbf{C}\mathbf{V}(x) j\omega + \mathbf{G}\mathbf{V}(x) \right) e^{j\omega t} \right) = 0 \tag{10}$$

So, the real part is:

$$-\frac{\partial \mathbf{V}(x)}{\partial x} = \left(j\omega \mathbf{L} + \mathbf{R}\right)\mathbf{I}(x) \tag{11}$$

$$-\frac{\partial \mathbf{I}(x)}{\partial x} = \left(j\omega \mathbf{C} + \mathbf{G}\right)\mathbf{V}(x) \tag{12}$$

If $\mathbf{Z} = \left(j\omega \mathbf{L} + \mathbf{R}\right)$ and $\mathbf{Y} = \left(j\omega \mathbf{C} + \mathbf{G}\right)$, finally one obtains:

$$-\frac{\partial \mathbf{V}(x)}{\partial x} = \mathbf{Z}\mathbf{I}(x) \tag{13}$$

$$-\frac{\partial \mathbf{I}(x)}{\partial x} = \mathbf{Y}\mathbf{V}(x) \tag{14}$$

where $\mathbf{Z}$ represents the impedance series matrix, $\mathbf{Y}$ represents the admittance in derivation, both in per unit length and calculated like a linear function of the frequency (jω).

Treating the line like a simple circuit means neglecting the traveling time by taking into account the line length in the phase angle of each node (phase representation), one obtains the following equation (from equation (13)):

$$\mathbf{V}_{\phi} = \mathbf{Z}\mathbf{I}_{\phi} \tag{15}$$

For a geometric configuration of a common three phase horizontal line, if the line is constructed with the same material and has equal radius; additionally if it is positioned at a same height respect to ground and by assuming some simplifications like: equal coupling between conductors and homogeneous earth resistivity, it will conduces to the special case which has the following symmetry:

$$\mathbf{Z} = \begin{bmatrix} Z_s & Z_m & Z_m \\ Z_m & Z_s & Z_m \\ Z_m & Z_m & Z_s \end{bmatrix} \tag{16}$$

This work makes special emphasis on the uses of the diagonalization methodology proposed in (Naredo et. al.,1987) to handle three phase systems like single ones. Using a modal transformation, for both voltage and current, one has:

$$\mathbf{V}_{\phi} = \mathbf{M}\mathbf{V}_{\mathrm{m}} \qquad \mathbf{I}_{\phi} = \mathbf{M}\mathbf{I}_{\mathrm{m}} \tag{17}$$

where $\mathbf{V}_{\mathrm{m}}$ and $\mathbf{I}_{\mathrm{m}}$ are called modal voltages and currents respectively and $\mathbf{M}$ is the modal transformation matrix.

By substituting equation (17) into equation (15), one obtains:

$$\mathbf{MV}_{\mathrm{m}} = \mathbf{ZMI}_{\mathrm{m}} \tag{18}$$

This process yields to:

$$\mathbf{V}_{\mathrm{m}} = \mathbf{M}^{-1}\mathbf{ZMI}_{\mathrm{m}} \tag{19}$$

It is demonstrated (Naredo 1992) (Strang 1998) (Greenspan D. & Casulli V., 1988 ) that the product $\mathbf{M}^{-1}\mathbf{ZM} = \Lambda$ lead to a diagonal matrix when it is used the self modes (eigenvectors of $\mathbf{Z}$), which is the required condition to treat the coupled three phase system like an uncoupled one. By using symmetrical components to diagonalizable this equation, which means by using

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 \\ a & a^2 & 1 \\ a^2 & a & 1 \end{bmatrix} \text{ and } \mathbf{M}^{-1} = \frac{1}{3}\begin{bmatrix} 1 & a^2 & a \\ 1 & a & a^2 \\ 1 & 1 & 1 \end{bmatrix} \tag{20}$$

then equation (19) is as follows:

$$\begin{bmatrix} V_m^+ \\ V_m^- \\ V_m^0 \end{bmatrix} = \begin{bmatrix} Z^+ & 0 & 0 \\ 0 & Z^- & 0 \\ 0 & 0 & Z^0 \end{bmatrix} \begin{bmatrix} I_m^+ \\ I_m^- \\ I_m^0 \end{bmatrix} \tag{21}$$

where $Z^+ = Z_8 - Z_m$, $Z^{-'} = Z_8 - Z_m$ and $Z^0 = Z_8 + 2Z_m$. Like these, the line coupled model could be solved like an uncoupled one, as:

$$V_m^+ = Z^+ I_m^+, \quad V_m^- = Z^- I_m^- \text{ and } V_m^0 = Z^0 I_m^0 \tag{22}$$

B) SOURCES

The steady state solution yields to the following case, for example, if it is placed an ideal voltage source of 60 Hz, 115 kV line to line, in one end; modal voltages are (by definition $\mathrm{V}_{ab} = \sqrt{3}\mathrm{V}_{an}\angle 30^{\mathrm{o}}$), so one has:

$$\mathrm{V}_{LL} = \begin{cases} 115kV\angle 0^{\mathrm{o}} \\ 115kV\angle 120^{\mathrm{o}} \\ 115kV\angle 240^{\mathrm{o}} \end{cases} \rightarrow \mathrm{V}_{LN} = \begin{cases} 66.4\angle -30^{\mathrm{o}} \\ 66.4\angle 90^{\mathrm{o}} \\ 66.4\angle 210^{\mathrm{o}} \end{cases} \tag{23}$$

The relation $\mathbf{V}_m = \left[\mathbf{M}^{-1}\mathbf{V}_{\phi}^{\text{initial}}\right]$ gives the modal components, so by using Symmetrical Components one obtains:

$$\begin{bmatrix} V_m^+ \\ V_m^- \\ V_m^0 \end{bmatrix} = \begin{bmatrix} \text{Voltage}\angle -30^{\mathrm{o}} \\ 0 \\ 0 \end{bmatrix} \tag{24}$$

So, using equation (24), the complete representation for a three phase network in steady state is solved with the positive sequence network as:

$$V_m^+ = Z_m^+ I_m^+ \tag{25}$$

C) CHARGES

The charges are represented like single impedances connected to the network, which means; by definition the charge is uncoupled so it does not need to be treated by applying the modal transformation.

**FINAL MODEL FOR DIAGNOSE PROCEDURE**

Using the line and the source modal transformations, the three phase coupled network is represented like three single uncoupled lines like it is shown in figure 5. By simple analysis one could note that in a steady state condition the three phase coupled system could be fully represented by the positive sequence network, so, by applying the inverse modal transformation to the solution of this network one obtains the solution of the three phase coupled one. In a fault condition the voltages and currents in the nearby node will have values not only for the positive sequence one, so this procedure could give enough information to diagnose the fault condition on the system and the kind of the fault. This is the normal procedure to diagnose and characterize a fault in a power electric system. So, based on the information of the symmetrical components the faults are detected and classified.

In figure 5, the dashed lines of the sources means that for negative and zero sequences the value is zero in steady state operation. In case of unsymmetrical charge; if the asymmetry is due a natural over charge, the voltage-current relationship (denoted as V and I respectively) in phase domain is healthy, so the relay does not process the asymmetry like a fault condition.

By example, in a three phase faults one does not has negative sequence current, nether zero sequence current. For single faults there are equal currents in the sequence networks, etc. Table 1 presents these and other features of the relay input signal when these are processed with symmetrical components.

| Kind of Fault | Characteristics of V | Characteristics of I |
|---|---|---|
| Three phase | $V(p) \neq 0$, $V(n) = V(0) = 0$ | $I(p) \neq 0$, $I(n) = I(0) = 0$ |
| Two phase | $V(p) \neq V(n) \neq 0$, $V(0)=0$ | $I(p) = I(n)$, $I(0) = 0$ |
| Two phase-ground | $V(p) \neq V(n) \neq V(0) \neq 0$ | $I(p) \neq I(n) \neq I(0)$ |
| Single phase | $V(p) \neq V(n) \neq V(0) \neq 0$ | $I(p) = I(n) = I(0)$ |

Table 1. Features of the relay input signals in symmetrical components.

### 2.3 Fault electrical features

Under the point of view of the relay, the power electrical system have three stages called; pre-fault condition, fault
condition (transient condition) and post-fault condition. In the next paragraphs each one will be briefly described.

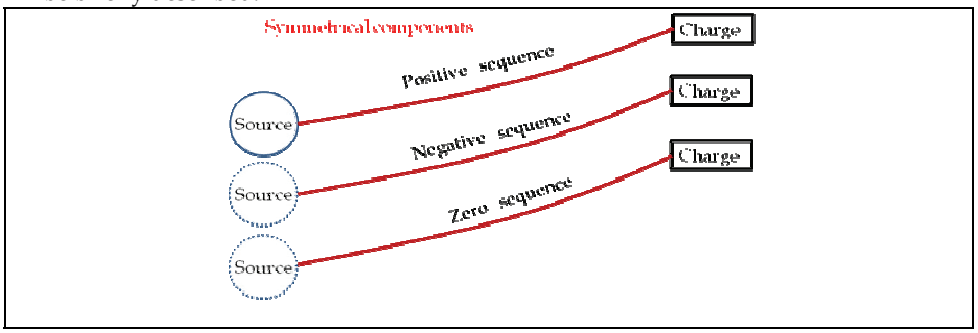



Fig. 5. Model of the three phase line configuration like three single lines.

*Pre-fault condition*.- Each energy control center takes information from the power electrical network (sub-stations) in real time, so it is possible to know and to supervise the operation limits. The operator engineer takes actions to maintain the steady state condition; that means, frequency, voltage limits, maximum power flows in each line, etc. Under this steady-state condition, the entire power electric system works close to their nominal frequency (60 Hz). This value is achievable by automatic action of the automatic generation control, which creates a balance between generated and consumed active power (Fink *et. al.* 1985), (IEEE Std C37.1 – 1994).

The relation between the voltage phasor and the current phasor is linear, so if the voltage is constant (operative condition) and the current grow up from its minimum value to its maximum depending of the demanding energy, the voltage-current relation each phase is out of the fault characteristic zone, which is defined with the relation resistance-inductance of the line. Figure 6 shows the characteristic zones of relay and the trajectory of a typical fault. This trajectory is out of the fault zones in the steady state operation, and it is maintain out of the fault zones for this operative condition.

*Fault condition*.- This is a transient condition phenomena because the protection devices acts as fast as possible to disconnect the faulted element. This is the main purpose of a relay and it is also part of the topic of the present work. This stage involves a number of actions beginning with topological changes, automatically produced by action of local protections. The fault clearing requires the use of substation measurements, relay coordination, and communication systems. This stage ends when protections stop disconnecting those elements that were affected by the fault or situated inside the protection zone. Nowadays during a transient state is common the use of load shedding, blocking schemes, and dedicated controls. Most of these schemes can take action in milliseconds, so these schemes become the first automatic corrective action (Madani *et. al.* 2004), (Guzman *et. al.* 2006). After that, like an off line procedure means after the logical decision of the relay (tripping decision), the relay classifies, locate, typifies and disconnect the fault. In a fault condition, the voltage-current relation in not linear because the inclusion of the fault resistance in the path of the energy, so it is beginning a trajectory to the fault zone. The inclusion into the fault zone is not instantaneously because the computation of the phasor is due with a cycle and all the samples but one are healthy. Sample by sample the data window to compute the phasor is fully with fault data, this defines a fault trajectory (L-R-t trajectory). Figure 6 shows the behavior of this trajectory for a typical fault.

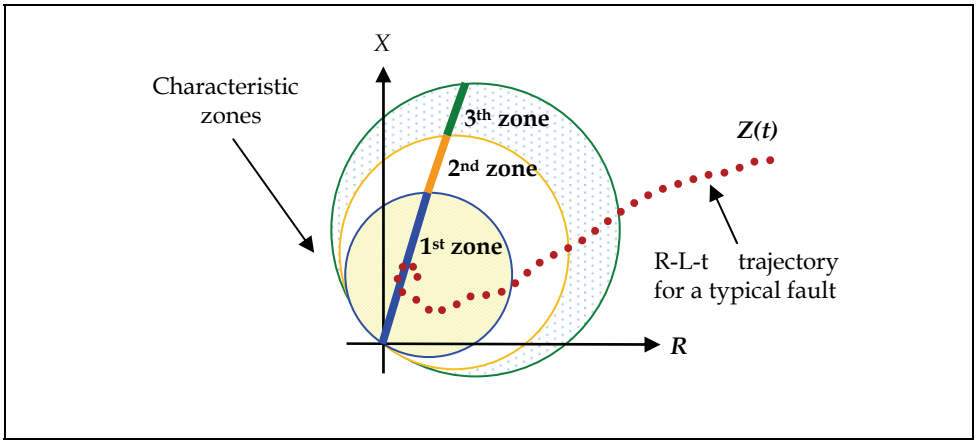



Fig.6. Characteristic zones of a relay

A fault classify algorithm identifies the kind of fault being based in the information of the modal transformation. By example, in a three phase faults one does not has negative sequence current, nether zero sequence current. For single faults there are equal currents in the sequence networks, etc. Note that table 1 presents these and other features of the relay input signal.

Physically, the line could present very complex faults, although protection is develop only to four groups of faults: a) three phase faults, b) two phase faults, c) two phase faults to ground and d) single faults. Figure 7 represents the faults statistical of one year in the Mexican Western Transmission Network. The RELIEVE was consulted to extract the events of the period with a confirmed fault. This system confirms that 69% were single faults, 19% two phase faults and 3% three phase faults.

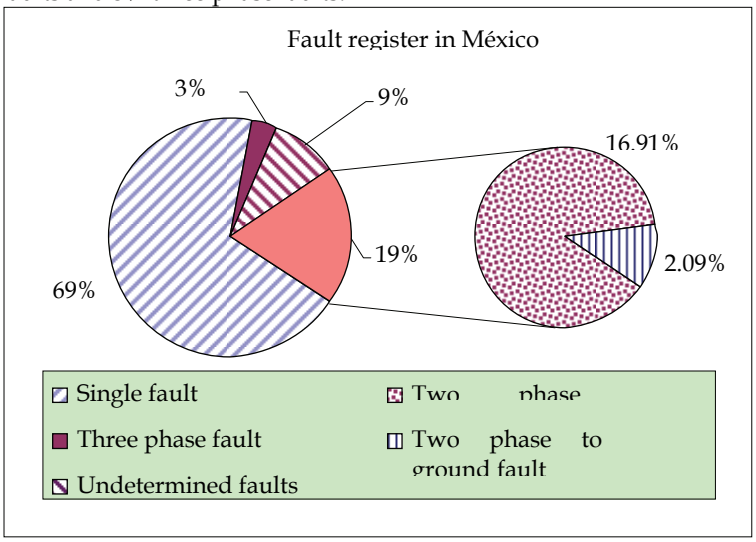



Fig. 7. Percent of faults in the MWTN in 2006

*Post-fault condition.-* Once the faulted elements are disconnected, the entire network arrives to a different steady state operation condition. The operator engineer takes corrective actions to reach the normal operation condition. Normally, he leads the network to the new steady state without losing more electrical elements, which means, he maintains the network as healthy as possible. This state can remain for several seconds or minutes until to reach another steady state normal condition, in which, there exist a balance between charge and generation.

### 2.4 Diagnosis Example

For illustrate the diagnoser behavior in power electrical systems consider the power electrical network of the figure 8. In a steady state operation, $I^{abc}$ measurements in each relay don't overpass the adjust pick up. In this specific point of operation the entire network is healthy, that mean, the electrical system is working normally. Figure 8 shows a simple network in a steady state condition, this figure shows how the power flows from generator 1 and 2 to the charge.

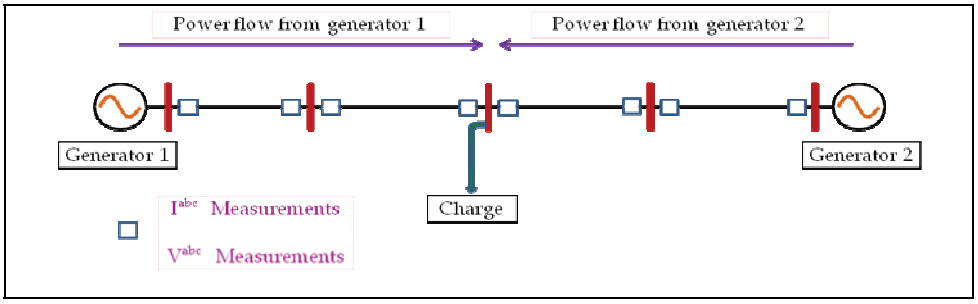



Fig. 8. Steady state operation

While the information of energy control center from the power electrical network (sub-stations) is into the operation limits, the operation of the electrical network is mantanied. When a fault occurs, it is established a trajectory from the generators to the fault point, this is shown in figure 9. In this case, the fault is in the closer line to the generator 2. Analyzing the fault from the point of view of the generator 1, there are four relays that detect the fault in from of them. All relays process the signal as it is shown in figure 2 (figure with the operative times), then each determine is fault zone (figure 6 has the relay circles). Like a resulting of this process, each relay determines its tripping time. The closer relay to the fault detects the fault at the first zone so acts instantaneously, the next in the second one so acts with a delay and the next in the third one so act with an additional delay, the last one (and so on in case of more relay) see the fault out of the protection zone for which they were programmed. There are some relays that detects the fault on behind, so these relays don't act. From generator 2, there is only one relay which determines the fault location at the first zone, so this relay acts instantaneously.

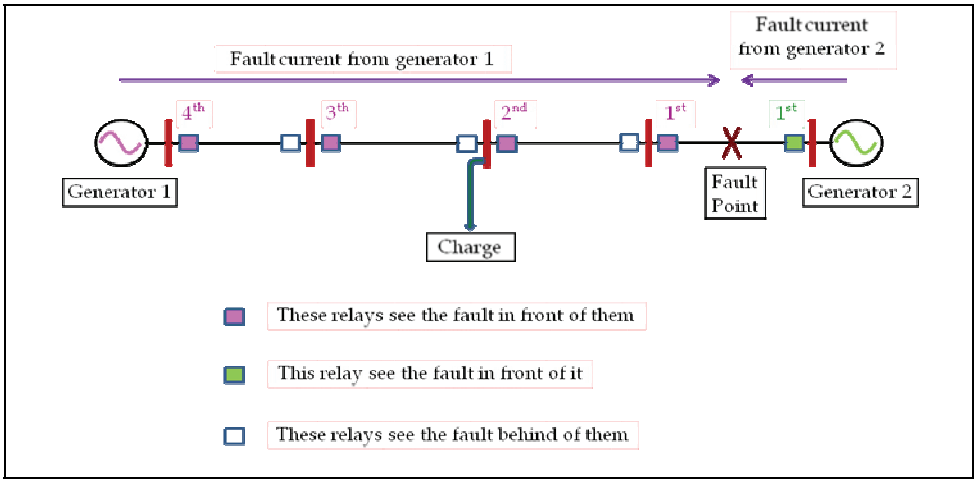



Fig. 9. Fault in the electrical system

Each relay into the protection zone 1, 2, 3 and 4 makes the digital process as illustrated in the figure 9 So, each relay into the protection zone 1 see the fault instantaneously, while that the others relays has a delay. In the case of the protection zone 2 the additional delay is one, into the protection zone 3 the additional delay is two, and in the protection zone 4 is considered as out of operation zone. Then all relays from the figure 1 see the fault, and the power electrical network is in pre-fault state.

The next step is analyzing the voltages and currents signals of the relay into the protection zone 1. These signals are passed through the relay internal transducers, and then the signals are filtered before the analogical-digital conversion. Later a digital filter is used to eliminate the superior harmonics and the no-periodic exponential component still remaining; the useful filters are the finite impulse response ones like the Fourier or cosine types. Remember that to fulfill the window with the fault signal, of one cycle the normal rate of relays is of 32 samples by cycle.

If the fault signal is complete, i.e. it is detected 32 samples in a cycle, and then power electrical network change of the pre-fault to fault state. In other case, when the fault signal is not complete, i.e. it is not detected 32 samples in a cycle, power electrical network change from the pre-fault to normal state.

For this example is considered that the fault signal is complete, then as next step the fault classify algorithm based in the information of the modal transformation identifies the kind of fault. Finally, the electrical components conected to the relay in protection zone 1 are desconnected.

Based on the voltages and currents measurements and its digital processing with a relay to maintain the operation of the power electric system; in the next section we will propos represent with an Interpreted Petri Net the operational behaviour each component for he network in steady state (normal condition), this Interpreted Petri Net will represent the diagnoser. Under the point of view of the relay, the power electrical system have three stages called; pre-fault condition, fault condition (transient condition) and post-fault condition, the Interpreted Petri Net model for the power electrical system includes some

transitions that captures these three stages. Also other real condition of the electrical systems is that a fault is detected with a delay of one cycle due to digital filter (32 samples), thus the pre – faulty transition is fired when a faulty condition appear. On the other hand the reset transition fires when is detected that the system operates in normal conditions and it is necessary reset the pre – faulty condition, so the system change from pre- fault state to normal state.

## 3. Petri nets and Interpreted Petri nets

This section presents a review the main concepts of the Petri net (PN) and Interpreted Petri Net (IPN) formalism used in this chapter. An interested reader can consult (Ramírez-Treviño et al., 2004), (Desel et al., 2005), (Santoyo et al., 2001) and (Ramírez-Treviño et al., 2003) for more details.

### 3.1 Petri Nets

*Definition 1:* A PN system is a pair $(N, M_0)$ where $N = (P, T, I, O)$ is a bipartite digraph which specifies the net structure and $M_0 : P \to Z^+$ is the initial marking. Each element of $N$ is defined as follows $P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of places; $T = \{t_1, t_2, \ldots, t_m\}$ is a finite set of transitions; $I : P \times T \to Z^+$ and $O : P \times T \to Z^+$ are functions representing the weighted arcs going form places to transitions and from transitions to places, respectively. The initial marking of PN $M_0$ is a function that assigns to each place of $N$ a non-negative number of tokens, depicted as black dots inside the places.

A PN structure $N$ can be represented by its incidence matrix $C = [c_{i,j}]_{n \times m}$, where $c_{i,j} = O(p_i, t_j) - I(p_i, t_j)$. The sets $\bullet t_j = \{ p_i \mid I(p_i, t_j) \neq 0 \}$ and $t_j \bullet = \{ p_i \mid O(p_i, t_j) \neq 0 \}$ are the set of input and output places of a transition $t_j$ respectively, which are denominated predecessors and successors of $t_j$ respectively. Analogously, the sets of input and output transitions of a place $p_i$ are $\bullet p_i = \{ t_j \mid I(p_i, t_j) \neq 0 \}$ and $p_i \bullet = \{ O(p_i, t_j) \neq 0 \}$ respectively. In a PN system, a self-loop is a relation where $c_{i,j} = O(p_i, t_j) - I(p_i, t_j) = 0$ and $O(p_i, t_j) \neq 0$, $I(p_i, t_j) \neq 0$.

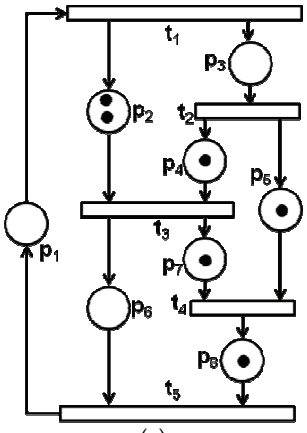
The marking at the *k*-th instant is often represented by a vector $M_k = [M_k(p_1) \;\; M_k(p_2) \;\; \cdots \;\; M_k(p_n)]^T$. Hereafter, a marking $M$ can be represented by a list $M = [1^{M(p_1)}, 2^{M(p_2)}, \cdots, i^{M(p_i)}, \cdots, n^{M(p_n)}]$ where *i*-th item is omitted if $M(p_i) = 0$ and exponents $M(p_i) = 1$ are also omitted. For example, a marking $M = [2 \;\; 0 \;\; 1 \;\; 1]^T$ can be represented by the list $M = [1^2, 3, 4]$.

A transition $t_j$ is enabled at marking $M_k$ if $\forall p_i \in P$, $M_k(p_i) \geq I(p_i, t_j)$; when an enabled transition $t_j$ is fired, then a new marking $M_{k+1}$ is reached. This new marking is computed as $M_{k+1} = M_k + Cv_k$, where $v_k$ is an *m*-entry firing vector whit $v_k(j) = 1$ when $t_j$ is fired once and $v_k(i) = 0$ if $i \neq j$ and $t_j$ is not fired, $v_k$ is called Parikh vector; the equation

$M_{k+1} = M_k + Cv_k$ is called the PN state equation. The *set of enabled transitions* at marking $M_k$ is $E(M_k) = \{t \mid \forall p \in P,\ M_k(p) \geq I(p,t)\}$.

A *firing sequence* of an PN system $(N, M_0)$ is a transition sequence $\sigma = t_i t_j \ldots t_k$ such as $M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} \cdots \xrightarrow{t_k} M_k$ .. The *firing language* of $(N, M_0)$ is the set $L(N, M_0) = \{\sigma \mid \sigma = t_i t_j \ldots t_k \wedge M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} \cdots \xrightarrow{t_k} M_k\}$, while the Parikh vector $\bar{\sigma}: T \to (Z^+)^m$ of $\sigma$ maps every $t \in T$ to the number of occurrences of $t$ in $\sigma$. The fact of reaching $M_k$ from $M_0$ by firing an enabled sequence $\sigma$ is denoted by $M_0 \xrightarrow{\sigma} M_k$. The set of all reachable markings from $M_0$ is $R(N, M_0) = \{M_k \mid M_0 \xrightarrow{\sigma} M_k \text{ and } \sigma \in L(N, M_0)\}$ and it is called *reachability set*.

*Example 1:* Consider the PN of Figure 10.a. The net consists of 8 places $P = \{p_1, p_2, \ldots, p_8\}$ and 5 transitions $T = \{t_1, t_2, \ldots, t_5\}$. The incidence matrix is illustrated in the figure 10.b.




(a)

$$C = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 \\ 2 & 0 & -2 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

(b)

Fig. 10. a) Petri Net System A, b) Incidence matrix of Petri Net System A.

The sets of input and output places of $t_1$ are $\bullet t_1 = \{p_1\}$ and $t_1 \bullet = \{p_2, p_3\}$ respectively. The initial marking is $M_0 = [0\ \ 2\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1]^T$ or $M_0 = [2^2, 4, 5, 7, 8]$. The set of enabled transitions at $M_0$ is $E(M_0) = \{t_3, t_4\}$. When transition $t_3$ fires the net reaches the marking $M_1 = [5, 6, 7^2, 8]$.■

### 3.2 Interpreted Petri Nets

This chapter uses Interpreted Petri Nets (IPN), an extension to PN (Meda M. E., et al., 1998). This extension consists in assigning input and output signals to PN models. Formally IPN are defined as follows.

*Definition 2:* An Interpreted Petri Net (IPN) system is a 6-tuple $Q = (N', \Sigma, \Phi, \lambda, \Psi, \varphi)$ where $N' = (N, M_0)$ is a PN system; $\Sigma = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the input alphabet, where $\alpha_i$ is an input symbol; $\Phi = \{\delta_1, \delta_2, \ldots, \delta_s\}$ is the output alphabet of the net, where $\delta_i$ is an output symbol;

$\lambda : T \to \Sigma \cup \{\varepsilon\}$ is a function that assigns an input symbol to each transition of the net, with the following constraint: $\forall t_j, t_k \in T$, $j \neq k$, if $\forall p_i \;\; I(p_i, t_j) = I(p_i, t_k) \neq 0\}$ and both $\lambda(t_j) \neq \varepsilon$, $\lambda(t_k) \neq \varepsilon$, then $\lambda(t_j) \neq \lambda(t_k)$. In this case, $\varepsilon$ represents an internal system event. $\Psi : P \to \Phi \cup \{\varepsilon\}$ is a labelling function of places that assigns an output symbol or the null event $\varepsilon$ to each place of the net as follows: $\Psi(p_i) = \delta_k$ if $p_i$ represents a output signal, in otherwise $\Psi(p_i) = \varepsilon$. In this case $P_m = \{p_i \mid \Psi(p_i) \neq \varepsilon\}$, $p_i \in P_m$ is called measurable place and $q = \mid P_m \mid$ is the number of measured places. Finally, $\varphi : R(N, M_0) \to (Z^+)^q$ is a function that associates an output vector to every reachable marking of the net as follows: $\varphi(M_k) = M_k \mid_{Pm}$, where $M_k \mid_{Pm}$ is the projection of $M_k$ over $P_m$ i.e. if $M_k = [M_k(p_1) \;\; M_k(p_2) \;\; \cdots \;\; M_k(p_n)]^T$ and $P_m = \{p_i, p_j, \ldots, p_h\}$ then $M_k \mid_{Pm} = [M_k(p_i) \;\; M_k(p_j) \;\; \cdots \;\; M_k(p_h)]^T$ :

Notice that function $\varphi$ is linear and can be represented as a matrix $\varphi = [\varphi_{ij}]_{q \times n}$, where each row $\varphi(k, \bullet)$ of this matrix is an elementary vector where $\varphi(k, i) = 1$ if place $p_i$ is the k-th measured place and otherwise $\varphi(k, i) = 0$ and it is called non- measured.

The transition input alphabet $\Sigma$ of an IPN can be thought as actuator signals attached to the transitions of the net. Similarly, the output alphabet can be thought as sensor signals attached to places. In this context, it is possible to distinguish between controllable and uncontrollable transitions, and between measured and non-measured places of the net as established in the following definitions.

*Definition 3.* If $\lambda(t_i) \neq \varepsilon$ then transition $t_i$ is said to be *controlled*, otherwise *uncontrolled*. $T_c$ and $T_u$ are the sets of controlled and uncontrolled transitions, respectively.

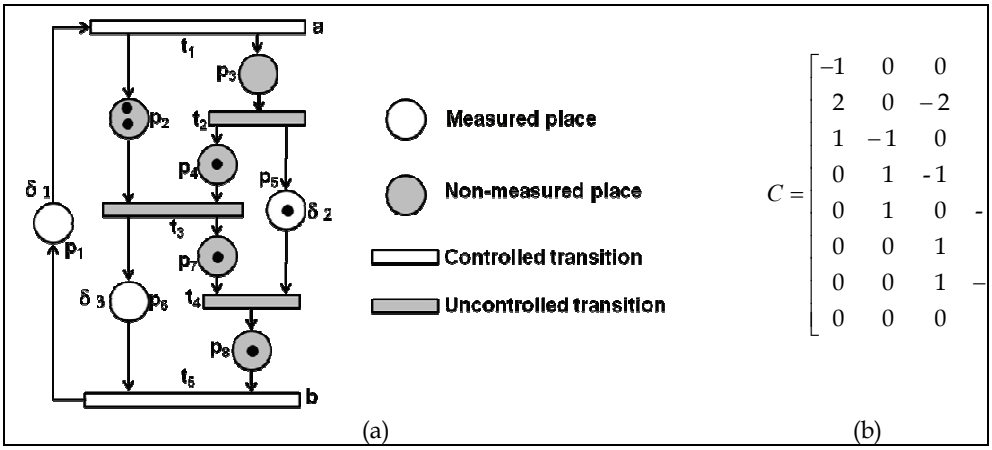
*Definition 4.* A place $p_i \in P$ is said to be *measured* if the *i*-th column vector of $\varphi$ is not null, i.e. $\varphi(\bullet, i) \neq 0$ ; otherwise $p_i$ is *non-measured*. Thus, the set $P_m$ =$\{ p \mid \exists j \in 1,2,\ldots,r$ such that $\varphi(\bullet, j) \neq 0 \}$ is the set of measured places and $P_{nm} = P \setminus P_m$ is the set of non-measured places.

In this chapter, a measured place is depicted as a unfilled circle, while a non-measured place is depicted as a filled circle. Similarly, non-manipulated transitions are depicted by filled bars and manipulated transitions are depicted by unfilled bars. Also, $(Q, M_0)$ will be used instead of $Q = (N', \Sigma, \Phi, \lambda, \Psi, \varphi)$ to emphasize the fact that there is an initial marking in an IPN.

*Example 2:* Consider the IPN shown in Figure 11a. The input and output alphabets are $\Sigma = \{a, b\}$ and $\Phi = \{\delta_1, \delta_2, \delta_3\}$ respectively. Functions $\Psi$ and $\lambda$ are given by:

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $\Psi(p_i)$ | $\delta_1$ | $\varepsilon$ | $\varepsilon$ | $\varepsilon$ | $\delta_2$ | $\delta_3$ | $\varepsilon$ | $\varepsilon$ |

| $k$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $\lambda(t_k)$ | $a$ | $\varepsilon$ | $\varepsilon$ | $\varepsilon$ | $b$ |




$$C = \begin{bmatrix} -1 & 0 & 0 \\ 2 & 0 & -2 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Fig. 11. a) Interpreted Petri Net System B, b) Incidence matrix of Interpreted Petri Net System B.

Thus, the controlled transitions are $T_c = \{t_1, t_5\}$ and the uncontrolled ones are $T_u = \{t_2, t_3, t_4\}$. The measured places are $P_m = \{p_1, p_5, p_6\}$ and the non-measured are $P_{nm} = \{p_2, p_3, p_4, p_7, p_8\}$. In this case, the output function is the matrix:

$$\varphi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{26}$$

The initial output is $y_0 = \varphi(M_0) = [0 \quad 1 \quad 0]^T$. ■

Similarly to a PN, in an IPN system, a transition $t_j$ is enabled at marking $M_k$ if $\forall p_i \in P$, $M_k(p_i) \geq I(p_i, t_j)$; however $t_j$ has fire conditions. When $t_j$ is enabled and $t_j$ is controllable for that $t_j$ fire, it is neccesary that the input signal $\lambda(t_j) \neq \varepsilon$ must be given as input. Otherwise when enabled transition $t_j$ is uncontrollable, then it can be fired. In both cases, when a transition $t_j$ is fired a new marking $M_{k+1} = M_k + C(\bullet, t_j)$ is reached and the output symbol $y_{k+1} = \varphi(M_{k+1})$ is observed.

The following definitions relate the input an output symbol sequences with the firing sequences and the corresponding generated marking sequences. These concepts are useful in the study of the diagnosability property since they are relating to the observed input and output information as the system evolves.

*Definition 5.* A *firing sequence* of an IPN $(Q, M_0)$ is a transition sequence $\sigma = t_i t_j \ldots t_k \ldots$ such as $M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} \cdots \quad M_w \xrightarrow{t_k} \cdots$ The *firing language* of $(Q, M_0)$ is the set $L(Q, M_0) = \{\sigma \mid \sigma = t_i t_j \ldots t_k \wedge \; M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} \cdots \; M_w \xrightarrow{t_k} \ldots\}$.

*Definition 6.* A sequence of *input–output symbols* of an IPN $(Q, M_0)$ is a sequence $w = (\alpha_0, y_0)(\alpha_1, y_1) \cdots (\alpha_k, y_k)$, where $\alpha_i \in \sum \cup \{\varepsilon\}$ is the current input of $(Q, M_0)$ when the ouput changes from $y_i$ to $y_{i+1}$. It is assumed that $\alpha_0 = \varepsilon$, $y_0 = \varphi(M_0)$ and $(\alpha_{i+1}, y_{i+1})$

belongs to $w$ if the pair $(\alpha_i, y_i)$ belongs to the sequence and $y_{i+1} \neq y_i$ and $\neg\exists y_j$ different from $y_i$, $y_{i+1}$ occuring after $y_i$ and before $y_{i+1}$. $\alpha_{i+1}$ are the current input of the IPN when the output changes from the sequence when is the first output change since the occurrence of $y_i$. The set of all *input–output sequences* is denoted by $\Lambda(Q, M_0) = \{w | w$ is an input-output sequence$\}$, the *set of input-output sequences of length greater or equal than to k* is $\Lambda^k(Q, M_0) = \{w | w \in \Lambda(Q, M_0)$ and $|w| \geq k\}$.

*Definition 7*. Let $(Q, M_0)$ be an IPN and $w = (\alpha_0, y_0)(\alpha_1, y_1)\cdots(\alpha_k, y_k)$ be an input-output sequence. The *set of all possible firing transitions sequences that could generate w* is defined as $\Omega(w) = \{\sigma = t_i t_j \ldots t_k \mid \sigma \in L(Q, M_0)$ and $\lambda(t_i)\lambda(t_j)\ldots\lambda(t_k) = \alpha_0\alpha_1\ldots\alpha_k\}$. However, the *transition sequence whose firing actually generates* $w$ is denoted by $\sigma_w$. Notice that $\sigma_w \in \Omega(w)$.

*Definition 8*. The *set of all input-output sequences leading to an ending marking* in the IPN is $\Lambda_B(Q, M_0) = \{w \in \Lambda(Q, M_0) \mid \exists\sigma \in \Omega(w)$ such that $M_0 \xrightarrow{\sigma} M_j$ and if $t_j \in E(M_j)$ then $C(\bullet, j) = \vec{0}\}$. An *ending marking* is a reachable marking that has no enabled transitions or enables only self-loop transitions.

*Definition 9*. Let $(Q, M_0)$ be an IPN and $w = (\alpha_0, y_0)(\alpha_1, y_1)\cdots(\alpha_k, y_k)$ be an input-output sequence. *The set of all possible marking sequences corresponding to* $w$ is $S_w = \{M_i M_j \ldots M_k \mid M_k \xrightarrow{t_x} M_j \xrightarrow{t_y} \cdots \xrightarrow{t_z} M_k$ and $\sigma = t_x t_y \ldots t_z \in \Omega(w)$ and $y_0 = \varphi(M_i)$, $y_1 = \varphi(M_j)$, … , $y_k = \varphi(M_k)\}$.

*Example 3*: Consider the IPN shown in Figure 11.a where $P_m = \{p_1, p_5, p_6\}$ and $T_c = \{t_1, t_5\}$, its reachability graph and its oupts is showed in the figure 12. The following languages can be obtained: $L(Q, M_0) = \{t_4, t_4t_3, t_4t_3t_5, t_4t_3t_5t_1, \ldots\}$,

$\Lambda(Q, M_0) = \{(\varepsilon, [0 \quad 1 \quad 0]^T), (\varepsilon, [0 \quad 1 \quad 0]^T)(\varepsilon, [0 \quad 0 \quad 0]^T), (\varepsilon, [0 \quad 1 \quad 0]^T)$

$(\varepsilon, [0 \quad 0 \quad 0]^T)(\varepsilon, [0 \quad 0 \quad 1]^T), \ldots\}$,

$\Lambda^2(Q, M_0) = \{(\varepsilon, [0 \quad 1 \quad 0]^T)(\varepsilon, [0 \quad 0 \quad 0]^T), (\varepsilon, [0 \quad 1 \quad 0]^T)(\varepsilon, [0 \quad 0 \quad 0]^T)(\varepsilon, [0 \quad 0 \quad 1]^T), \ldots\}$

If $w = (\varepsilon, [0 \quad 1 \quad 0]^T)(\varepsilon, [0 \quad 0 \quad 0]^T)$ then $\Omega(w) = \{t_4\}$ and $S_w = \{M_0 M_2, M_6 M_7\}$ . ■

## 4. Diagnosability and Diagnosable IPN systems

Diagnosability is the property of detecting and locating the fault occurrence in the system trough the available information of the system inputs, outputs and structure. This section reviews the diagnosability definition and a characterization of diagnosable IPN. The reader can consult (Aguirre-Salas & Santoyo-Sanchez, 2009), (Ramírez-Treviño et al., 2007), (Ruiz-Beltrán et al., 2007) and (Santoyo-Sanchez et al., 2008) for more details.

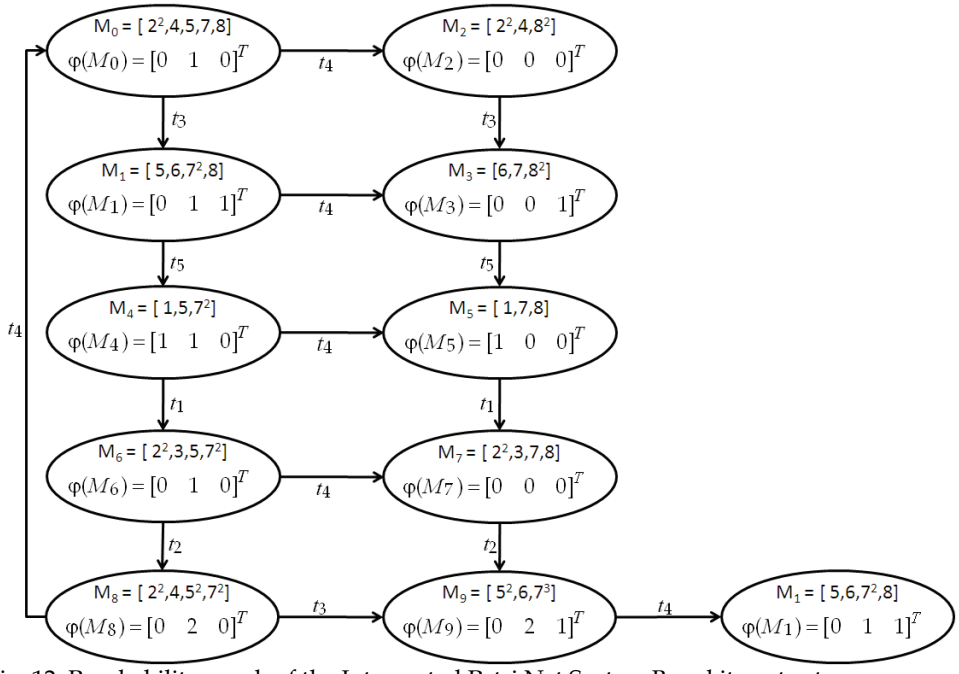



Fig. 12. Reachability graph of the Interpreted Petri Net System B and its outputs.

### 4.1 Concepts

*Definition 10*: An IPN given by $(Q, M_0)$ is *input–output diagnosable* if there exist an integer $k < \infty$ such that $\forall w \in \Lambda^k(Q, M_f) \cup \Lambda_B(Q, M_f)$ it holds that the information provides by $w$ and $(Q, M_0)$ suffices to distinguishable any marking $M_f \in F$ from any other $M_k \in R(Q, M_0)$, by the firing of the transition sequence $\sigma_w$ where $F$ is the set of fault markings. A *fault marking* is a reachable marking that indicates that the system has reached fault conditions.

This definition implies that, for any fault marking of the Interpreted Petri net, its marking becomes distinguish after a finite number of $k$ events. From this definition in order to achieve diagnosability in an IPN, it is necessary to detect the occurrence of each system event, i.e. the firing of each transition from $M_0$ to $M_f$. This leads to the following concepts: sequence-detectability (the possibility of determinate the ocurrence of each firing sequence), event-detectability (the possibility of determine the firing of each transition after its ocurrence and before another transition ocurrence) and marking detectability (the possibility of determinate the system state after the observation of a finite sequence of input-output symbols). The reader can consult (Aguirre-Salas & Santoyo-Sanchez 2009) for more details.

*Definition 11*. An IPN given by $(Q, M_0)$ is *sequence-detectable* if there is an integer $k < \infty$ such that $\forall w \in \Lambda^k(Q, M_0) \cup \Lambda_B(Q, M_0)$ it holds that the information provided by $w$ and $(Q, M_0)$ suffices to uniquely determine $\sigma_w$.

Sequence-detectability implies the knowledge of all firing sequences of an IPN. In others words, there is a function $\Delta_s : \Lambda^k(Q,M_0) \cup \Lambda_B(Q,M_0) \times (Q,M_0) \rightarrow L(Q,M_0)$ where $\Delta_s(w,(Q,M_0)) = \sigma_w$. The problem of determining whether or not a system is sequence-detectable has a high computational complexity. However, the following definition provides conditions that reduce the computational complexity.

*Definition 12.* An IPN given by $(Q,M_0)$ is *event detectable* if the firing of any transition $t_k \in T$ at a marking $M_k \in R(Q, M_0)$ can be uniquely determined through the information provided by the input symbol $\alpha = \lambda(t_k)$ and the output signals $\varphi(M_k)$, where $\varphi C(\bullet,k)$ is the column of $\varphi C$ corresponding to transition $t_k$.

Note that this definition implies that, all events can be detected and distinguisable, i.e. their firing can be detected and distinguisable from each other after its occurrence and before another event occurs. Thus, an event-detectable IPN system is also sequence-detectable. Event detectable has a structural characterization captured in the next lemma that can be tested in a polynomial time.

*Lemma 1*: An IPN given by $(Q,M_0)$ is *event detectable* if only if $\forall i \in [1,2,3,\ldots,m]$, $\varphi C(\bullet,i) \neq \bar{0}$ and $\forall j \neq k \in [1,2,3,\ldots, m]$ such $\varphi C(\bullet,j) = \varphi C(\bullet,k)$, then $\lambda(t_j) \neq \lambda(t_k)$.

Proof. You can find the proof in (Ramírez-Treviño et al., 2003). ■

Observe that an IPN given by $(Q,M_0)$ where it has transitions $t_k, t_j \in T$ with $\lambda(t_k) = \lambda(t_j)$ and $\varphi C(\bullet,k) = \varphi C(\bullet,j) \neq 0$ the firing of $t_k, t_j$ cannot be distinguisable, in this case $t_k, t_j$ are called *indistinguishable*.

*Definition 13.* An IPN given by $(Q,M_0)$ is *marking-detectable* if there is an integer $k < \infty$ such that $\forall_w \in \Lambda^k(Q,M_0)$ it holds that the information provided by $w$ and $(Q,M_0)$ suffices to uniquely determine the marking $M_i$ reached by firing $\sigma_w$.

In other works, there is a function $\Delta_M : \Lambda^k(Q,M_0) \cup \Lambda_B(Q,M_0) \times (Q,M_0) \rightarrow R(Q,M_0)$ where $\Delta_M(w,(Q,M_0)) = M_i$, $\Delta_s(w,(Q,M_0)) = \sigma_w$ and $M_0 \xrightarrow{\sigma_w} M_i$.

*Example 4*: Consider the IPN shown in figure 11.a where $P_m = \{p_1, p_5, p_6\}$ and $T_c = \{t_1, t_5\}$. Its incidence matrix shown in the figure 11.b and its ouput function is the matrix of the equation (26). In this case, $\varphi C$ is the matrix:

$$\varphi C = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \end{bmatrix} \tag{27}$$

Note that the IPN system in the figure 11.a is event detectable by the lemma 1. Then the fire of all events in this IPN can be detected and distinguisable after its occurrence and before another event occurs. To illustrate as the fire of any event in IPN system is detected consider that the information provide by the IPN system is:

$$\text{input } u_1 = \varepsilon \text{ ; output } \begin{cases} y_0 = [0 \quad 1 \quad 0]^T \\ y_1 = [0 \quad 0 \quad 0]^T \end{cases} \tag{28}$$

In this case, $\Delta\varphi = y_1 - y_0 = [0 \quad -1 \quad 0]^T$ and $\varphi C(\bullet,4)$ is the column of $\varphi C$ corresponding to transition $t_4$. Then the fire of the transition $t_4$ is detected trough the information of the input, outputs and structure. Also $\sigma_w = \lambda(t_4)$ where $w = (\varepsilon,[0 \quad 1 \quad 0]^T)(\varepsilon,[0 \quad 0 \quad 0]^T)$, and the IPN system in the figure 11.a is sequence-detectable. Note that also it IPN system is marking-detectable in 2-steps. Finally, consider that set of fault marking is $F = \{M_2, M_7\}$ (see the figure 12) in this case the information provided by $w$ and $(Q, M_0)$ suffices to uniquely determine the marking $M_i \in F$ reached by firing $\sigma_w$, then the IPN system shown in the figure 11.a is input-output diagnosable in 2-steps. ■

### 4.2 Characterization

Sequence-detectability and marking-detectability are both necessary and sufficient conditions for input-output diagnosability, for provide a sufficient condition of input-output diagnosable with a reduced computational complexity is considered a class of IPN defined as follows.

*Definition 14.* A *subnet* of an IPN given by $(Q, M_0)$ with $Q = (N, \Sigma, \Phi, \lambda, \Psi, \varphi)$ and $N = (P, T, I, O)$ is a net $(\tilde{Q}, \tilde{M}_0)$ such that $\tilde{P} \subseteq P$, $\tilde{T} \subseteq T$, $\tilde{\Sigma} \subseteq \Sigma$, $\tilde{\Phi} \subseteq \Phi$ and functions $\tilde{I}, \tilde{O}$ are restrictions of $I$ and $O$ over $\tilde{P} \times \tilde{T}$ respectively, $\tilde{\lambda} : \tilde{T} \to \tilde{\Sigma} \cup \{\varepsilon\}$, $\tilde{\Psi} : \tilde{P} \to \tilde{\Phi} \cup \{\varepsilon\}$ and $\tilde{\varphi} : R(\tilde{Q}, \tilde{M}_0) \to (Z^+)^q$.

*Definition 15.* The *set of fault transitions* of an IPN given by $(Q, M_0)$ with $Q = (N, \Sigma, \Phi, \lambda, \Psi, \varphi)$ and $N = (P, T, I, O)$ is the set $T_f = \{t_k \mid \varphi O(\bullet, t_k) = \bar{0} \text{ and } \lambda(t_k) = \varepsilon\}$.

*Definition 16.* Let $(Q, M_0)$ with $Q = (N, \Sigma, \Phi, \lambda, \Psi, \varphi)$, $N = (P, T, I, O)$ an IPN where $T_f \neq \{\}$. The subnet induced by $T_f$ is $(Q_{Tf}, M_0^{Tf})$ with $N_{[Tf]} = (P_{Tf}, T_{Tf}, I_{Tf}, O_{Tf})$ where $T_{Tf} = T_f$, $P_{Tf} = \bigcup_{i=1}^{k} \bullet t_i \bullet \in T_f$ and $I_{Tf}$, $O_{Tf}$ are restrictions of $I$ and $O$ over $P_{Tf} \times T_{Tf}$ respectively, $\Sigma_{Tf} = \{\}$, $\Phi_{Tf} \subseteq \Phi$ and functions $\lambda_{Tf} : T_{Tf} \to \varepsilon$, $\Psi_{Tf} : P_{Tf} \to \Phi_{Tf} \cup \{\varepsilon\}$ and $\varphi : R(Q_{Tf}, M_0^{Tf}) \to (Z^+)^q$. This subnet is named *fault model*.

*Definition 17.* Let $(Q, M_0)$ with $Q = (N, \Sigma, \Phi, \lambda, \Psi, \varphi)$, $N = (P, T, I, O)$ an IPN where $T_f \neq \{\}$. The *subnet of normal behavior* is $(Q^N, M_0^N)$ with $Q^N \subseteq Q$ where $T_N = T - T_f$, $P_N = P - \{\bigcup_{i=1}^{k} t_i \bullet \in T_f\}$, $\Sigma_N = \Sigma$, $\Phi_N \subseteq \Phi$ and function $I_N$, $O_N$ are restrictions of $I$ and $O$ over $P_N \times T_N$ respectively. In this case $\lambda_N : T_N \to \Sigma \cup \{\varepsilon\}$, $\Psi_N : P_N \to \Phi_N \cup \{\varepsilon\}$ and $\varphi_N : R(Q^N, M_0^N) \to (Z^+)^q$. This subnet is named *diagnoser model*.

*Example 5*: Consider the IPN shown in Figure 13a. The input and output alphabets are $\Sigma = \{a, b\}$ and $\Phi = \{\delta_1, \delta_2, \delta_3\}$ respectively. Functions $\Psi$, $\lambda$, $C$ and $\varphi$ are given by:

| $i$ | 1 | 2 | 3 |
|---|---|---|---|
| $\Psi(p_i)$ | $\delta_1$ | $\delta_2$ | $\delta_3$ |

| $k$ | 1 | 2 | 3 |
|---|---|---|---|
| $\lambda(t_k)$ | $a$ | $b$ | $\varepsilon$ |

$$C = \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \varphi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{29}$$

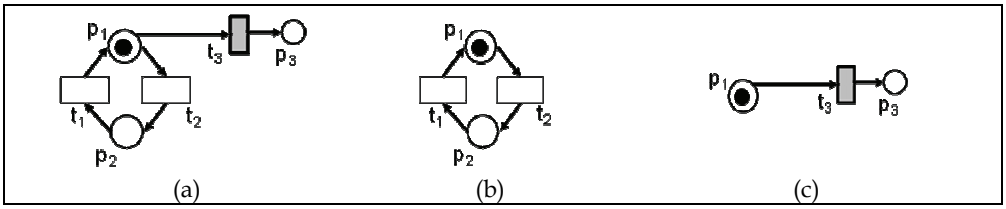



Fig. 13. a) Interpreted Petri Net System C, b) A subnet of Interpreted Petri Net System C, c) Subnet induced by $T_f$ .

Thus, the controlled transitions are $T_c = \{t_1, t_2\}$ and the uncontrolled ones are $T_u = \{t_3\}$. The measured places are $P_m = \{p_1, p_2, p_3\}$ and the non-measured are $P_{nm} = \{\}$. In this case, the figure 13.b is a subnet of normal behavior Interpreted Petri Net system of the figure 13.a with functions $\Psi_N$, $\lambda_N$ , $C_N$ and $\varphi_N$ are given by:

| $i$ | 1 | 2 |
|---|---|---|
| $\Psi_N(p_i)$ | $\delta_1$ | $\delta_2$ |

| $k$ | 1 | 2 |
|---|---|---|
| $\lambda_N(t_k)$ | $a$ | $b$ |

$$C_N = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad \varphi_N = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (30)$$

The subnet induced by $T_f = \{t_3\}$ is showed in the figure 13.c where functions $\Psi_{Tf}$ , $\lambda_{Tf}$ , $C_{Tf}$ and $\varphi_{Tf}$ are given by:

| $i$ | 1 | 3 |
|---|---|---|
| $\Psi_{Tf}(p_i)$ | $\delta_1$ | $\delta_3$ |

| $k$ | 3 |
|---|---|
| $\lambda_{Tf}(t_k)$ | $\varepsilon$ |

$$\varphi_{Sf} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (31)$$

In this chapter to emphasize the fact that the IPN system captures the normal and fault behaviour (illustrated as the IPN systems of the figures 13.b and 13.c respectively) the next definition is proposed.

*Definition 18.* An *IPN system to diagnose* given by $(Q^D, M_0^D)$ is a net where $T_f \neq \{\}$ and the funtions $I$ , $O$ and $C$ are:

$$I = \begin{bmatrix} I_N & I^{Tf} \\ \bar{0} & \end{bmatrix}; \; O = \begin{bmatrix} O_N & O^{Tf} \\ \bar{0} & \end{bmatrix} \text{ and } C = \begin{bmatrix} C_N & C^{Tf} \\ \bar{0} & \end{bmatrix} \qquad (32)$$

where $I_N$ , $O_N$ , $C_N$ are restrictions of $I$ and $O$ over $P_N \times T_N$ , i.e. over the subnet indiced by normal behavior $(Q^N, M_0^N)$; and $I_{Tf}$ , $O_{Tf}$ , $C_{Tf}$ are restrictions of $I$ and $O$ over $P_{Tf} \times T_{Tf}$ , i.e. over the subnet induced by fault behavior $T_f$ ( $(Q_{Tf}, M_0^{Tf})$ ).

Note that the incidence matrix PN system 13.a is:

$$C = \left[\begin{array}{cc|c} 1 & -1 & -1 \\ -1 & 1 & 0 \\ \hline 0 & 0 & 1 \end{array}\right] \qquad \text{with } C_N = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \text{ and } C_{Tf} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \qquad (33)$$

The characterization of input-output diagnosable is based on the idea from the IPN system captures the normal and fault behaviour (illustrated as the IPN systems of the figures 13.b and 13.c respectively. Note that in the figure 13.c non-manipulated transition marking a faulty place. Then it is possible to known when a faulty place is marked and to determinate which is the marked place. This idea is formalized in the next theorem.

*Theorem 1*: Let $(Q^D, M_0^D)$ be an IPN, live, strongly connected and event detectable with $Q = (N, \Sigma, \Phi, \lambda, \Psi, \varphi)$ and $N = (P, T, I, O)$. Let $\{X_1, ..., X_r\}$ be the set of all T-invariants of $(Q^D, M_0^D)$. Let $(Q^N, M_0^N)$ be subnet induced by $T_f$. If $\forall p_i \in P_{Tf}$, where $p_i \in \bullet t_j$ and $t_j \in T_{Tf}$ (i.e. $p_i$ are predecessors of any fault transition) the following conditions hold:

1. $\forall r$, $\exists j \quad X_r(j) \geq 1$, where $t_j \in (p_i) - T_{Tf}$;
2. $\forall t_k \in (p_i) - T_{Tf}$, $\bullet(t_k) = \{(p_i)\}$ and $\lambda(t_k) \neq \varepsilon$.

Then the IPN $(Q^D, M_0^D)$ is input–output diagnosable.

Proof. You can find the proof in (Ramírez-Treviño et al., 2007). ■

### 4.3 Diagnoser Design

The issue with detection and localization of faults consist in identify the abnormal behavior in the systems and locate the root cause or resources that are working in a wrong way. Diagnoser design proposed in (Santoyo-Sanchez et al., 2008) is shows in the figure 14, this scheme consist of six components

1. *Diagnoser model*, which is an IPN, denoted by $(Q^N, M_0^N)$ that represents the normal behavior of the system.
2. *System model*, it is an IPN denoted by $(Q^D, M_0^D)$ which contains the normal and abnormal behavior from the system, where the diagnoser IPN is embedded into the IPN system.
3. *Firing events detector block*, which detect and determine which transitions was fired into the system.
4. *Error block*, it is an Error IPN defined between $(Q^D, M_0^D)$ and $(Q^N, M_0^N)$, which compares the behavior between both IPN systems.
5. *Detecting Fault Marking algorithm*, it detects and locates fault through the Error IPN, also indicating faulty state.
6. *Diagnoser Fault algorithm*, it indicate the component fault and specify the kind of fault that occur.

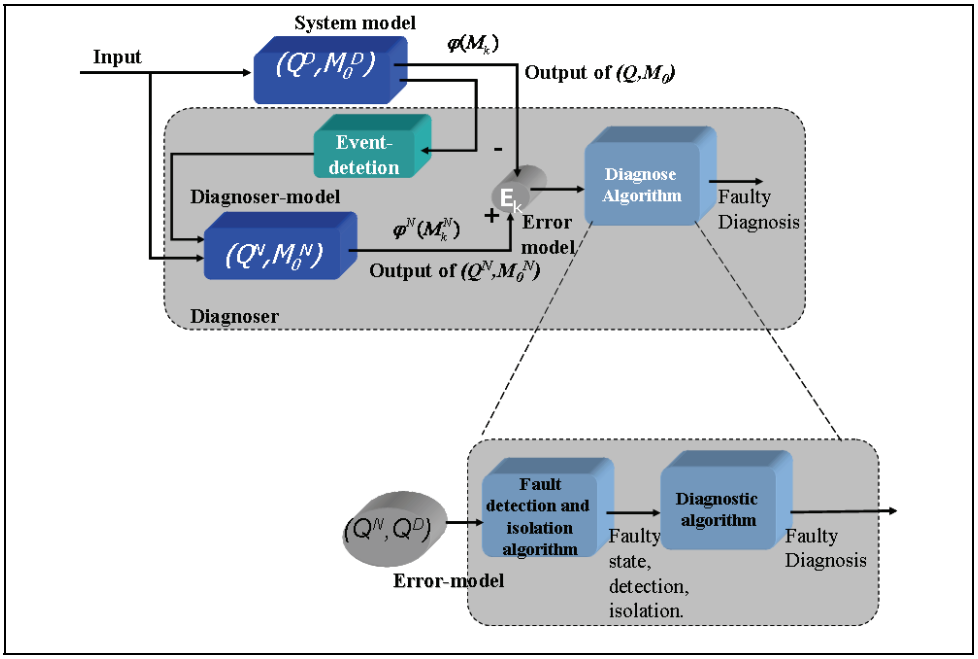



Fig. 14. Scheme for proposed diagnoser.

The diagnose process is based on the idea of that the system behavior is modeled as IPN, which contains the normal and fault behavior. When a transition fires, due to system IPN is event-detectable then it is possible to determine its fires (with the firing events detect block). Moreover, when the system does not fire fault transitions the output of both models (system and diagnoser) is equal, i.e. the system behavior only includes the fire of normal transitions. In the oher case, when a fault transition fires in the system, its fire is detected but this transition is not include into the diagnoser model, then the output of both models (system and diagnoser) is not equal. In this case a fault is detected, and the next steps are to locate the fault, indicate the component fault and specify the kind of fault. To illustrate the general diagnose process consider the next example.

*Example 6*: Consider the IPN shown in figure 13a. as the system model, and the IPN shown in figure 13.b as the diagnoser model. Both system the input and output alphabets are $\Sigma = \{a, b\}$ and $\Phi = \{\delta_1, \delta_2, \delta_3\}$ respectively. And its functions $\Psi$ , $\lambda$ , $C$ and $\varphi$ are given by the equations (29) and (30) respectively. In this case, $\varphi C$ and $\varphi_N C_N$ are the matrix:

$$\varphi C = \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \qquad \varphi_N C_N = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{34}$$

Note that the IPN system in the figure 13.a is event detectable by the lemma 1. Assume that from $M_0$ and $M_0^N$ the information provide by the IPN system is:

$$\text{input } u_1 = b \text{; output } \begin{cases} y_0 = [1 \quad 0]^T \\ y_1 = [0 \quad 1]^T \end{cases} \text{ and input}_N \ u_1 = b \text{; output}_N \begin{cases} y_0^N = [1 \quad 0]^T \\ y_1^N = [0 \quad 1]^T \end{cases} \tag{35}$$

In this c ase, $\Delta\varphi = y_1 - y_0 = [1 \quad -1]^T$ and $\varphi C(\bullet,2)$ is the column of $\varphi C$ corresponding to transition $t_2$. Also its output error between both systems is $\Delta y = y_1^N - y_1 = [0 \quad 0]^T$. In this case the system does not fire fault transitions. Assume that from $M_0$ and $M_0^N$ the information provide by the IPN system is:

$$\text{input } u_1 = \varepsilon \text{; output } \begin{cases} y_0 = [1 \quad 0]^T \\ y_1 = [0 \quad 0]^T \end{cases} \text{ and input}_N \ u_1 = \varepsilon \text{; output}_N \begin{cases} y_0^N = [1 \quad 0]^T \\ y_1^N = [1 \quad 0]^T \end{cases} \tag{36}$$

In this case, $\Delta\varphi = y_1 - y_0 = [-1 \quad 0]^T$ and $\varphi C(\bullet,3)$ is the column of $\varphi C$ corresponding to transition $t_3$. And its output error between both systems is $\Delta y = y_1^N - y_1 = [1 \quad 0]^T$. In this case a fault transition fires in the system, and a fault is detected, i.e. an error different to zero.

In general, the error concept among systems is computed as differences among theirs outputs. In the observer and controller design when the error is zero then the system reach a required behavior (De Jesús & Ramírez-Treviño, 2001). In the context of diagnoser design to localize the fault transition and the place of fault is used an error structure introduced in (De Jesús & Ramírez-Treviño, 2001) and (Santoyo-Sanchez et. al, 2008), which is presented in the next definition.

*Definition 19* Let $(Q^D, M_0^D)$ and IPN where $(Q^N, M_0^N)$ and $(Q_{Tf}, M_0^{Tf})$ are its subnets IPN diagnoser model and IPN fault model respectively. Structure Error between $(Q^N, M_0^N)$ and $(Q^D, M_0^D)$ is defined as $(N^E, M_0^E)$ where $N^E = (P^E, T^E, I_E, O_E)$ with $P^E = P^D$, $T^E = T^D$, $I_E : P^E \times T^E \to Z^+$, $O_E : P^E \times T^E \to Z^+$ and $C_E : P^E \times T^E \to Z^+$ defined as:

$$I_E = \begin{bmatrix} I_N & -I_N & -I_{Tf} \\ \vec{0} & \vec{0} & \end{bmatrix}; \ O_E = \begin{bmatrix} O_E & -O_E & -O_{Tf} \\ \vec{0} & \vec{0} & \end{bmatrix} \text{ and } C_E = \begin{bmatrix} C_N & -C_N & -C_{Tf} \\ \vec{0} & \vec{0} & \end{bmatrix} \tag{37}$$

The initial marking of IPN $M_0^E : P^E \to Z$ defined as:

$$M_0^E = \begin{bmatrix} M_0^N \\ \vec{0} \end{bmatrix} - \begin{bmatrix} M_0^N \\ M_0^{Tf} \end{bmatrix} \tag{38}$$

The marking at the *k*-th instant is:

$$M_k^E = \begin{bmatrix} M_k^N \\ \vec{0} \end{bmatrix} - \begin{bmatrix} M_k^N \\ M_k^{Tf} \end{bmatrix} \tag{39}$$

A transition $t_j \in T^N$ is enabled at marking $M_k^E$ if $\forall p_i \in P^N$, $M_k^E(p_i) \geq I^N(p_i, t_j)$; while that a transition $t_j \in T^D$ is enabled at marking $M_k^E$ if $\forall p_i \in P^D$, $M_k^E(p_i) < I^D(p_i, t_j)$. When a transition $t_j$ is fired, then a new marking $M_{k+1}$ is reached. This new marking is computed as:

$$M_{k+1}^{E} = M_{k}^{E} + \begin{bmatrix} C^{N} & -C^{N} & -C^{F} \\ \vec{0} & \vec{0} & \end{bmatrix} \begin{bmatrix} \sigma_{k}^{N} \\ \gamma_{k}^{D} \\ \beta_{k}^{F} \end{bmatrix} \tag{40}$$

where $\sigma_k^N$ is an $m$-entry firing vector of structure $(Q^N, M_0^N)$, $\gamma_k^N$ is an $m$-entry firing vector of structure $(Q^D, M_0^D)$ and $\beta_k^F$ is an $m$-entry firing vector of structure $(Q_{Tf}, M_0^{Tf})$.

*Example 7*: Consider the IPNs shown in Figure 13.a, 13.b and 13.c note that this IPN are $(Q^D, M_0^D)$, $(Q^N, M_0^N)$ and $(Q_{Tf}, M_0^{Tf})$ respectively. Structure Error between $(Q^N, M_0^N)$ and $(Q^D, M_0^D)$ is:

$$M_0^E = \begin{bmatrix} M_0^N \\ \vec{0} \end{bmatrix} - \begin{bmatrix} M_0^N \\ M_0^{Tf} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}; \qquad C = \left[\begin{array}{cc|cc|c} -1 & 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & -1 \end{array}\right] \quad \varphi = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{41}$$

which is showed in the figure 15. In this error marking there not are enabled transitions. Assume that from $M_0$ and $M_0^N$ the information provide by the IPN system is:

$$\text{input } u_1 = \varepsilon \text{; output } \begin{cases} y_0 = [1 \quad 0]^T \\ y_1 = [0 \quad 0]^T \end{cases} \text{ and input}_N \ u_1 = \varepsilon \text{; output}_N \ \begin{cases} y_0^N = [1 \quad 0]^T \\ y_1^N = [1 \quad 0]^T \end{cases} \tag{42}$$

In this case, $\Delta\varphi = y_1 - y_0 = [-1 \quad 0]^T$ and $\varphi C(\bullet,3)$ is the column of $\varphi C$ corresponding to transition $t_3$. And its output

error between both systems is $\Delta y = y_1^N - y_1 = [1 \quad 0]^T$. In this case an error is detected and the error marking is:

$$M_1^E = \begin{bmatrix} M_1^N \\ \vec{0} \end{bmatrix} - \begin{bmatrix} M_1^N \\ M_1^{Tf} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{43}$$

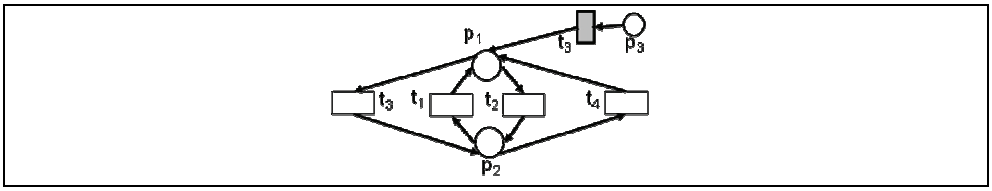



Fig. 15. Representation of the structure error model.

In this marking, the enabled transitions are $t_2 \in T^N$ and $t_3 \in T_{Tf}$, if $t_2$ fires into the structure error the new marking $M_{k+1}$ is reached. This new marking is computed as:

$$M_2^E = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} + C\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} \tag{44}$$

In the case of that $t_3$ fires into the structure error the marking:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} + C\begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T \tag{45}$$

Then from $M_1^E$ to reach the error zero, it is necessary fires the fault transition $t_3$ and $p_3 = {}^\bullet t_3$ is the fault place. ■

The following theorem characterizes the diagnosis based on structure error models.

### 4.4 Diagnose Fault

*Theorem 2*: Let $(Q^D, Q^N)$ be a pair system-diagnoser with the state equations (46) and (47) respectively. Where the diagnoser is based on the idea that the diagnoser IPN $(Q^N, M_0^N)$ is embedded into the IPN system $(Q^D, M_0^D)$ and each faulty transition is non-manipulable (see the figure 13.a).

$$Q^D = \begin{cases} M_{k+1}^D & = M_k^D + C^D \vec{\sigma}_k^D \\ y_k^D & = \varphi_N(M_k^D) \end{cases} \tag{46}$$

$$Q^N = \begin{cases} M_{k+1}^N & = M_k^N + C^N \vec{\sigma}_k^N \\ y_k^N & = \varphi_N(M_k^N) \end{cases} \tag{47}$$

If $(Q^D, M_0^D)$ is input–output diagnosable (Ramírez-Treviño et al., 2007) and the structure Error between $(Q^N, M_0^N)$ and $(Q^D, M_0^D)$ is defined as $(N^E, M_0^E)$ then it is possible to detect and isolate a fault into the system $(Q^D, M_0^D)$.

Proof. You can find the proof in (Santoyo-Sanchez et al., 2008). ■

Based on the theorem 2 the following algorithm is presented to detect and isolate error marking.

---

Algorithm 1: Detecting and isolate error marking.

---

*Inputs*: The IPN model of the pair system-diagnoser.

*Outputs*: The error marking $M_k^E$, faulty place $p_{F\,i}$ and faulty transition .

*Procedure*:

1. Define the structure error.
2. When $M_k^E \neq \vec{0}$ then:

2.1 Faulty places are $p_F = \{\forall p_i \mid M_k^E(p_i) = -1\}$

2.2 $\forall p_i \in p_F$ faulty transitions are $t_F = \{\forall t_k \mid (p_F)\bullet\}$ .

3. Return $M_k^E$ , $p_F$ and $t_F$ .

---

According with the scheme of the figure 14 the diagnose algorithm has two parts; in the first one the algorithm 1 detects and locates fault through the structure error, also indicating faulty state. In the second one, it is necessary to define a diagnostic fault algorithm, which indicate the component fault and specify the kind of fault. In this case, it is necessary to consider the characteristics of power electrical network.

### 4.5 Diagnostic fault in power electrical networks

Under the point of view of the analogical-digital conversion, the minimal elements of the power electrical network are: A) lines, B) sources and C) charges; using the methodology proposed in (Santoyo et. al, 2001) each minimal element is represented as Interpreted Petri Net. Additionally in (Santoyo-Sanchez et. al, 2008) the power flow from the generator to charge is considered as an element of the power electrical network. For illustrate the IPN modeling of power electrical systems consider the IPN model of the figure 16, which represent the IPN model of the power electrical network of the figure 8 only of power flow from the generator 1. Note that in the figure 16 for each line with a relay the fault behavior is modeled in two parts. The first one represents the fault window, i.e. the normal rate of relay; the second one represents when the fault condition is reached.

To capture the protection zone by each relay (see the figure 6) in the context of IPN, in this chapter is proposed to define a new function $\varphi_Z$ given by a matrix, where each row $\varphi_Z(k,\bullet)$ represents the protection zone, which is defined considering the trajectory of energy distribution (because a relay can detect the fault in from of them) and its fault zone (first, second and third).

*Definition 20.* Let $\vartheta : Z^+ \to \{Z^+\}$ a relation that indicates which lines is front other in second and third zone . The *protection zone by each relay* is defined as:

$$\varphi_z(k,i) = \begin{cases} 1 & \text{if place } p_i \text{ is the fault place of } k-th \text{ relay.} \\ 1 & \text{if place } p_i \text{ is the fault place of } \vartheta(k). \\ 0 & \text{in other case.} \end{cases} \tag{48}$$

*Example 8.* For illustrated the protection zone consider the IPN system of the figure 16.b. . In this case $\vartheta(2) = \{3,4,5\}$ , $\vartheta(3) = \{4,5\}$ , $\vartheta(4) = \{5\}$ and $\vartheta(5) = \{\}$ . Then the protection zone induced by $\vartheta$ is:

$$\varphi_z = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix} \tag{49}$$

For the relay 1, note that $p_{32}$ is its fault place, $p_{33}$ and $p_{34}$ are the faulty places of line 3 and 4 respectively. Assume that the marking is $M_k = [1,3,5,10,15,20,22,25,27,28,29,30,31,32,33,34]$ the compute of potection is

$\varphi_z(M_k) = [3 \quad 2 \quad 1 \quad 0]^T$ then relay 1(into the line 2) is in the third protection zone, relay 2 (into the line 3) is in the second protection zone, and finlay the relay 3 (into the line 4) is in the first protection zone. ■

When a fault occurs, the marking is analized to determine each fault zone (figure 6). Like a resulting of this process, each relay determines its tripping time. The relays acts instantaneously when the fault location at the first zone. Then the protection devices acts as fast as possible to disconnect the faulted element, the element into the first zone. The next algorithm captures this idea.

---

Algorithm 2. Diagnosis Fault.

---

*Inputs*: The IPN model of system, the structure error, error marking $M_k^E$, faulty place $p_F$ and faulty transition .

*Outputs*: Faulty component $Comp_F$, the sets Fault, $\Pr oteccD_{Comp_F}$ and $\Pr oteccI_{Comp_F}$ .

*Procedure*:

When $M_k^E \neq \bar{0}$ do:

1. Compute the protection zone using $\varphi_z$ .
2. Define $Comp_F = \{\forall c_i \mid p_i \in (t_F)\bullet$ and $p_i$ is the place used to represent the system element $c_i$ and $c_i$ into the protection zone 1}.
3. Define the Protection behavior.
   3.1 $\Pr oteccD_{Comp_F} = \{\forall c_i \mid p_i \in M(p_i) = 1$, where $p_i$ is the place used to represent the disconnection of the system element $c_i$ by the fault in $Comp_F$ }.
   3.2 $\Pr oteccI_{Comp_F} = \{\forall c_i \mid p_i \in M(p_i) => 1$, where $p_i$ is the place use to describe how the electrical system distribute power electrical and $c_i$ indicates the electrical element that is stressed due to fault into $Comp_F$ }.
4. Diagnostic. Return the sets Fault, $\Pr oteccD_{Comp_F}$ and $\Pr oteccI_{Comp_F}$ .

---

The algorithm 2 indicates fault and disconnection of elements by the protections occurrence in the electrical system; thus it is possible to distinguish between fault elements and consequences of the faults. In this case the IPN model and the algorithms (1 and 2) are desingned for a line, due to the line has three phases the model is generated for each phase (sequence positive, negative and zero). Finally using the information of the table 1 is specify the kind of fault.

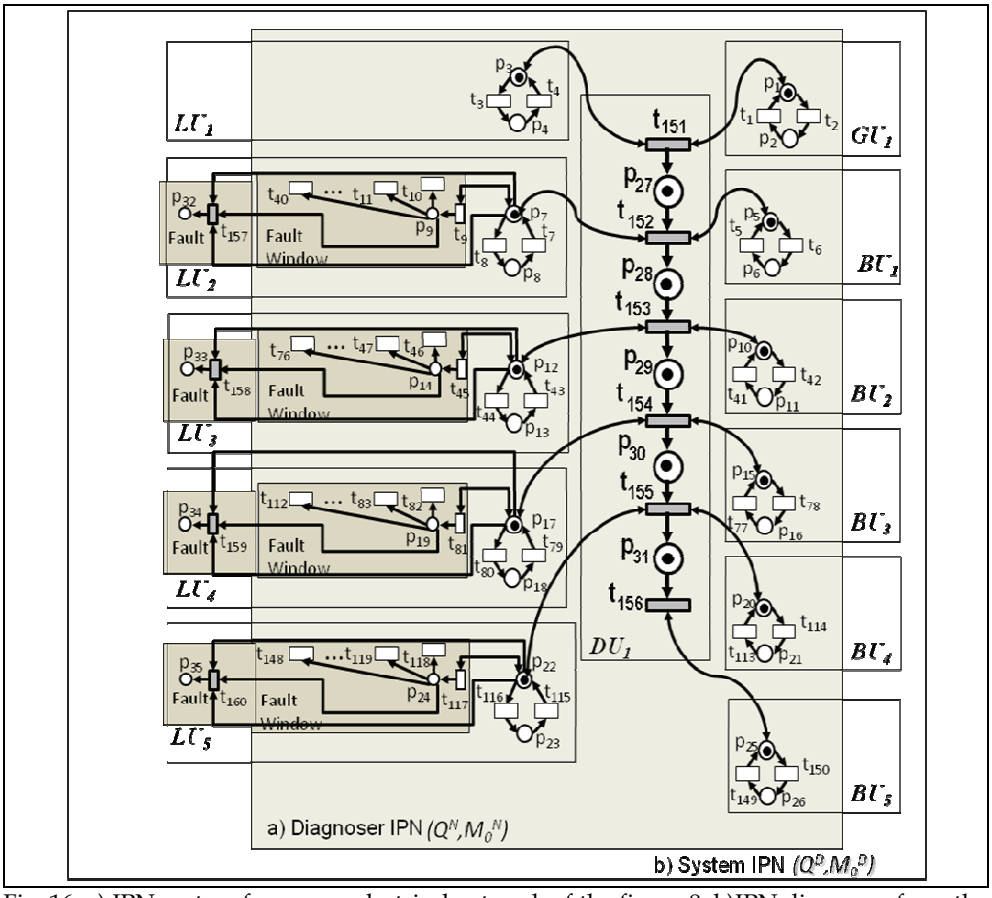



Fig. 16. a) IPN system for power electrical network of the figure 8. b)IPN diagnoser from the power electrical network of the figure 8.

## 5. Application Example

For illustrate the diagnoser behavior consider the figure 16.a and 16.b which are the pair system $(Q^D, M_0^D)$ -diagnoser $(Q^N, M_0^N)$ for the power electrical network of the figure 8 and the power flow from the generator 1, $\varphi_Z$ defined in the equation (49), $\varphi_N$ is a identity matrix of 31 x 31 and $\varphi_D = [\varphi_N \quad D]$ where $D$ is matrix of 31 x 4 with zero. Consider that a fault ocurrs in $LU_5$ (see the figure 9), when the window fault is completed all the realy front the marking in the system is $M_k^D = [1,3,5,10,15,20,25,27,28,29,30,31,32,33,34,35]$ while that in the diagnoser is $M_k^N = [1,3,5,9^{32},10,14^{32},15,19^{32},20,24^{32},25,27,28,29,30,31]$, then its output error between both systems is $\Delta y = y_k^N - y_k^D = [0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 32 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 32 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 32 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 32 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]^T$

in this case all fault transitions fire in the system, and faults are detected, i.e. an error different to zero.Marking error of structure error between $(Q^N, M_0^N)$ and $(Q^D, M_0^D)$ in the $k$-th instant is: $M_k^E = [\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 32\ 0\ 0\ 0\ 0\ 32\ 0\ 0\ 0\ 0\ 32\ 0\ 0\ 0\ 0\ 32\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ -1\ -1\ -1\ -1\ ]^T$. Using the algorithm 1, it is obtained $p_F = \{p_{32}, p_{33}, p_{34}, p_{35}\}$, $t_F = \{t_{157}, t_{158}, t_{159}, t_{160}\}$. The diagnosis algorithm (algorithm 2) computes the protection zone as $\varphi_z(M_k^D) = [4\ \ 3\ \ 2\ \ 1]^T$ (illustrated in the figure 9), $Comp_F = \{LU_5\}$, i.e. the faulty component is the transmission line 5 because $p_{35}$ is used to represent a part of behavior of $LU_5$ into the first protection zone. $\mathrm{Pr}oteccD_{Comp_F} = \{\}$; i.e. all others electrical elements are connected without fault. Moreover no one other component of the electrical system is disconnecting as consequence of the fault. $\mathrm{Pr}oteccI_{Comp_F} = \{BU_4\}$, i.e. the bus (BU) is stressed due to the fault in $LU_5$. Note that electrical element stressed is easy to computed, because the faulty in $Comp_F = \{LU_5\}$ produces that transition $t_{155}$ cannot be fired, and the predecessor and successors of $\bullet t_{155} \bullet = \{p_{20}, p_{22}, p_{30}, p_{31}\}$, where $p_{20}$ and $p_{22}$ represent the normal operation of Bus $(BU_4)$ and line $LU_5$ respectively; $p_{30}$ is the place into $DU_1$ such that the tokens will be accumulated into it, and $p_{31}$ is the place into $DU_1$ such that the tokens are decreased. Thus using the structure of the IPN it is possible to anticipate a future fault in the electrical component stressed. Thus the kind fault depends of the characteristical of V and I for each phase (sequence positive, negative and zero). If the fault is maintain then eventually the algorithm 2 compute:
$\mathrm{Pr}oteccD_{Comp_F} = \{BU_4, LU_5, LU_4, BU_3\}$; i.e. electrical elements disconnected by protection, in this case by the previous relay. And $\mathrm{Pr}oteccI_{Comp_F} = \{LU_3, BU_2\}$; i.e. the next protections operations will activate. ■

## 6. Conclusions

If the most important design consideration of relaying is security then is really important and reasonable to have a support methodology to assure the relay operation. Security involves the ability to avoid operations for which tripping is not desired because in core the electrical system is designed to maintain the electric service. In this sense IPN is an adequate methodology to watch over the correct operation of the entire power system network, IPN diagnoses sample by sample is the electric system is working in a steady state condition, therefore IPN adds redundancy to the relay. IPN do not substitute the relay, IPN increases security in the system operation which is the main goal of relaying.
We have been proposed a diagnosis scheme allowing detecting and locating faults of electrical systems modeled as IPN. Since the electrical elements can fault simultaneously, the system IPN contain relations denoted as DU representing the electrical flow. The method that is proposed for diagnosis consists in two algorithms. In the first algorithm is detected and located the error marking through of the structure error. While the second algorithm classified the set of faulty systems in power systems in order to estimate the origin of the fault and consequences of the fault. The diagnoser function was illustrated with a study case. The proposed is based on the voltages and currents measurements and its digital processing with a relay to maintain the operation of the power electric system.

## 7. References


Aguirre-Salas L. & Santoyo-Sanchez A. (2009). Sequence-detectability analysis of Interpreted Petri nets under partial state observations, To be published in *Proceedings of IEEE International Conference on Emerging Technologies and Factory Automation*, Location: Palma de Mallorca Span, September 2009, IEEE Press, USA.

Desel J., Esparza J. & van Rijsbergen C. J. (2005), *Free choice Petri nets*, Cambridge University Press, ISBN-13: 9780521019453 | ISBN-10: 0521019451,Cambridge, UK.

De Jesús C. A. & Ramírez-Treviño A. (2001). Controller and Observer Synthesis in Discrete Event Systems Using Stability Concepts, *Proceedings of IEEE System Man and Cybernetics*, pp. 664-668, ISBN: 0-7803-77-2, Location: Tucson Arizona USA, October 2001, IEEE Press, USA.

Fink L.H. Badley D.E, Koehelr J.E., Mcinnis D.A. & Redmond O.H., (1985), Emergency Control Practices, In: *IEEE Transactions on PAS*, Vol. 104, September 1985, pp. 2336-2341, ISSN: 0018-9510.

Genc S., & Lafortune S. (2003). Distributed diagnosis of discrete-event systems using petri nets, *Proceeding of International Conference on Applications and Theory of Petri Nets*, pp 316-336, ISBN 3-540-40334-5, Location: Eindhoven, The Netherlands, June 2003, Springer, Berlin, Germany.

Greenspan D. & Casulli V. (1988), *Numerical analysis for applied mathematics, science, and engineering*, Addison-Wesley Publishing Company, Inc., ISBN 0-201-09286-7, Redwood, California USA.

Guzman A., Schweitzer III E. O., Tziouvaras D. A. & Martin K. (2006). Local and Wide-Area Network Protection Systems Improve Power System Reliability, ***Power Systems Conference: Advanced Metering, Protection, Control, Communication, and Distributed Resources***, pp.174-181, ISBN: 0-615-13280-4 Location: Clemson, SC, March 2006, IEEE Press, USA.

Hadjicostis C. N. & Verguese G. C. (2000). Power Systems Monitoring using Petri Net Embeddings, *Proceedings of the IEE Generation, Transmission and Distribution*, pp. 299-303, ISSN: 1350-2360, Location: Washington D.C. USA, October 2003, Publisher IEEE Press, USA.

Hadjicostis C. N. & Verghese G. C. (1999a). Monitoring Discrete Event Systems using Petri Net Embeddings, In: *Application and Theory of Petri Nets 1999, No. 1639 in Lecture Notes in Computer Science*, S. Donatelli, J. Kleijn, pages 188-208, Springer-Verlag, ISBN:3-540-66132-8 , London, UK .

Hadjicostis C. N. & Verghese G. C. (1999b). Structured Redundancy for Fault Tolerance in LTI State-Space Models and Petri Nets, In: *Kybernetika*, Vol. 35, January 1999, page 39-55, ISSN 0023-5954.

IEEE Std C37.1 - 1994: *Definition, Specification, and Analysis of Systems used for Supervisory Control, Data Acquisition and Automatic Control*.

Lefebvre D. & Delherm C. (2007). Diagnosis of DES with Petri Net models, In: *IEEE Transactions on Automation Science and Engineering*, Vol. 4, No. 1, January 2007, pages 114-118, ISSN 1545-5955.

Madani V., Novosel D., Apostolov A. & Corsi S. (2004), Innovative Solutions for Preventing Wide Area Disturbance Propagation, *Proceeding of the IREP Symposium for Bulk Power Systems Dynamics and Control VI*, pp. 729 - 750, ISBN 88-87380-47-3, Location Cortina d'Ampezzo, Italy, August 2004, IEEE PES, USA.

Meda M. E., Ramirez A. & Malo A (1998), Identification in discrete event systems. *Proceeding of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 740-745, ISBN: 0-7803-4778-1, Location: San Diego CA., October 1998, IEEE Press, USA.

Naredo Villagran J. L. A. (1992), *The Effect of Corona on Wave Propagation on Transmission Lines*, Ph. D. Thesis, Department of Electrical Engineering, Faculty of Applied Science, University of British Columbia, British Columbia.

Naredo J.L., Silva J.L., Romero R., Moreno P. (1987). Application of Approximated Modal Analysis Methods for PLC System Design, In: *IEEE Transactions on PWRD*, Vol. 2, No. 1, January 1987, pages 57-63, ISSN: 0885-8977.

Proth J. M., DiCesare F., Silva M., Harhalakis G. & Vernadat F. B. (1993), *Practice of Petri Nets in Manufacturing*, Chapman & Hall, ISBN 9780412412301, London.

Ramírez-Treviño A., Rivera-Rangel I. & López-Mellado E. (2003) Observability of discrete event systems modeled by interpreted Petri nets, In: *IEEE Transactions on Robotics and Automation*, Vol. 19, No. 4, August 2003, pages 557–565, ISSN: 1042-296X.

Ramírez-Treviño A., Ruiz-Beltrán E., Rivera-Rangel I. &. López-Mellado E. (2007), On-line Fault Diagnostic of Discrete Event Systems. A Petri Net Based Approach, *IEEE Transactions on Automation Science and Engineering*, Vol. 4, No. 1, January 2007, pages 31-39, ISSN 1545-5955.

Ramírez-Treviño A., Ruiz-Beltrán E., Rivera-Rangel I. & López-Mellado E. (2004). Diagnosability of Discrete Event Systems. A Petri Net Based Approach, *Proceedings of the IEEE International Conference on Robotic and Automation*, pp. 541-546, ISBN: 0-7803-8232-3, Location: New Orleans LA , April-May 2004, IEEE Press, USA.

Ren H. & Zengqiang M. (2006), Power Systems Fault Diagnosis Modeling Techniques based on Encoded Petri Nets, *Proceeding of Power Engineering Society General Meeting*, pp. 18-22, ISBN 1-4244-0493-2 , Montreal Canada, June 2006, IEEE Press, USA.

Ren H., Zhao Hongshan, Mi Zengqiang & Liu Yan (2004), Power Systems Fault Diagnosis by Use of Encoded Petri Net Models, *Proceeding of Power System Technology*, pp. 64-68, ISBN: 0-7803-8465-2 China, Vol. 28, No. 5, IEEE Press, USA.

Ruiz-Beltrán E., Ramírez-Treviño A., López-Mellado E. & Arámburo-Lizárraga M. (2007). A structural characterization of diagnosticable Petri net models, *Proceeding of the IEEE International Conference on Automation Science and Engineering*, pp. 1137-1142, ISBN: 978-1-4244-1154-2, Location: Scottsdale, September 2007, IEEE Press, USA.

Sampath M., Sengupta R., Lafortune S., Sinnamohideen K., & D. Teneketzis (1995), Diagnosability of discrete event systems, *IEEE Transactions on Automatic Control*, Vol. 40, No. 9, September 1995, pages 1555-1575, ISSN: 0018-9286 .

Sampath M., Sengupta R., Lafortune S., Sinnamohideen K., & Teneketzis D. (1996), Failure Diagnosis Using Discrete-Event Models, *IEEE Transactions on Control System Technology*, Vol. 4, No.2, March 1996, pages 105-124, ISSN: 1063-6536.

Santoyo A., Jiménez-Ochoa I. & Ramírez-Treviño A. (2001). A complete cycle for controller design in Discrete Event System, *Proceedings of IEEE System Man and Cybernetics*, pp. 2688-2693, ISBN: 0-7803-77-2, Location: Tucson Arizona USA, October 2001, IEEE Press, USA.

Santoyo-Sanchez A., Ruiz-Beltrán E., Aguirre-Salas L.I., & Ortiz-Muro V.H. (2008), Fault Diagnosis of Electrical Systems using Interpreted Petri Nets, *Proceedings of IEEE International Conference on Emerging Technologies and Factory Automation*, pp 538 –

546, ISBN 978-1-4244-1505-2, Location Hamburg Germany, September 2008, IEEE Press, USA.

Sheng-Luen C., Chien-Chung W. & Muder J. (2003). Failure Diagnosis: A case study on Modeling and Analysis by Petri Nets, *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, pp. 2727-2732, ISBN 0-7803-7952-7, Location: Washington D.C. USA, October 2003, IEEE Press, USA.

Strang G. (1988), *Linear Algebra and its Applications*, Third Edition, Harcourt Brace & Company, ISBN 0-15-551005-3, Orlando, Florida USA.

Ulerich N.H. & Powers G. J. (1988), On-line hazard aversion and fault diagnosis in chemical process, *IEEE Transactions on Reliability*, Vol. 37, June 1988, pages 171 – 177, ISSN: 0018-9529.

Yang B., Jeong S., Oh Y. & Tan A.C. (2004), Case-based reasoning system with Petri nets for induction motor fault diagnosis, *Expert Systems with applications*, Vol 27, August 1994,pp. 301 – 311, ISSN: 0957-4174.

# GPenSIM: A New Petri Net Simulator

Reggie Davidrajuh
*University of Stavanger*
*Norway*

## 1. Introduction

Petri net is being widely accepted by the research community for modeling and simulation of discrete event-driven systems, mainly due to Petri net's rigorous modeling techniques. There are a number of Petri net tools available for free academic use; see PNWorld (2009) for a list of tools. These tools are advanced tools flexible enough to model complex and large systems. This paper talks about developing a new Petri net simulator. The reasons for building a new simulator are:

- Flexible: the simulator should enable easy integration with other libraries and tools, so that developing hybrid models (e.g. Fuzzy Petri nets, by integrating Petri net with Fuzzy Logic) becomes easy
- Extensible: the simulator should enable users writing their own extensions, either extending or rewriting the existing functions or developing new functions.
- Easy of use: for those who doesn't want to use mathematics when developing a model, the tool should provide a natural language user interface, so that the mathematical details are abstracted away from the user.

General-purpose Petri net simulator (GPenSIM, 2009) is developed by the first author of this paper, in order to satisfy the three criteria stated above (flexible, extensible, and ease of use). GPenSIM is realized as toolbox for the MATLAB platform, so that diverse toolboxes that available in the MATLAB environment (e.g. Fuzzy Logic Toolbox, Control Systems Toolbox) can be used in the models that are developed with GPenSIM.

## 2. Existing Tools for Discrete Event Simulation

Many tools satisfy some of the three criteria mentioned above. Automata, Stateflow, and Petri nets are the well-known tools used for simulation of discrete event systems. Though automata have a strong footing in computer science, the serious shortcoming with it is the lack of structure – the ability to modularize a system (decompose a system into modules) [2]. Stateflow is commercial software that runs in MATLAB environment [8]. Stateflow is similar to Petri net; converting a Petri net model of a discrete event system into a Stateflow model and vice versa is easy. However, learning Stateflow, with its syntactic, semantic, and graphical details, is much more difficult than learning Petri net. In addition, Stateflow also demands some knowledge of Simulink, in addition to MATLAB.

Petri net is being widely accepted for modeling and simulation of discrete event systems and there is a number of Petri net tools available free-of-charge for academic usage (PNWorld, 2009). These tools are sophisticated tools flexible enough to model complex and large systems. However, these tools are stand-alone systems, and for integrating the functions of these tools with other tools or libraries, one need to program in either high-level languages like Java or C++, or use XML as an intermediary. Thus seamless integration of these Petri net tools with other types of tools (e.g. Control Systems) is not possible.
GPenSIM, written in MATLAB language, allows seamless integration with the other toolboxes that also available in the MATLAB environment. Programming in MATLAB Language is also extremely easy as the language resembles the BASIC language.

## 3. Architecture of GPenSIM

GPenSIM is designed using the well-proven paradigms in software engineering such as: layered architecture, modular components, and natural language interface.

### 3.1 Layered architecture

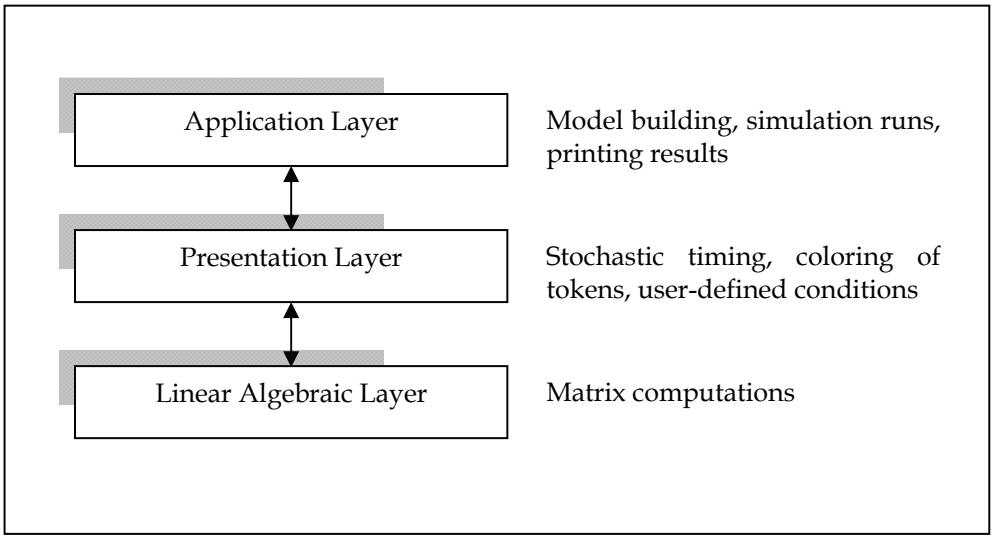



Fig. 1. 3-layer architecture

GPenSIM is built following 3-layer architecture; see figure 1. The bottom layer deals with Petri net run-time dynamics; this layer computes newer states with the help of linear algebraic equations and matrix manipulations. The middle layer adds more high-level functionality such as stochastic timing, coloring of tokens, user-defined conditions ('guard-conditions' in some literature), etc. The top layer offers applications such building a Petri net based model, running simulations, determining coverability tree, printing the simulation results, etc.

### 3.2 Modular components

A model of a discrete event system developed with GPenSIM consists of a number of files. The main simulation file (MSF) is the file that will be run directly by the MATLAB platform. In addition to the main simulation file, there will be one or more Petri net definition files (PDFs); definition of a Petri net graph (static details) is given in the Petri net Definition File. There may be a number of PDFs, if the Petri net model is divided into many modules, and each module is defined in a separate PDF. While the Petri net definition file has the static details, the main simulation file contains the dynamic information (such as initial tokens in places, firing times of transitions) of the Petri net. In addition to these files (main simulation file and Petri net definition files), there can be a number of transition definition files (TDFs) too.

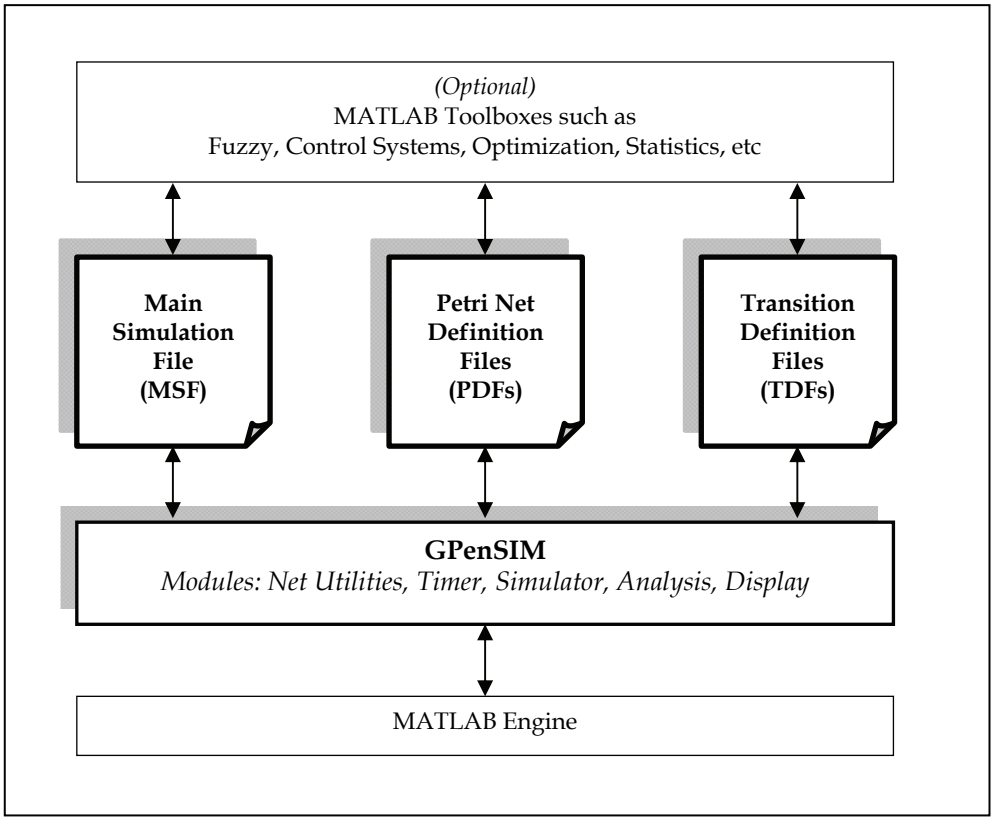



Fig. 2. The architecture of GPenSIM

A transition definition file consists of additional conditions that determine whether an enabled transition can fire or not. The additional conditions are called 'user defined condition' in GPenSIM terminology, whereas in some other literature (e.g. Colored Petri Net (CPN)) it is referred to as 'guard-functions'). There can be a separate transition definition file for each transition in a Petri net model.

### 3.3 Natural language interface

Users need not know Petri net mathematics when creating a Petri net model of a discrete event system. GPenSIM offers a natural language interface with which model building mainly deals with identifying the basic elements of a system and establishing the connections between these elements. Figure 2 shows the overall architecture of GPenSIM.

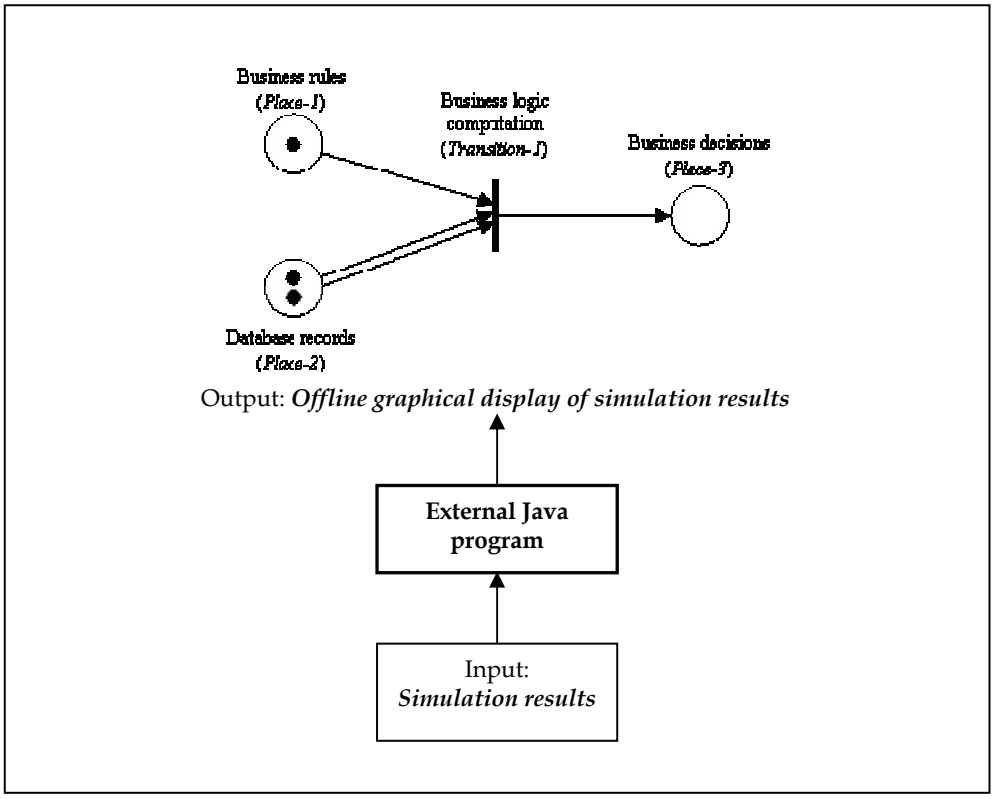



Fig. 3. Offline graphical display of simulation results

### 3.4 Offline graphical display

After simulation runs, the simulation results can be used for printing results both in ASCII and in graphic format. The results can be also used for off-line (non-interactive) graphical display of step-by-step simulation run; to do the offline display, we need an external program, written in high level language like Java or C#. At present, an external Java based program is under construction. However, step-by-step online (interactive) monitoring of simulation run in progress is neither available at present nor planned for construction in the near future.

### 3.5 The main loop

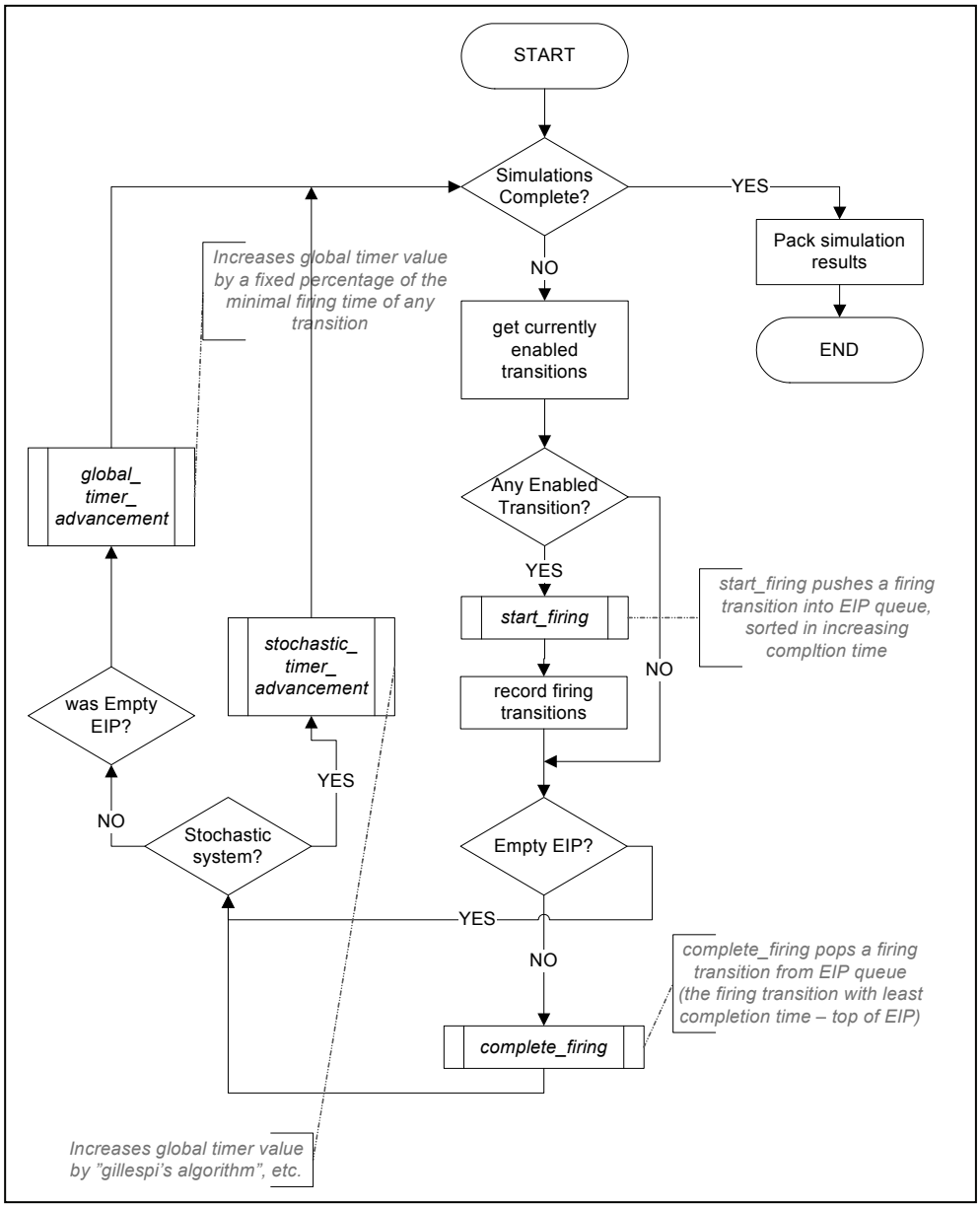



Fig. 4. The main loop of the simulation runs

Figure 4 shows the main loop of the simulator. As in any Petri net simulator, the main loop consists of a simple cycle that first checks whether any transitions are enabled and then it

puts the enabled transitions into the firing queue, provided that the transitions satisfy additional user defined conditions, if any; inputs tokens are also taken away (consumed) by the corresponding transitions. Then, the loop checks whether any firing transitions are completing or have completed. In this case, the firing transitions are popped out of the firing queue and output tokens are deposited into the respective output places.

Figure 4 also shows that there are two kinds of timers are in use. The first timer – called global timer is the one that is normally used. The second timer called stochastic timer is used only for 'stochastic systems'. Stochastic systems can be leisurely defined as continuous systems (as opposed discrete systems) that are to be discretized first into discrete systems so that a Petri net model can be created for them. A case study on stochastic system is done in section 5.

Finally, the main loop shown in figure 4 hints an extension to GPenSIM: by using a Real-Timer (computer's real-time clock) instead of stochastic or global timer, a Real-Time GPenSIM version can be developed. This Real-Time GPenSIM is basically a soft Programmable Logic Controller (PLC), which will use a Digital & Analogue Input Output Card (DAC) to read sensor inputs from the outside world and will also output digital signals to triggers via the card. In this real.-time version, the main loop should read the sensor data at the start of each cycle, and the state of the firing transitions should be mapped to the output triggers.

## 4. Methodology for Modelling and Simulation with GPenSIM

Creating a Petri net model consists of two steps:

1) Defining the static Petri net graph, and
2) Assigning initial dynamics in the main simulation file

Step-1) Defining the Petri net graph in one or more Petri net Definition Files (PDF): this is the static part. This step consist of three sub-steps:

   a. Identifying the basic elements of a Petri net graph: the places,
   b. Identifying the basic elements of a Petri net graph: the transitions, and
   c. Connecting the elements with arcs

Step-2) Assigning the dynamics of a Petri net in the Main Simulation File (MSF):

   a. The initial markings on the places, and possibly
   b. The firing times of the transitions

After creating a Petri net model, simulations can be done.

## 5. Application Example

GPenSIM has been used for modeling different types of discrete event system; e.g. Davidrajuh (2007) presents model of an adaptive supply chain; Davidrajuh (2009) presents a simulation study of a Bluetooth Wireless technology based classroom tool.

This application example deals with discretizing of continuous systems. Generally, Petri net is for discrete event simulations only. However, if a continuous system can be discretized, then this system could also be modeled with Petri nets. However, discretizing of a continuous system is not easy and needs some understanding of Petri net formalism and matrix representation; interest reader is referred to Wilkinson (2006).

The application example is a prey-predator (e.g. rabbit-fox) ecological equilibrium. The equilibrium is stated by 2 simple differential equations (known as Lotka & Volterra equation):

- The specimen prey (e.g. rabbit - r) mutates by itself and depleted by predators (e.g. foxes - f):

$$\frac{dr}{dt} = (\alpha \cdot r) - (\beta \cdot r \cdot f) \tag{1}$$

- The specimen predator (e.g. fox) grows due to rabbits (access to food) and depleted by its own population (competition for food):

$$\frac{df}{dt} = -(\gamma \cdot f) + (\delta \cdot r \cdot f) \tag{2}$$

- $\alpha, \beta, \gamma,$ and $\delta$ are parameters representing the interaction of the two species.

The equilibrium is determined by partial differential equations; equivalent Petri net model for the interaction is given in figure 5.

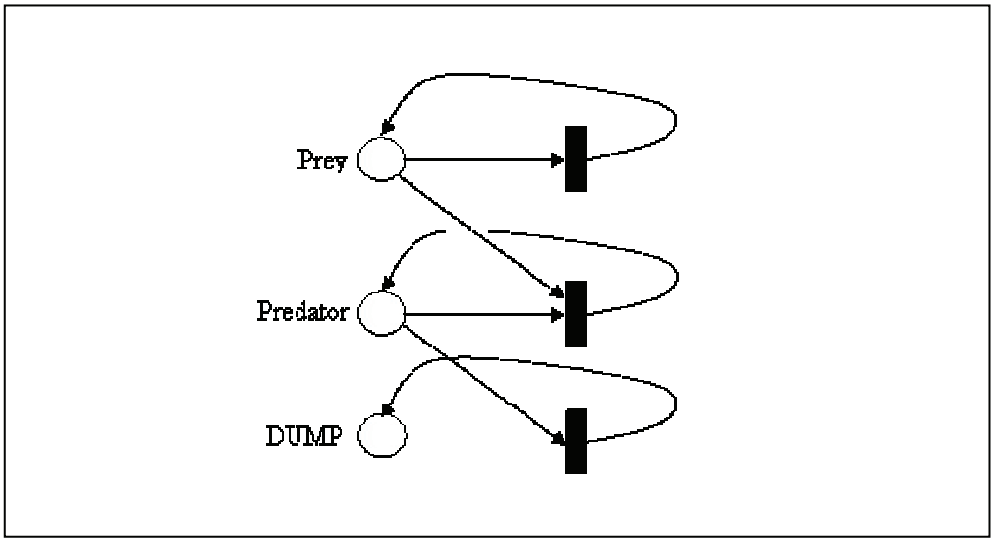



Fig. 5. Petri net model of Pre-Predator interaction

### 5.1 Creating the model

Petri net definition file (PDF) that defines the static Petri net graph of figure 5 is given below:

```
% function: Petri Net Definition File (PDF)
% filename: predator_prey_def.m

function [PN_name, set_of_places, set_of_trans, set_of_arcs]...
                = predator_prey_def(global_info)
PN_name='predator-prey p/151';
set_of_places = {'Prey', 'Predator', 'DUMP'};
set_of_trans = {'t1','t2','t3'};
set_of_arcs = {'Prey','t1',1, 't1','Prey',2,...
   'Prey','t2',1, 'Predator','t2',1, 't2','Predator',2,...
   'Predator','t3',1, 't3','DUMP',1};
```

The main simulation file (MSF) is give below. MSF first identifies the PDF and then assigns the initial dynamics. Then, it runs the simulations using the function ‘gpensim’. Finally, the simulation results are printed.

```
%  function: MAIN SIMULATION FILE (MSF)
%  filename: predator_prey.m
global_info.MAX_LOOP = 10000;    %  many simulation runs
global_info.c = [1 .005 .6];
global_info.STOCHASTIC = 1; %  stochastic timer
pn = petrinetgraph('predator_prey_def');
dynamicpart.initial_markings = {'Prey',50, 'Predator', 100};
sim = gpensim(pn, dynamicpart, global_info);
```

```
M = plotp(pn, sim, {'Prey','Predator'});   %figure 6a
plot(M(:,1), [M(:,2), M(:,3)]);   % figure 6b
```

Stochastic timer: Due to discretization, the simulations should use stochastic clock, rather than the inbuilt global timer Wilkinson (2006). The realization of *Gilespi* algorithm (Gilespi, 1977) for advancing stochastic timer is given below.

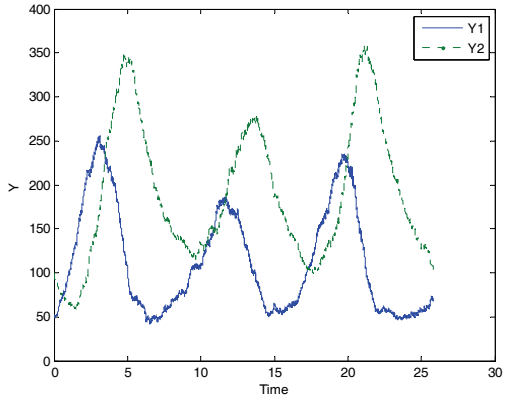
```
%  function: realization of Gilespi's algorithm
%  filename: time_advancement.m
function [pn, global_info] = time_advancement(pn, global_info)
c1=global_info.c(1);  c2=global_info.c(2);  c3=global_info.c(3);
Prey = get_place(pn, 'Prey');
PRED = get_place(pn, 'Predator');
h1 = c1 * Prey.tokens;
h2 = c2 * Prey.tokens * PRED.tokens;
h3 = c3 * PRED.tokens;
H = h1 + h2 + h3;
%%%%  probabilities
global_info.pro1 = (h1/H);
global_info.pro2 = (h2/H);
global_info.pro3 = (h3/H);
delta_T = 1-exp(-1/H);
pn.current_time = pn.current_time + delta_T ;
```

Finally, Transition Definition File (TDF) for the transition t1 is given below. TDFs for the transitions t2 and t3 are similar.

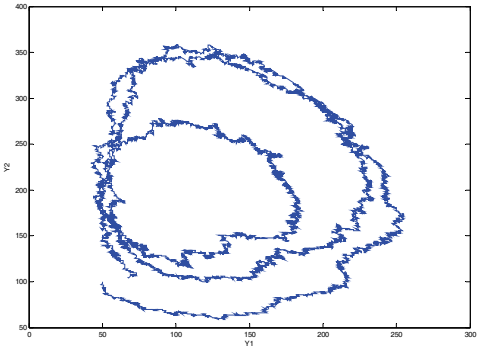
```
%  function: Transition Definition File (TDF) for transition t1
%  filename: t1_def.m
function [fire, new_color, override, selected_tokens,global_info] = ...
        t1_def (pn, new_color, override, selected_tokens,global_info)

c1=global_info.c(1); c2=global_info.c(2); c3=global_info.c(3);
Prey = get_place(pn, 'Prey');
PRED = get_place(pn, 'Predator');
h1 = c1 * Prey.tokens;
h2 = c2 * Prey.tokens * PRED.tokens;
h3 = c3 * PRED.tokens; H = h1 + h2 + h3;
%%%%  probabilities
pro1=(h1/H); pro2=(h2/H); pro3=(h3/H);

R = rand*(1);
fire =  (R <= pro1);
```

### 5.2 Simulation results

Figure 6a shows variation of prey and predator population with time. Figure 6b shows how the prey population varies against the predator population (prey-predator equilibrium).




6a) Composition of specimens Prey-Predator with time




6b) Prey-Predator equilibrium

Fig. 6. Simulation results

## 6. Conclusion

This chapter presents a new Petri net simulator, called General Purpose Petri Net simulator (GPenSIM), for modeling and simulation of discrete event systems. The tool is devised to achieve the following:

- Flexibility: ability to cooperate with diverse tools and libraries
- Extensibility: ability to offer support for rewriting or extending existing functions or new functions
- Ease of use: Tool may be based on rigorous mathematics, but the user need not know it

At present, GPenSIM has the following limitations:

- It is based on a commercial platform (MATLAB), which is not free for academic (or commercial) use.
- GPenSIM does not incorporate online (interactive) simulator. Thus, monitoring the system during the simulation run is not possible. A Java based program for offline graphical display of the simulation results is under construction.
- Though GPenSIM offer extensibility, it comes with a cost: one need to program in MATLAB language. Though programming in MATLAB is easy as this language resembles a simpler BASIC language, still one need to spend some time to learn the language.

**Further Work:**

There are numerous possibilities for extending GPenSIM. We give blow just two:

- Adaptive GPenSIM: a version of GPenSIM in which the arc weights are not fixed and can vary during the simulation run.
    - Self adaptive: In each TDF, the arc weight of the transition can be changed.
    - Forced adaptive: in a specific TDF, arc weights of any transition can be varied
- Real-time ("soft PLC") simulator: Instead of global timer, the real-time clock of the computer can be used. In this case, the GPenSIM is no longer just a simulator, but it becomes a soft Programmable Logic Controller.

## 7. References

Cassandras, G. and LaFortune, S. (1999) *Introduction to Discrete Event Systems*. Hague, Kluwer Academic Publications

Davidrajuh, R. (2007). "A Service-Oriented Approach for Developing Adaptive Distribution Chain", *International Journal of Services and Standards*, Vol. 3, No.1, pp. 64 – 78

Davidrajuh, R. (2009)."Evaluating Performance of a Bluetooth-based Classroom Tool". *International Journal of Mobile Learning and Organisation,* Vol. 3, No. 2, pp. 148-163

Extend (2009). Available: http://www.imaginethatinc.com/

Gilespi, D. (1977) "Exact Stochastic Simulation of Coupled Chemical Reactions". *The Journal of Physical Chemistry*, Vol. 1, No. 25, pp. 2340 – 2351

GPenSIM (2009). Available: http://www.davidrajuh.net/gpensim/

LabView (2009). Available: http://www.ni.com
MATLAB (2009). Available: http://www.mathworks.com
Petri net world (2009). Available:
http://www.informatik.uni-hamburg.de/TGI/PetriNets/
Pritsker Corporation (1990). SLAM II Quick Reference Manual. Pritsker Corporation, West Lafayattee, IN, USA
SIMCSRIPTII (2009). Available: http://www.simscript.com/
Wikipedia (2009). Available: http://www.wikipedia.org
Wilkinson, D. (2006) *Stochastic Modelling for Systems Biology*. Chapman & Hall / CRC, NY

26

# Assessing Risks in Critical Systems using Petri Nets

Lucio Flavio Vismari and João Batista Camargo Junior
*Computer and Digital Systems Engineering Department*
*School of Engineering at University of São Paulo (Poli-USP)*
*São Paulo, Brazil*

## 1. Introduction

The advent of ubiquitous distributed computing, communication and sensing systems has created an environment in which one can access, process and communicate huge amounts of data. This environment could be mentioned as the major enabler for new applications of control for large-scale, complex systems (Murray et al., 2003). Currently, energy and water distribution, transportation and manufacturing processes are some application areas that use the integration of distributed computing and communication in supervision and control systems to optimize their processes performance and to increase their service capacity. In order to reduce time and cost of development, increase flexibility for systems expansions or modifications, promote greater reuse of proven components and, consequently, reduce the system lifecycle costs and enable a higher number of applications (Hammett, 2003), some systems are being implemented with distributed computing architectures controlled over open (standard-oriented) communication networks (mostly commercial communication networks) and with "Commercial Off The Shelf" (COTS) hardware and software components (Ahlström & Torim, 2002; Knight, 2002b).

If, on the one hand, the characteristics of distributed computing systems with supervision and control loop closed over networks make some applications feasible, on the other hand they pose new challenges and problems to be solved. One of the main challenges for applying this new system paradigm over real applications is its complexity. The huge number of functional elements distributed and interacting over open communication networks makes the system behavior unpredictable, both in operational condictions and in presence of failures. Another challenge comes from the interdisciplinary characteristic of the new paradigm. Particular concepts of control, computer sciences and communication disciplines must be considered together. As a consequence, new points-of-view to be considered are introduced into the systems lifecycle. As an example, concepts of time delays, data transfer rates and packet length (from communication discipline); stability (from control discipline), security and real-time computing (from computing discipline) must be systemically considered in all system lifecycle stages. A last challenge mentioned here is the lack of well-established methods to assess systems based on this new paradigm, which is a need considering all the new paradigm characteristics relationship between

supervision and control system and the application process. In this case, one of the current problems here is modeling and analyzing hybrid systems, in which continuous and discrete variables must be modeled and analyzed together.

Challenges increase when this new supervision and control paradigm is applied to dependable critical applications, mainly regarding safety, where failures could result in loss of lives, significant damage to property or to the environment (Knight, 2002a). Due to the unacceptable consequences that could impact the environment when critical systems are working out of their specifications, any (dependable) critical system must be submitted to a rigorous process of assessment to guarantee that all the risks contained in its lifecycle is acceptable to the society. Hence, due to the characteristics of complexity (defined by the strong interdependence between multidisciplinary variables) in the current critical systems, the challenge is to deal with all of these relevant characteristics during the system assessment process in a complete and efficient way. Therefore, the mission of this chapter is presenting an assessment framework based on the Fluid Stochastic Petri Net (FSPN), the Reward Structures and Variables and the Discrete Event Simulation (DES), which could deal with the characteristics and properties found in the current (dependable) critical systems.

The Fluid Stochastic Petri Net (FSPN) is a formalism commonly applied to model and analysis systems with the same characteristics identified in the current dependable critical systems (concerning safety domain, security domain, time-related domain and hybrid systems domain). In FSPN formalism, the primitive elements of a Stochastic Petri Net (SPN) – such as places (P), transitions (T) and arcs (A) – can represent continuous and discrete variables in hybrid systems, besides representing the (stochastic) dynamic behavior between those interdependent parts (a desirable property when complex systems are modelled). Using FSPN, it is possible to perform a stochastic analysis obtaining performance, reliability, availability and safety measures (dependability measures) based on the uncertainty parameter of the systems. Finally, the FSPN has the capacity to model continuous state spaces, interdependent concerning the discrete process, resulting in a useful modeling formalism to the real systems.

The FSPN, in its pure formalism, allows representing a limited number of system metrics. Improving the representativeness of metrics in the proposed framework, the concept of *Reward Structures and Variables* was incorporated. The existence of the reward structures in the FSPN formalism allows assessing complex and hybrid systems using a variety of measures, defined during the modeling phase. Obtaining the measures of interest in a quantitative way requires calculating the state probability of the modeled system. Thus, in this framework we propose the use of Discrete Event Simulation (DES) to obtain the expected values of reward functions. *Simulation* is a better choice to solve non-Markovian Petri nets, as well as complex and non-limited nets in which state spaces are analytically unsolvable. Since the Stochastic Petri Net is a formalism applied to describe the dynamics of discrete systems, where the state evolution (mark changing) occurs in discrete and well defined events, the solution of nets by *Discrete Event Simulation* (DES) is an extension of the formalism itself. However, the simulation technique only allows statistically estimating the values of interest. Therefore, the quantitative analysis using simulation techniques, mainly DES, must incorporate statistical inference concepts as a way to analyze values with acceptable confidence levels, and this approach was incorporated in the proposed framework.

In short, this chapter will address the key topics related to the proposed assessment framework. Both the formal concepts of FSPN and its ability to represent hybrid systems will be presented. Also, the formalisms of reward nets and their use in quantitative assessment process will be discussed. A topic about resolution methods in FSPN and the ways to apply the DES to estimate measures of interest in FSPN models will be included. Finally, the application of the proposed framework (presented in this chapter) to a real dependable critical system – the Air Traffic Control (ATC) system - will be shown.

## 2. A Framework to assess risks in dependable critical systems

### 2.1 The current system paradigm and the proposed assessment framework

An accident can be defined as a consequence of a hazardous (or unsafe) state that promotes unacceptable losses of lives, injuries and material and environmental losses (Kirner, 1997). Therefore, an accident is an event (or a sequence of events) that violates system safety requirements and, consequently, the main objective of a safety-critical system is to minimize, to acceptable levels, conditions that can contribute to the occurrence of an accident. In this way, a safety assessment process needs to identify what hazard (or unsafe) states can be reached in a system, assess their risks and, finally, to establish countermeasures to minimize and protect a system against the occurrence of hazard states that can lead it to an accident (Leveson, 1995). Hence, it is mandatory to have a complete and efficient way to consider all the current safety-critical system characteristics during the assessment process, allowing the complete identification and the assessment of emergent risks in systems.

As mentioned before, dependable critical systems are being implemented with distributed computing architectures over open-standard communication networks, and the introduction of this paradigm promotes a huge number of interdependent, multidisciplinary characteristics in systems. Two of the most important characteristics that need to be considered in the safety assessment process are regarded to time-related domain and hybrid domain:

***Time-related domain:*** implementation of distributed architectures controlled over networks introduces time-related characteristics in systems, reflected by strong timing constraints. In the case of control being closed-loop over networks, stochastic variations of time delay in networks and in control processing must be considered. Delays in control system may promote effects such as performance degradation and destabilization (Tipsuwan & Chow, 2003) that are dangerous in safety critical applications. Another temporal constraint aspect is related with task sequences. In safety critical systems, tasks must be rigorously triggered. Currently, time-triggered protocols are recommended in safety-critical applications controlled over networks since they guarantee, with some level of confidence, a correct task sequence, despite its lower performance compared to event-triggered protocols (Philippi, 2002; Heck et al., 2003). Concluding, network characteristics such as delays and latency must be considered over safety critical systems requirements.

***Hybrid domain:*** as reported by Heck et al. (2003), a hybrid system is an abstract system with both continuous and discrete components. Current safety critical systems have computer-based systems controlling safety critical applications. Computer-based systems have their behavior performed by discrete-state machines, modeled by discrete states, and safety-critical applications have their behavior modeled by continuous variables (e.g. aircraft velocity, wind speed, temperature, etc). A hybrid model is necessary to modeling and

analyzing the affectation of disturbances/failures in control system over safety related applications and vice-versa.

Based on the new paradigm characteristics listed before, it was verified in the literature (Tuffin et al., 2001; Wolter, 2000; Gribaudo, 2003) that Fluid Stochastic Petri Nets (FSPN) are a formalism applied to modeling and assessing systems with the same characteristics identified for the current safety critical system paradigm. In FSPN formalism, primitive elements of a Stochastic Petri net (SPN) such as places (P), transitions (T) and arcs (A) can represent both continuous ($P_c$, $T_c$, $A_c$) and discrete ($P_d$, $T_d$, $A_d$) variables in a hybrid system, besides the dynamic behavior between those parts. FSPNs are fully applied to modeling and verify functional and safety properties in hybrid systems (Gribaudo et al., 2002). A formal definition of FSPN can be made by (adapted from Computer Science Department (2000) and Ciardo et al. (1999)):

$$\text{FSPN} = (P, T, A, f, \Pi, g, \text{Dist}, r, \text{Afet}, w, b, \mu_0, \chi_0) \quad (1)$$

where some import arguments of (1) are defined as (the other will be defined further on):

$P = P_d \cup P_c = (p_1, p_2, ..., p_{\#Pd}) \cup (q_1, q_2, ..., q_{\#Pc})$, $P_d \cap P_c = \varnothing$.

$\mu_{(t)}(p_i) \in N$, $\forall p_i \in P_d$ is a marking (state) in discrete net at time $t$, $t \in R^+$,

$\chi_{(t)}(q_i) \in R^+$, $\forall q_i \in P_c$ is a marking (state) in continuous net at time $t \in R^+$,

$T = T_T \cup T_I = (t_1, t_2, ..., t_{\#Tt}) \cup (u_1, u_2, ..., u_{\#Ti})$, $T_T \cap T_I = \varnothing$.

$A = A_d \cup A_c = ( ((P_d \, x \, T) \cup (T \, x P_d)) \, x \, \hat{S} \rightarrow N ) \cup ( ((P_c \, x \, T_I) \cup (T_I \, x P_c)) \, x \, \hat{S} \rightarrow R^+ )$.

$f = ((P_c \, x \, T_T) \cup (T_T \, x P_c)) \, x \, \hat{S} \rightarrow R^+$): marking-dependent *fluid rate.*

Graphical representation of FSPN elements are illustrated in Fig. 1.

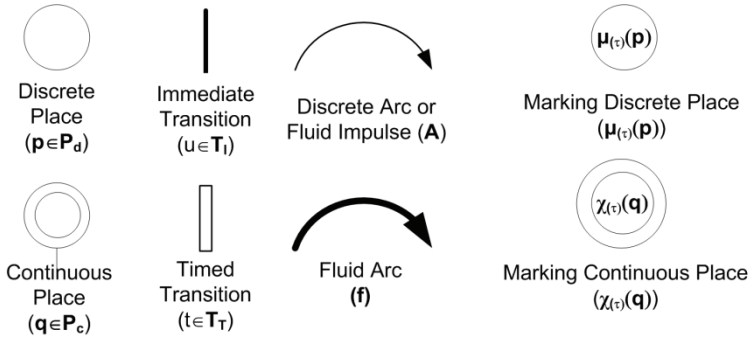



Fig. 1. Graphical representation of FSPN elements

According to Gribaudo et al. (2003), it is possible to unify stochastic and formal methods using FSPN to modeling and analyzing systems with discrete and continuous parts. Considering the evolution of FSPN from elementary Petri nets, FSPN can be used for modeling system functional properties related to their discrete states, such as conformity and reachability states. Considering the evolution of FSPN from stochastic Petri nets (SPN), it is possible to use FSPN to perform stochastic analysis obtaining performance, reliability, availability and safety measures (dependability measures) based on the systems uncertainty parameter. Finally, FSPN is capable of modeling continuous state spaces interdependent of discrete process, resulting in a useful modeling formalism for real systems.

A systematic way to define measures of interest in a net, including models in FSPN, is using *reward functions*, where *reward structures* can be represented, at a high abstraction level, with conditional expressions (German, 2000). Union of Stochastic Petri Nets (SPN) and Reward Structures formalism are named Stochastic Reward Nets (SRN) (Muppala et al., 1994). In SRN, measures are obtained by reward variables, as instantaneous reward (Equation 2), accumulated reward (equation 3) and average reward (equation 4) variables.

$$R_{inst}(t) = \sum_{n \in S} rr_n . 1_{\{N(t)=n\}} + \sum_{g \in T} \sum_{n \in S} ri_n^g . 1_{\{g \ \ fires \ \ at \ \ t \ \ in \ \ state \ \ n\}} . \delta \ , \tag{2}$$

$$R_{acum}(t) = \int_0^t R_{inst}(\varphi) d\varphi \tag{3}$$

$$R_{aver}(t) = \frac{1}{t} . R_{acum}(t) \tag{4}$$

$R_{inst}(t)$, $R_{acum}(t)$ and $R_{aver}(t)$ present stochastic behavior if their impulsive reward vector (ri) is not considered. They can thus be characterized by their probability distributions. According to Muppala et al. (1994), output measures from a Stochastic Reward Net (SRN) can be represented by expected values of reward variables. Therefore, considering $E[R_{acum}(t)]$ the expected value for reward variable $R_{acum}(t)$ (German, 2000):

$$E[R_{acum}(t)] = \sum_{n \in S} rr_n . \int_0^t \pi_n(x) dx \ , \text{ where } \pi_n(t) = Pr(N(t) = n), \ n \in S. \tag{5}$$

$E[R_{inst}(t)]$ is calculated as a consequence of (5), where:

$$E[R_{inst}(t)] = \frac{d}{dt} E[R_{acum}(t)] = \sum_{n \in S} rr_n . \pi_n(t) \tag{6}$$

Getting quantitative measures from Stochastic Petri nets (and their evolutions) using reward structures demands the calculation of state probabilities $\pi_n(t) = Pr(N(t) = n)$. Vector $\pi(t) = (\pi_1(t), \pi_2(t), ..., \pi_m(t))$, $S = \{1, 2, ..., m\}$ can be calculated using 2 possible ways: *Analytical Techniques* and *Simulation* (Computer Science Department, 2000). Applying *analytical techniques* to solve stochastic Petri nets (and their evolutions) requires nets with Markovian (memoryless) characteristics. This fact narrows modeling capabilities, restricting net transitions to exponential and geometric fire time probability distribution only. Besides constraints related to transition types, complex and non-limited nets can be unsolvable by analytical techniques because of the large number of states they produce.

*Simulation* is a better choice for solving non-Markovian Petri nets; solving complex and non-limited nets that state space size is analytically unsolvable. Once stochastic Petri nets are a formalism applied to describe dynamics of discrete systems, where state evolution (mark changing) occurs in discrete and well defined events, nets solution by *Discrete Event Simulation* (DES) is an extension of the Petri nets formalism. However, the simulation technique allows only doing an inference of values of interest. In this way, quantitative analysis using simulation techniques, mainly DES, must incorporate statistical inference concepts as a way of evaluating values with acceptable confidence levels. Therefore, it is

necessary to have good statistical estimators to calculate values (5) and (6) to use the simulation technique.

Fig. 2 summarizes the process of obtaining measures of interest in the Assessment Framework, where FSPN, DES, SRN and Statistical Inference are put together as the way to assess the characteristics and properties found in the current dependable critical systems. The main concepts of FSPN, SRN and DES used in the framework are detailed in the following items.

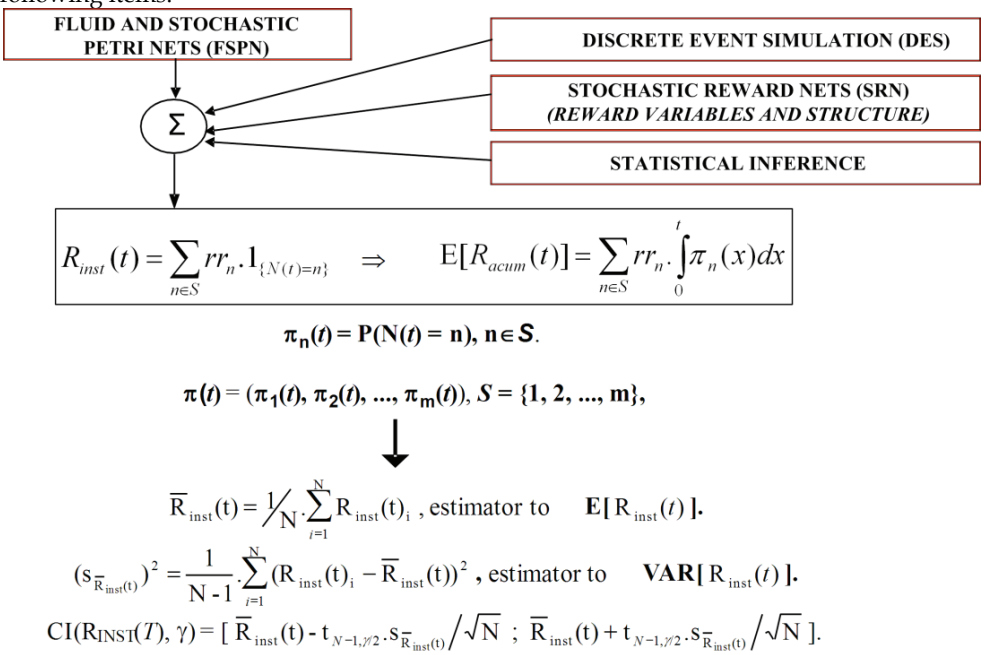



Fig. 2. Obtaining measures of interest using the Assessment Framework

### 2.2 The Fluid and Stochastic Petri nets and its ability to represent Hybrid Systems

#### 2.2.1 Petri nets and their evolution in the assessment of systems

Petri nets (PN) are a graphical and mathematical formalism used in the specification, modeling, verification and analysis of systems properties. Particularly, PN are used to model complex systems that have concurrent, parallel, asynchronous, distributed and/or non-deterministic processes (Murata, 1989), and to assess availability, performance and other systems characteristics (Computer Science Department, 2000; Bobbio, 1988). For example, the Petri nets have a wide application in modeling computer systems (Peterson, 1989 apud Labeau et al., 2000), considering the relationship between Hardware, Software and Human-Machine Interface in a systemic context (Leveson & Stolzy, 1987).

However, the elementary structure of a Petri Net is restricted to deal with real systems. For example, an elementary PN can not assess whether a place has no tokens (test of zero). Nor is it possible to relate the cardinality of the arcs with the markings of the net (marking-dependent arcs). Another elementary PN limitation is not solving the firing of two or more transitions in conflict. Thus, as a way of increasing the power of modeling, some structural

extensions have been created in the basic concept of PN, such as guards, priority, arc multiplicity and marking-dependent arc cardinality.

An important extension to Petri nets is considering, beyond immediate firing transitions, firing time to the transitions, allowing the use of PN in the assessment of important aspects of systems, such as availability, performance, reliability, safety and other measures based on time. At first, 2 extensions were adopted in the timed Petri nets concept (Murata, 1989): the deterministic-timed Petri nets and the Stochastic Petri nets (SPN). In both cases, time delays ($\tau$) were associated with transitions (T) so that, if $\mathbf{t} \in \mathbf{T}$ is timed with $\tau_t \in \mathbf{R}^+$, and $\delta(\mu_k, t_k) = \mu_{(k+1)}$ (where $\delta : \mathbf{N}^n \times \mathbf{T} \rightarrow \mathbf{N}^n$ is the *next state function* and $\mu: \mathbf{P} \rightarrow \mathbf{N}$ is the *marking function* that maps the set of places P in a Petri net to non-negative integers), the transition from state k to state k+1 ($\mu_k \rightarrow \mu_{(k+1)}$) will consume a time $\tau_{t,k}$ between enabling the transition $\mathbf{t}$ ($\mathbf{t} \in H(\mu_k)$), where $H(\mu) \in \mathbf{T}$ is the set of enabled transitions in the marking $\mu$ of the Petri net), and its effective firing event ($\uparrow \mathbf{t}$). The difference between these extensions is the category of delay (variable) used. In the deterministic-timed Petri nets, $\tau_t$ is deterministic (not a random variable). In the case of Stochastic Petri nets (SPN), the set of time delays is created by random variables $\tau_t : \Omega \rightarrow R$ defined in a space of probability ($\Omega$, P) with probability density function $F\tau_t(x) = P(\tau_t < x)$.

When time delays are introduced in Petri net transitions, marking (states) are represented by $s_k = (\mu_k , \tau_{t,k}) \in \mathbf{N}^{\#P} \mathbf{x} (\mathbf{R}^+)^{\#T}$. Thus, a timed Petri net has a firing sequence $\mathbf{T}\uparrow = (t_0, t_1, \ldots , t_n)$ that produces a marking sequence ($s_0 , s_1 , \ldots, s_n$) associated to the events of time ($e_0 , e_1 , \ldots, e_n$), where ($e_{k+1} - e_k$) = $\tau_{t,k}$. Hence, the reachability set S to a timed Petri net is the set of markings defined by $s_k \in S$.

Due to the dynamic execution of SPN having stochastic characteristic related to the time (stochastic transitions), the firing dynamics can be represented by a stochastic process (German, 2000). A stochastic process is formally defined as a family of random variables $\{\mathbf{N}(t), t \in \Upsilon\}$, where $\mathbf{N}(t)$ is defined on a space of probability $S=(\Omega, \mathbf{P})$ and indexed by the parameter $t$, $t$ varying on a set *index* $\Upsilon$. For an SPN, we have $\mathbf{N}(t) \in S$, where $S$ is the set of tangible states of an SPN and $t$ is the parameter of time, which can be both discrete ($t = e_k = e_0 + \sum_{t \in \sigma, n=[0,k]} \tau_{t,n}$) and continuous ($t \in \mathbf{R}^+$), describing the relationship between the states of a stochastic process at different time instants.

Stochastic Petri nets with all their transitions associated with exponential distribution functions are called Generalized Stochastic Petri nets (GSPN), and due to their memoryless property they can be represented by a Markovian process. In this case, the dynamic behavior of markings are mapped into a *time-continuos homogeneous Markov chain* with an isomorphic state space to the reachability tree of the GSPN (Bobbio, 1988). Thus, when a net is memoryless (a Markovian net), one can ignore the time delay ($\tau_t$) of enabled transitions at $\mu_{(k)}$ that were not fired in this marking (state) and, after any transition firing, became disabled at marking $\mu_{(k+1)}$. Moreover, if a net has transitions with distribution functions different from an exponential, the SPN can not be represented by a Markovian process. Besides, it is not possible to use the properties of a Markov chain; the execution dynamics of a SPN must establish what policy of time resampling to $t_q$ must be used when $\{t_j, t_q\} \in H(\mu_k)$, $\delta(\mu_k, t_{jk}) = \delta(\mu_k, t_{qk}) = \mu_{(k+1)}$ and $t_q \notin H(\mu_{(k+1)})$, but $t_q \in H(\mu_{(m > k+1)})$. 3 resampling policies are defined: PRD (preemptive repeat different), PRI (preemptive repeat identical) and PRR (preemptive repeat resume). PRD is the policy used in a Markovian model (memoryless),

where a new time delay to $\mathbf{t_q}$ is resampled when it is enabled again. PRI uses the same sampled time delay at the last $\mathbf{t_q}$ enabling event. PRR uses the remaining time delay at the last $\mathbf{t_q}$ disabling event with no $\mathbf{t_q}$ firing.

### 2.2.2 The FSPN and the Hybrid Systems (HS)

The Stochastic Petri nets (SPN) can model and assess dynamic systems with both discrete state spaces and stochastic properties by incorporating characteristics of elementary Petri nets and stochastic temporal characteristics. However, a large segment of real world systems is hybrid, where continuous processes (which contain real values) are controlled by discrete logic processes (such as computers). The control system of an aircraft, the manufacturing processes, the communication networks and intelligent systems are examples of real-world hybrid systems (Tuffin et al., 2001).

According to Varaiya (1999) apud Tuffin et al., (2001), a Hybrid System (HS) consists of 2 interdependent components: a continuous component, $\mathbf{q \in R^n}$, and an event-driven finite state space $\mathbf{p \in Q}$. Given a state $\mathbf{(q_k, p_k)}$, only the continuous state evolves according to the set of differential equations $dq(t)/dt \in F(q(t), p_0) \subset R^n$, during the time interval $\mathbf{[t_k, t_{k+1})}$. At time $\mathbf{t_{k+1}}$, there is $\mathbf{q(t_{k+1}) \in G(p_k, p_{k+1})}$, where **G** is an "enabling zone" that checks the permission of migration for the discrete state $\mathbf{p_k}$ to $\mathbf{p_{k+1}}$. If the migration is allowed, there is an instantaneous state migration in the discrete model ($\mathbf{p_k}$ para $\mathbf{p_{k+1}}$) and the continuous state is brought to $\mathbf{q_{k+1}}$ according to the relation $\mathbf{(q_{(t_{k+1})}, q_{k+1}) \in R(p_k, p_{k+1}) \subset R^n x R^n}$, where **R()** is defined for each pair ($\mathbf{p_k}$ , $\mathbf{p_{k+1}}$) in the discrete model. The dynamics of the model evolves in the same way for all the horizon time.

Thus, the continuous part of a hybrid system is modeled by differential equations, while the discrete events evolves in temporal events (Computer Science Department, 2000), and both are related through the mechanism explained above. The formalism of the Fluid and Stochastic Petri nets (FSPN) embodies the representation of continuous (fluid) places and arcs to the characteristics of Stochastic Petri Nets (SPN), where the evolution of continuous values is represented by fluid rate functions (related to fluid arcs), and the variation of discrete states follows the same rules of Stochastic Petri Nets. Primarily, the concept of FSPN introduced by Trivedi & Kulkarni (1993) intended to solve problems of state explosion in the reachability trees of nets, avoiding the accumulation of marks (tokens) in discrete places. Therefore, a FSPN was not defined for modeling hybrid systems.

Still relating the differences between formalisms (FSPN and HS), and due to the legacy of SPN, the FSPNs are focused on the evaluation of measures in stochastic processes involved in the modelled system. The formalism of hybrid systems (HS) focuses on system reachability and decision analyses (model of control). The influence of uncertainties in the HS modeling is represented by the boundaries of the states in model. Even with these differences in formalism and application, Tuffin et al., (2001) proved that hybrid systems can be represented by the Fluid Stochastic Petri nets.

As shown by equation (1), a FSPN can be formally defined by a 13-upla as FSPN = (P, T, A, f, Π, g, Dist, r, Afet, $w$, b, $\mu_0$, $\chi_0$), where:

$P = P_d \cup P_c = (p_1, p_2, ..., p_{\#Pd}) \cup (q_1, q_2, ..., q_{\#Pc})$, $P_d \cap P_c = \varnothing$.

$\mu_{(t)}(p_i) \in N$, $\forall p_i \in P_d$ is a marking (state) in discrete net at time $t$, $t \in R^+$,

$\chi_{(t)}(q_i) \in R^+$, $\forall q_i \in P_c$ is a marking (state) in continuous net at time $t \in R^+$,

$T = T_T \cup T_I = (t_1, t_2, ..., t_{\#Tt}) \cup (u_1, u_2, ..., u_{\#Ti})$, $T_T \cap T_I = \varnothing$.

$A = A_d \cup A_c = ( ((P_d \; x \; T) \cup (T \; x P_d)) \; x \; \hat{S} \rightarrow N ) \cup ( ((P_c \; x \; T_I) \cup (T_I \; x P_c)) \; x \; \hat{S} \rightarrow R^+ )$, where:

$A_d = I_d \cup O_d \in (N^{\#Pd} \rightarrow N)^{(\#Pd \; x \; \#T)} \cup (N^{\#Pd} \rightarrow N)^{(\#T \; x \; \#Pd)}$: marking-dependent cardinality of input and output arcs (I e O) between discrete places $P_d$ and transitions T.

$A_c = I_c \cup O_c \in (N^{\#Pc} \rightarrow N)^{(\#Pc \; x \; \#Ti)} \cup (N^{\#Pc} \rightarrow N)^{(\#Ti \; x \; \#Pc)}$: marking-dependent *fluid impulse* of input and output arcs (I e O) between fluid (continuous) places $P_c$ and the immediate transitions $T_I$.

$f = ((P_c \; x \; T_T) \cup (T_T \; x P_c)) \; x \; \hat{S} \rightarrow R^+)$: marking-dependent *fluid rate.*

$\Pi \in N^{\#Ti}$ is the *priority vector* of immediate transitions $T_I$.

$Dist \in T_T \; x \; \hat{S} \rightarrow F$ are the marking-dependent firing time distributions of transitions $T_T$ (where *F* is the set of positive distribution functions $[0, \infty[$).

$r \in ( \{PRD , PRS , PRR\} )^{\#Tt}$ are the resampling policies to the transitions $T_T$.

$Afet : T \rightarrow T_T \in (\{PRD , PRS , PRR\} \; x \; \{PRD , PRS , PRR\})^m$ is the resampling policy that a transition $t_i \in T_T$ must adopt on marking $\mu_{(k+1)}$ where $\{t_i, t_j \cup u_j\} \in H(\mu_k)$, $\delta(\mu_k, t_{ik}) = \delta(\mu_k, t_j \cup u_j) = \mu_{(k+1)}$ e $t_i \in H(\mu_{(k+1)})$. This property is named *afectation*.

$w \in N^{\#T} \rightarrow R^+$ are the marking dependent weigths of transitions T.

$b: P_c \; x \; N^{\#Pd} \rightarrow R_0$ are the fluid limits to each fluid (continuous) place $P_c$.

$g : (\hat{S} \rightarrow \{True = 1, False = 0\})^{\#T}$ is a guard vector *of* T.

$\mu_0 = \mu_{(t=0)}(p_i), \; \forall p_i \in P_d$: marking from discrete net at $t = 0$ (*discrete initial marking*).

$\chi_0 = \chi_{(t=0)}(q_i), \; \forall q_i \in P_c$ e $\chi(q_i) \leq b_q$: marking from discrete net at $t = 0$ (*continuous initial marking*).

$S = (\mu, \chi)_{(t)} \subseteq \hat{S} = N^{\#Pd} \; x \; (R^+)^{\#Pc} \; x \; R^+$,where $\hat{S}$ is the potential state space and *S* is the effective state space of FSPN. In the same way as SPN, FSPN can be represented by a stochastic process $\{N(t) \in \hat{S}, \; t \in R^+\}$.

Regarding the FSPN execution dynamics, a transition $\mathbf{t \in T}$ is enabled in a marking $\mathbf{(\mu, \chi)_{(t)} \in \hat{S}}$ at time $\boldsymbol{t \in R^+}$, $\mathbf{t \in H((\mu, \chi)_{(t)})}$, for the same conditions of a Stochastic Petri net, i.e.:

$$A_{pi,t}((\mu, \chi)_{(t)}) \leq \mu_{(t)}(p_i) \vee f_{qi,t}((\mu, \chi)_{(t)}) \leq \chi_{(t)}(q_i) \; , \; \forall p_i \in P_d, \; \forall q_i \in P_c. \tag{7}$$

and

$$g_t((\mu, \chi)_{(t)}) = 1. \tag{8}$$

Similarly, for $\mathbf{t \in H((\mu, \chi)_{(t)})}$, a marking $\mathbf{(\mu, \chi)_{(t)}}$ is considered *immediate* when $\mathbf{t \in T_I}$, and a marking $\mathbf{(\mu, \chi)_{(t)}}$ is considered *tangible* when $\mathbf{t \in T_T}$.

Given $\mathbf{t \in H((\mu, \chi)_{(t = k)})}$, $\mathbf{t \in T}$, marking $\mathbf{(\mu, \chi)_{(t = k)}}$ is changed to the state $\mathbf{(\mu, \chi)_{(t = k+\tau)}}$ in 2 ways (Computer Science Department, 2000) (where $\boldsymbol{\tau}$ is the time between the event $\mathbf{t \in H((\mu, \chi)_{(t=k)})}$ and $\uparrow \mathbf{t}$, i.e. the timing of **t** at $\boldsymbol{t = k \in R^+}$:

- **1st way:** with respect to *discrete arcs* ($\mathbf{A_d}$) and *fluid impulses* ($\mathbf{A_c}$), $\mathbf{(\mu,\chi)_{(t = k)}}$ changes in form:

$$\mathbf{\mu_{(t = k+\tau)}(p) = \mu_{(t = k)}(p) + A_{t,p}((\mu, \chi)_{(t = k)}) - A_{p,t}((\mu, \chi)_{(t = k)})}, \; \forall p \in P_d \tag{9}$$

$$\mathbf{\chi_{(t = k+\tau)}(q_i) = \min( b_q(\mu_{(t = k+\tau)}) \; ; \; \max( 0 \; ; \chi_{(t = k)}(q) + A_{t,q}((\mu, \chi)_{(t = k)}) - A_{q,t}((\mu, \chi)_{(t = k)})) )}, \; \forall q \in P_c \tag{10}$$

- **2nd way:** with respect to *fluid arcs* **f**, if the marking $\mathbf{(\mu, \chi)_{(t = k)}}$ is *tangible*, the fluid level flow continuously through arcs **f** enabled by transitions $\mathbf{t_i \in T_T}$ until $\mathbf{t_i \notin H((\mu, \chi)_{(t = k)})}$.

Considering $\delta_q^{pot}((\mu,\chi)_{(t=k)}) = \sum_{t\in\Lambda}\left(\mathbf{f}_{t,q}(\mu,\chi)_{t=k} - \mathbf{f}_{q,t}(\mu,\chi)_{t=k}\right)$, where $\Lambda = H((\mu,\chi)_{(t=k)})$, as the *potential rate of change of fluid level* on state $(\mu, \chi)_{(t = k)}$. The fluid level of $q\in P_c$, during the state $(\mu,\chi)_{(t = k)}$, changes continuously as:

$$\frac{d\chi_q}{dt} = 0, \text{ if } ( b_q((\mu, \chi)_{(t = k)}) = 0 ) \vee ( ( \chi_{(t = k)}(q) = b_q((\mu, \chi)_{(t = k)}) ) \wedge ( \delta_q^{pot}((\mu,\chi)_{(t = k)}) > 0) ) \vee ( \chi_{(t = k)}(q) = 0 \wedge \delta_q^{pot}((\mu,\chi)_{(t=k)}) < 0 ). \quad (11)$$

$$\frac{d\chi_q}{dt} = \delta_q^{pot}((\mu, \chi)_{(t= k)}), \text{ to the other cases.} \quad (12)$$

The stochastic evolution of the FSPN markings follows the same dynamic execution rules of a SPN, unless a transition $\mathbf{t_i}\in H((\mu, \chi)_{(t = k)})$ was disabled because $\chi_{(t = k)}(\mathbf{q_i})$ achieved a value before $\mathbf{t_i}$ fires. Moreover, the evolution of the discrete markings of the net occurs in an atomic way, in the same way as the SPN.

### 2.3 Improving the quantitative assessment with Reward nets

The Stochastic Petri Nets (SPN) and their extensions, such as FSPN, allow obtaining relevant system measures, such as availability, performance, reliability, safety and other measures based on time. The stochastic measures of a SPN is obtained by calculating the probability of occurrence of the system states (Bobbio, 1988), where $\boldsymbol{\pi(t) = (\pi_1(t), ..., \pi_m(t))}$, $\boldsymbol{S = \{1, ..., m\}}$ , and $\boldsymbol{\pi_n(t) = P(N(t) = n), n\in S}$ (probability of transient state, which represents the probability of the net being in the marking $\mathbf{n}$ at time $\boldsymbol{t}$, $N(t) \in \hat{S}$, $t \in R^+$). Other measures are obtained by the stochastic analysis of the model elements, such as the frequency of transition firing or the number of tokens in one place at time t. However, only these sets of measures are available to achieve with Stochastic Petri Nets (SPN) and their extensions.

A systematic way to define other measures of interest in a net other than those listed above is through the *reward function*, where the reward structures can be expressed, at a high abstraction level, by conditional marking-dependent expressions (German, 2000). The net formalism which allows specifying output measures with reward-based functions are defined as "Reward Nets". The fusion between the Stochastic Petri Nets and the reward formalism is defined as Stochastic Reward Nets (SRN) (Muppala et al., 1994). In the stochastic process level, a reward structure is defined by a *reward rate vector* "rr" (where $rr_n\subset$rr, state $n\in S$ and S is the net state space), and an *impulsive reward vector* "ri" (where $ri_{g_n}\subset$ri, transition $g\in T$ (transition set of net) in state $n\in S$) (German, 2000). As was presented before, 3 reward variables could be defined in a given reward structure: instantaneous reward (equation 2), accumulated reward (equation 3) and average reward (equation 4) variables. Considering Equations (2)(3)(4):

$R_{inst}(t) = \sum_{n\in S} rr_n . 1_{\{N(t)=n\}} + \sum_{g\in T}\sum_{n\in S} ri_n^g . 1_{\{g \text{ fires at } t \text{ in state } n\}} . \delta$ is an instantaneous reward variable at time $t$, where $\mathbf{1_{\{predicate\}}}$ returns 1 if "{predicate} = TRUE", and returns 0 in the other cases; $\mathbf{N(t) \in S}$ is the stochastic process defined to the net in assessment; $\boldsymbol{\delta}$ is a Dirac

impulsive function, where $\boldsymbol{\delta(t-\tau)} = 0$ to $\forall\ \boldsymbol{t \neq \tau}$ e $\boldsymbol{\delta(t-\tau)} = \infty$ to $t = \boldsymbol{\tau}$, where $\int_{-\infty}^{+\infty} \delta(x)dx = 1$, the area of which is given by $\mathbf{rig_n}$ at firing of **g**.

$R_{acum}(t) = \int_0^t R_{inst}(\varphi)d\varphi$ is an *accumulated reward variable* at *t*.

$R_{aver}(t) = \frac{1}{t}.R_{acum}(t)$ is an *average reward variable* at *t*.

The variable $R_{inst}(t)$ has no stochastic behavior due to the presence of impulses (δ) (German, 2000). However, variables $R_{acum}(t)$ and $R_{aver}(t)$ have stochastic behavior and can be characterized by their probability distributions. As mentioned by Muppala et al., (1994), the SRN output measures are expressed in terms of expectation (**E[]**) of the reward variables, defined as the amount of interest in the analysis.

Therefore, considering $\mathbf{E[R_{acum}(t)]}$ as the expectation of *accumulated reward variable* $R_{acum}(t)$ (German, 2000):

$$\mathrm{E}[R_{acum}(t)] = \sum_{n\in S} rr_n.\int_0^t \pi_n(x)dx + \sum_{g\in T}\sum_{n\in S} ri_n^g.\int_0^t \varphi_n^g(x)dx \tag{13}$$

Where $\pi_n(t)$=P(N(*t*)=n), n∈*S*, is the probability of transient state, which represents the probability of the net being in the marking n at time *t*, given that $\mathrm{E}[1_{\{N(t)=n\}}] = \pi_n(t)$; and $\int_0^t \varphi_n^g(x)dx$ = E [number of fires of g∈T at marking n∈*S* and during the period [0 a *t*)].

Considering $\mathbf{E[R_{aver}(t)]}$ as the expectation of the *average reward variable* $\mathbf{R_{aver}(t)}$:

$$\mathrm{E}[R_{med}(t)] = \mathrm{E}[\frac{1}{t}.R_{acum}(t)] = \frac{1}{t}.\mathrm{E}[R_{acum}(t)] \tag{14}$$

For the *instantaneous reward variable* $\mathbf{R_{inst}(t)}$, it is possible to calculate its expectation if $\mathbf{R_{inst}(t)}$ does not contain impulses (δ). Therefore, obtaining $\mathbf{E[R_{inst}(t)]}$ through $\mathbf{R_{acum}(t)}$, we have:

$$\mathrm{E}[R_{inst}(t)] = \frac{d}{dt}\mathrm{E}[R_{acum}(t)] = \sum_{n\in S} rr_n.\pi_n(t) + \sum_{g\in T_I}\sum_{n\in S} ri^g\varphi_n^g(t) \tag{15}$$

So as to simplify the equating, the definition of the $\mathbf{rig_n}$ (impulsive reward vector) can be supressed from a net under consideration. Thus, making $\mathbf{rig_n = 0}$, $\forall \mathbf{g\in T}$, $\forall\ \mathbf{n\in S}$, we have:

$$R_{inst}(t) = \sum_{n\in S} rr_n.1_{\{N(t)=n\}} \Rightarrow \mathrm{E}[R_{acum}(t)] = \sum_{n\in S} rr_n.\int_0^t \pi_n(x)dx \tag{16}$$

And the other variables are calculated as a consequence of the definition above.

The analysis of a net using reward equations requires the calculation of its probability of state $\boldsymbol{\pi_n(t) = P(N(t) = n)}$, $\mathbf{n\in S}$. Redefining it as the vector $\boldsymbol{\pi(t)} = (\boldsymbol{\pi_1(t), \pi_2(t), ..., \pi_m(t)})$, $\boldsymbol{S = \{1, 2, ..., m\}}$, if $\boldsymbol{\pi} = \lim_{t\to\infty} \pi(t)$ exists, where $\boldsymbol{\pi}$ is considered the *limit state probability*. If this probability does not exist, the definition $\boldsymbol{\pi_n} = \lim_{t\to\infty} \frac{1}{t}\int_0^t \pi_n(x)dx$ can be used, where $\boldsymbol{\pi_n}$ is the time-averaged limit probability to a state $\mathbf{n\in S}$. However, if both $\boldsymbol{\pi}$ and $\boldsymbol{\pi_n}$ exist, both probabilities have the same value, and both can be interpreted as the fraction of time in which a process remains in a state when it is observed for a long period of time (German, 2000).

Based on the presented reward structure and on the possible calculations over the state probabilities, both the transient analysis as the stationary analysis can be performed on nets. In the transient analysis, the state of the system can be observed at specific instants of time. In stationary analysis, the behavior of the system can be obtained when it is in stable operation (steady-state behavior).

### 2.4 Resolving FSPN models with Discrete Event Simulation

#### 2.4.1 Analytical Techniques versus Discrete Event Simulation

As could be noticed before, obtaining quantitative measurements for Stochastic Petri Nets (and their evolutions) involves calculating the probability of the state vector $\boldsymbol{\pi_n(t)=P(N(t)=n)}$. There are 2 different techniques for solving these nets: Analytical Techniques and Simulation (Computer Science Department, 2000).

Using analytical techniques for the solution of Stochastic Petri Nets (SPN) assumes that the nets have Markovian characteristics (memoryless systems), which restricts the use of transitions with exponential and geometric distribution functions to the nets. To these memoryless nets, it is possible to transform them into a Continuous Time Markov Chain (CTMC) or into a Discrete Time Markov Chain (DTMC), and solving them using the properties of the Markov theory, e.g., calculating the probability of the state equations using its transition probability matrix **P**, where $\boldsymbol{\pi(k) = \pi(0).P^k}$ (considering a DTMC) or $\boldsymbol{\pi(t) = \pi(0).e^{Pt}}$ (considering a CTMC) (German, 2000).

In addition to the restrictions of the types of transition used in SPN, the modeling of complex and non limited nets may result in an excessive number of states, making the computational resolution of their state equations unfeasible and forcing the application of numerical approaches and simplifications to the Analythical Technique. Related to this states space problem is the resolution of the Fluid and Stochastic Petri nets (FSPN) by analytical techniques. The proposed techniques in Trivedi & Kulkarni (1993) and Horton et al (1998) are feasible only to certain strict conditions of FSPN. Because their reachability and characteristics of their decision processes, FSPN of up to 2 dimensions can be calculated (Asarin, Maler & Pnueli (1995) apud Ciardo, Nicol & Trivedi (1999)).

The *simulation* technique is a feasible solution to the resolution of non-Markovian Petri nets, as well the solution of nets the complexity of which makes its state space unsolvable by analytical methods. Since Stochastic Petri Nets (SPN) is a method used to describe the dynamics of discrete event systems, the evolution of states (markings) of which occurs in well defined and discrete events, its resolution by simulation, mainly *discrete event simulation,* becomes an extension of this method.

According to Gordon (1978), the analysis of a system by Discrete Event Simulation (DES) is modeled by a set of elements called State Descriptors $\boldsymbol{(D_e)}$ - which represents the system states $\boldsymbol{(d_i \in N,\ \forall i \in D_e)}$ - and by an element that represents the *simulation time* $\boldsymbol{(t)}$, so that $\boldsymbol{t \in T_S}$ and $\boldsymbol{T_S \in [0, T_{MAX}]}$), the goal of which is representing the time evolution of the simulation and identifying the moment in which changes in the system state occur.

The state descriptors values $\boldsymbol{(d_i,\ \forall i \in D_e)}$ are changed in discrete events $\boldsymbol{(e_n)}$, so that $\boldsymbol{e_n \in E}$ and $\boldsymbol{n = \{1, 2, ...\}}$, where ***E*** is the set of conditions that promote immediate change in one or more values of the state descriptors $\boldsymbol{(d_i)}$. The simulation dynamics occurs by changes in the values of state descriptors $\boldsymbol{(d_i)}$ for each event $\boldsymbol{(e_n)}$ runned until the time specified for simulation ($\boldsymbol{T_{MAX}}$) is fully covered or until there are no more conditions for the occurrence of events.

The system events occur in chronological order, generating a sequence of executions $(e_0, e_1, ..., e_n)$ at instants $(t = t_0, t_1, ..., t_n)$, where $(t_0 \le t_1, t_1 \le t_2, ..., t_{n-1} \le t_n \le T_{MAX})$.
Table I shows the equivalence between the Discrete Event Simulation (DES) and Stochastic Petri nets (SPN) concepts, considering the characteristics of time and state for a modeled system.

| | **Discrete Event Simulation (DES)** | **Stochastic Petri nets (SPN)** |
|---|---|---|
| ***State*** | *State Descriptor values* $(d_i \in N, \forall i \in D_e)$ | *Marking on nets* $(\mu(k) \in N, \forall k \in P)$ |
| ***Time*** | *Simulation Time* $(t \in T_S)$ **(instants** $t_0, t_1, ..., t_n$**)** | *Firing time* $(e_0, e_1, ..., e_m)$ to the transitions $(t_0, t_1, ..., t_n)$ |

Table I – Equivalence between DES and SPN models

As an example, given a SPN with initial marking $\mu_0$, which occurs in the firing sequence $(t_0, t_1, ..., t_n) \in \sigma$, promoting the sequence of markings $(\mu_0, \mu_1, ..., \mu_n) \in \mu$ and the spent time in each marking is $(\tau_{t0,0}, \tau_{t1,1}, ..., \tau_{tn,n})$, corresponding to the notation $(s_0, s_1, ..., s_n) = ((\mu_0, \tau_{t\,0,0}), (\mu_1, \tau_{t1,1}), ..., (\mu_n, \tau_{tn,n}))$, the SPN for a DES could be written as:

| **SPN dynamics** | **Discrete Event Simulation** |
|---|---|
| $(s_0, s_1, s_2, ..., s_n) =$ | . |
| $= ((\mu_0, \tau_{t,0}), (\mu_1, \tau_{t,1}), ..., (\mu_n, \tau_{t,n})) =$ | . |
| $= ((\mu_0, e_0), (\mu_1, e_1), ..., (\mu_n, e_n)) =$ | . |
| $= ((\mu_0, 0), (\mu_1, \tau_{t,0}), ..., (\mu_n, \sum_{i=0}^{n-1} \tau_{t,i}))$. ⟶ | $((d, t_0 = 0), (d, t_1), ..., (d, t_n))$. |

Given that the system state at instant $t$, modeled for Discrete Event Simulation (DES) assessment, is represented by the set of values of state descriptors $(d = (d_0, d_1, ..., d_n))$ in an instant $t \in T_S$, resulting in the 2-upla $(d, t)$, the sequence of markings $\mu_k$ obtained by the SPN running in the instants of time $e_k$ is equivalent to the evolution of the values of $d$ occurred in the instants $t_k$, where $((\mu_0, 0), (\mu_1, e_1 = 0 + \tau_{t,0}), ..., (\mu_n, e_n = e_{n-1} + \tau_{t\,,\,n-1})) = ((d, t_0), (d, t_1), ..., (d, t_n))$.
For a Petri net, the set of conditions $(E)$ to the Discrete Event Simulation model is obtained by means of the execution rules related to that Petri net. As an example, the discrete event simulation process to a DTMC (Discrete Time Markov chain) $\{N(k) = s \mid s \in S = \{s_1, s_2, ..., s_n\}, k \in N\}$ with a transition matrix $P$ can be executed, according to Ripley (1987):

If $N(n) = s$, choose $N(n+1) = i$ using the discrete distribution $\{p_{s,i} \in P \mid i \in S\}$.

In the case of a Stochastic Petri net modeled by transitions with any probability distribution, it can be simulated by means of the following algorithm:

1. Given $(\mu_k, e_k)$, get $t_i \in H(\mu_k)$, $\forall t_i \in T$.
2. Sample $\tau_{ti}$ to $\forall t_i \in H(\mu_k)$ by means of their probability distribution functions.

3. Respecting the net execution rules, make $\delta(\mu_k, t_{ik}) = \mu_{k+1}$ and $e_{k+1} = e_k + \tau_{ti}$.
4. Return to step "1" and repeat the procedure to the next reached state ($\mu_{k+1}$) and while $\exists t_i \in H(\mu_{k+1}), \forall t_i \in T \vee e_{k+1} \leq T_{MAX}$.

The temporal representation of a Discrete Event Simulation (DES) process (the value of the *simulation time* element) may change according to 2 different methods (Gordon, 1978):

**Event-oriented**: the value of the *simulation time* advances to the instant in which the next detected event happens. This process is known as the *next event simulation*, due to both the next state and its time of occurrence are simulated from the current state of the model.

**Interval-oriented**: the value of *simulation time* advance at time intervals (usually uniform and short intervals), and where the existence of a condition is verified for a migration of state.

According to Gordon (1978), each method of temporal evolution in DES is appropriate for a particular type of system to be simulated. The event-oriented method is recommended for the simulation of purely discrete systems, such as the systems modeled by Stochastic Petri nets. On the other hand, the event-oriented method is recommended for the simulation of purely continuous systems, due to the evolution of states in the time being modeled by differential equations.

### 2.4.2. Simulation of Continuous-Discrete (hybrid) models

In addition to the purely discrete systems and the purely continuous systems, there are systems in which the mechanism of change of states regarding time occurs in a discrete-continuous way. The Fluid and Stochastic Petri Nets (FSPN), as defined above, are a method for modeling and evaluating continuous-discrete systems. In these systems, the states can change either in a continuous way, in a discrete way or in a continuous way added by discrete jumps.

The process of simulating continuous-discrete systems depends on the modes of interaction between the continuous and the discrete model variables. According to Computer Science Department (2000), there are 3 possible modes of interaction between $\mu(p)$ and $\chi(q)$, where $p \in P_d$ and $q \in P_c$ in a FSPN:

1. Discrete variations in $\chi(q)$ dependent on $\mu(p)$: continuous variables $(\chi(q))$ can receive discrete changes in their values. In the FSPN, this is equivalent to modelling fluid impulses, represented by arcs $A_c$.
2. Variations in $\chi(q)$ and/or $\mu(p)$ dependents of $\chi(q)$: an event that changes both discrete variables $(\mu(p))$ and continuous $(\chi(q))$ may be executed if the value of continuous variables reach pre-defined levels. In FSPN, this is equivalent to modelling (fluid) marking-dependent *guards* for immediate transitions $(g:((q \in P_C) \rightarrow \{1, 0\})^{\#Ti})$.
3. Continuous variation rate in $\chi(q)$ dependent on $\mu(p)$: a discrete event in time can promote the change in the functional description of continuous variables. In FSPN, this is equivalent to modelling marking-dependent *fluid rates* $(f=((P_c \, x \, T_T) \cup (T_T \, x P_c)) x \hat{S} \rightarrow R^+)$.

2 types of events can promote those possible interactions between continuous and discrete variables in a continuous-discrete system:

a. *Time events*: events that occurs in specific instants of time.
b. *State events*: events that occur when a determined state condition is satisfied.

As can be noticed, the Discrete Event Simulation process for Fluid and Stochastic Petri nets requires an approach that considers both their time events and their state events. The time

events are related to the event-oriented method of temporal representation, and its execution dynamics makes use of simulation queues, where events are scheduled for future executions. On the other hand, the state events are related to the interval-oriented method of temporal representation, in which the time events are monitored during the simulation, without being queued (Computer Science Department, 2000).

A FSPN can be solved using DES since its execution rules be respected. For example, given a marking $(\mu,\chi)\in\hat{S}$, the content of fluid places $\chi_{(t)}(q_i)$, $q_i\in P_c$, can change while $t\in H((\mu, \chi)_{(t)})$, $t\in T$. If $t\in T_T$ and $(t,q_i) \subseteq f$, the value of $\chi_{(t)}(q_i)$ changes during the interval $[t, t+\tau)$ in which $t\in H((\mu, \chi)_{(t+\tau)})$ through the system of differential equations:

$$\forall q \in P_C, \frac{d\chi_q(\tau)}{dt} = \sum_{t\in\Lambda}\left(f_{t,q}(\mu,\chi)_{t=k} - f_{q,t}(\mu,\chi)_{t=k}\right) \quad \text{, where } \Lambda = H((\mu,\chi)_{(t)}). \tag{17}$$

The initial condition $(\mu, \chi)_{(t)}$ is unchanged during the interval $[t, t+\tau)$, and the system perceives that the continuous part of the state has changed only at the end of this time range, where $t\notin H((\mu,\chi))$. Due to this fact, the events of state, especially those dependent on $\chi(q)$ are related to the interval-oriented method of temporal representation. By moving up the time in small steps, it is possible to maintain a significantly high update rate to $\chi(q)$ values (inversely proportional to the time step), reducing the latency between the occurrence of a "real condition" of state change and its detection by the process of state monitoring of the simulation, or even the loss of the desired conditions.

Therefore, during the simulation execution, the resolution process of the systems differential equations requires to know the evolution of $H((\mu,\chi)_{(t)})$ (transitions enabled at time $t$) in function of time. Thus, $H((\mu,\chi)_{(t)})$ can have the following interdependencies to the model states (Ciardo, Nicol & Trivedi, 1999): $H((\mu,\chi)_{(t)})$ can be *independent* of $\chi(q)$ and $H((\mu,\chi)_{(t)})$ can be *dependent* on $\chi(q)$ values, $q\in P_c$.

The simplest case to handle in a DES of FSPN is when $H((\mu,\chi)_{(t)})$ is independent of $\chi(q)$, where the cardinality of the arcs and conditions of guards do not depend on $\chi(q)$. Thus, the discrete part of the model evolves through the execution of discrete events, and the continuous part can evolve independently, updating the values of $\chi(q)$ only in the discrete instants $e_n$ and based on the time difference between the current event and the previous event ($\tau$). This is equivalent to using only the event-oriented method of temporal representation on the system simulation. The simulation process becomes complex in case $H((\mu,\chi)_{(t)})$ *depends on* $\chi(q)$, the arcs cardinality and the conditions of guards of which are in function of $\chi(q)$. In this case, it is mandatory to know the future instant of time $\tau e$ at which a change in $H((\mu,\chi)_{(t+\tau e)})$ occurs. At first, obtaining $\tau e$ demands to know $\chi(q)$ during $[t, t + \Delta)$ by solving the differential equations of the model, in which $\Delta$ is a future time step used to obtain the behavior of $H((\mu,\chi))$ in that period. If there is no change in $H((\mu,\chi)_{(t+\Delta)})$ during $[t, t + \Delta)$, the values of $\chi(q)$ are updated to $t + \Delta$, and the simulation continues with the search. This is equivalent to using the interval-oriented method of temporal representation in the system simulation.

### 2.5. Estimation of the Assessment Measures by Simulation

As mentioned earlier, it is possible to obtain multiple measures of interest in nets using reward functions; their formalization is done by means of the Reward Stochastic nets (SRN).

A reward structure is characterized by a *reward rate vector* "rr" and an *impulsive reward vector* "ri", which produces 3 different reward variables, $\mathbf{R_{inst}(t)}$, $\mathbf{R_{acum}(t)}$ e $\mathbf{R_{aver}(t)}$, where it is possible to obtain their expectations **(E[])** used for analysis.

Using analytical methods, it is possible to calculate the expectations $\mathbf{E[R_{inst}(t)]}$, $\mathbf{E[R_{acum}(t)]}$ and $\mathbf{E[R_{aver}(t)]}$ by means of their literal expressions, but this could be not feasible in some cases and, therefore, the simulation technique is required. However, the use of the simulation technique can only infer (or estimate) the values of these statistical parameters. This limitation to obtain the desired measures of interest do not invalidate the assessment framework proposed here, but the framework must incorporate concepts of statistical inference to the assessment process as a means to calculate the desired measures with significant confidence, as illustrated in Fig. 2.

Therefore, it is necessary to have good statistical estimators to calculate values of Equations (5) and (6) by Discrete Event Simulation (DES). $\bar{X} = 1/N \cdot \sum_{i=1}^{N} x_i$ and $s_X^2 = \frac{1}{N-1} \cdot \sum_{i=1}^{N} (x_i - \bar{X})^2)$ are good estimators for expectation and variance of statistical variable "X" (E[X] and VAR[X], respectively), where $x_i$ is the i-upla sample obtained from the Discrete Event Simulation process and "N" is the size of the sample. Confidence Interval (CI) for statistical variable "X" at confidence level "γ", γ∈[0,1], is given by $[\bar{X} - t_{N-1,\frac{\gamma}{2}} \cdot s_{\bar{X}} / \sqrt{N} \; ; \; \bar{X} + t_{N-1,\frac{\gamma}{2}} \cdot s_{\bar{X}} / \sqrt{N}]$, where $t_{N-1,\frac{\gamma}{2}}$ is a t-Student distribution (to a large sample size "N", respecting the well-known Central Limit Theorem).

Therefore, the expectation of variables $\mathbf{R_{inst}(t)}$, $\mathbf{R_{acum}(t)}$ e $\mathbf{R_{aver}(t)}$ could be estimated by means of estimators $\bar{R}_{inst}(t)$, $\bar{R}_{acum}(t)$ e $\bar{R}_{aver}(t)$, where:

$$\bar{R}_{inst}(t) = 1/N \cdot \sum_{i=1}^{N} R_{inst}(t)_i \text{ , estimator to } \mathbf{E[R_{inst}(t)]}. \quad (18)$$

$$(s_{\bar{R}_{inst}(t)})^2 = \frac{1}{N-1} \cdot \sum_{i=1}^{N} (R_{inst}(t)_i - \bar{R}_{inst}(t))^2 \text{ , estimator to } \mathbf{VAR[R_{inst}(t)]}. \quad (19)$$

$$\mathbf{CI(R_{inst}(t), \gamma)} = [\bar{R}_{inst}(t) - t_{N-1,\gamma/2} \cdot s_{\bar{R}_{inst}(t)} / \sqrt{N} \; ; \; \bar{R}_{inst}(t) + t_{N-1,\gamma/2} \cdot s_{\bar{R}_{inst}(t)} / \sqrt{N}] \quad (20)$$

Estimators $\bar{R}_{acum}(t)$ and $\bar{R}_{aver}(t)$ follow the same structure of Equations (18), (19) and (20).

$R_{inst}(t)_i$ is the i-th sample obtained from variable $\mathbf{R_{inst}(t)}$ to constitute the estimator of parameter $\mathbf{E[R_{inst}(t)]}$. Please note that $\mathbf{R_{inst}(t)}$ is considered a random variable when its *impulsive reward vector* "ri" is null.

As previously mentioned, these properties are valid only if $\bar{R}_{inst}(t)$ is a random variable with normal distribution, where:

$$\bar{R}_{inst}(t) \sim N(E[\bar{R}_{inst}(t)], VAR[\bar{R}_{inst}(t)]) = N(E[R_{inst}(t)], VAR[R_{inst}(t)]/N) \quad (21)$$

If not, the Central Limit Theorem (CLT) can be applied, which involves using a sample N as large as enough:

$$\lim_{n \to \infty} \left( \frac{\bar{R}_{inst}(t) - E[R_{inst}(t)]}{\sqrt{\frac{VAR[R_{inst}(t)]}{n}}} \right) = Z = N(0,1) \quad (22)$$

### 2.6. Runs versus Simulation length

A system assessment process by simulation is related to the trade-off between the *simulation length* ($T_{length}$) and the number of *runs* (*Runs*) to be used, because, besides the total processing time (time of computational resources required) is given by the product $T_{length}$ **x** *Runs*, the $T_{length}$ and *Runs* characteristics influence the values of the estimates obtained. For example, each of the N samples $R_{inst}(t)_i$ used to estimate $\mathbf{E[R_{inst}(t)]}$ by means of estimator $\overline{R}_{inst}(t)$ are obtained by means of N independent *Runs* of length $t = T_{length}$ each. The process where samples are obtained through the execution of N simulation runs of length $T_{length}$, is named Replication. In this case, the following considerations can be made regarding the possible configurations of the simulation used to obtain an estimate of $\mathbf{E[R_{inst}(t)]}$:

- Using a large simulation length ($T_{length}$) and a small number of Runs, the influence of initial condition ($(\mu,\chi)_{(0)}$) over the estimative is eliminated. This is useful in the stationary (steady-state) analysis of systems. However, the small number of samples (Runs = N) undertakes the calculation of the estimation confidence interval (CI), as shown before.
- The CI of the estimate is improved, without burdening the total simulation time, increasing the number of Runs and reducing the length of the simulation ($T_{length}$). However, this fact makes the initial condition ($(\mu,\chi)_{(0)}$) have strong influence over the estimative, and it is useful in the transient analysis of the systems.

The process of estimation of parameters is only useful when it is possible to get an estimate with a satisfactory confidence interval, which requires a large number of samples (Runs). However, as it could be seen, a large number of Runs increases the processing time of the simulation. Since a stationary analysis requires a large simulation length ($T_{length}$), performing stationary (steady-state) analysis on systems can be computationally unfeasible. Hence, when replication is applied in stationary analysis, in which it is not possible to increase $T_{length}$ or *Runs* indefinitely, it is recommended to initiate the model ($(\mu,\chi)_{(0)}$) in a state closer to the stationary state so as to minimize the affectation of the initial condition over the estimatives obtained (Computer Science Department, 2000).

## 3. Case study: assessing risks in critical systems using FSPN

This chapter proposed an assessment framework that could deal with the characteristics found in the current paradigm of safety critical systems. After presenting the fundamental concepts of the framework proposed, this item presents, as a case study, its implementation in a real world safety critical system that has the characteristics of the paradigm presented: the Air Traffic Control system (ATC) under the CNS/ATM concept (Communication, Navigation, Surveillance / Air Traffic Management), especially using ADS-B (Automatic Dependent Surveillance by Broadcast) and GNSS (Global Navigation Satellite System) elements such as navigation and surveillance, respectively. This case study presents the main steps for the implementation of the framework, as the modeling of the system by FSPN and the definition of metrics of interest by reward nets, illustrating and validating the proposed framework. The details of this case study, as well as the definition of CNS/ATM and other system details were previously presented in detail in Vismari & Camargo Junior (2008).

As defined in ICAO (2007), CNS/ATM is "… employing digital technologies, including satellite systems together with various levels of automation, applied in support of a

seamless global air traffic management system". Applying this new paradigm will transform the current air traffic system, based on voice communications (Radio) and independent surveillance (Radar), into a large-scale distributed digital network. The CNS/ATM provides significant technological advances to communication, navigation and surveillance air traffic system elements to support a seamless global air traffic management system. Its key-elements are based on a global navigation satellite system (GNSS) – that will support navigation functionalities to aircrafts as well as surveillance information to air traffic control – and, mainly, based on an Aeronautical Telecommunication Network (ATN) – that will support efficient and safe digital communications among all the system users. One of the ATN air-ground applications is the Automatic Dependent Surveillance by Broadcast (ADS-B), which provides "ADS reports"( aircraft containing aircraft position and other information reports required for aircraft flight management ) to networks end users (e.g. controllers, pilots) (ICAO, 1999). ADS-B is to be applied as a surveillance system to air traffic control (ATC) service in areas where installing Radar equipments is unfeasible (e.g. ocean), and, besides that, ADS-B will complement surveillance capabilities is areas where Radar service is available.

The ADS-B is a fusion of communication (ATN) and navigation (GNSS) system elements (and their functionalities), being responsible for the surveillance system communication process. It is composed of a transmission element (responsible for generating and sending ADS reports), a data link (communication path) and a reception element (responsible for receiving, processing and displaying surveillance information to the end application). The data contained in the ADS-B report come from the navigation system (that will be at most represented by the GNSS in the CNS/ATM paradigm) obtained from the onboard aircraft navigation equipment.

Since ADS-B has caused such radical changes in the surveillance concept, it is necessary to verify if ADS-B could be used in air traffic control systems (ATC) – the most critical level of the air traffic management system. The ATC main responsibility is to apply and maintain the safe separation between aircrafts within its delegated space sector. Therefore, follow the application of the proposed framework to assess the ADS-B based ATC system.

### 3.1 Modelling the ATC system with FSPN

In this case study, the functional elements considered (modelled) to assess risks in Air Traffic Control systems (ATC) were: *Airspace* (route configuration, flight plans and so on), *Aircrafts* (flight dynamic), *Navigation* (including *Pilot* characteristics), *Communication*, *Surveillance* and *Air Traffic Controller* (ATCo). Navigation was based on GNSS and surveillance was based on ADS-B. Fig. 3 shows, at a high abstraction level, the architecture of the ATC systems considered in this work (not including Airspace).

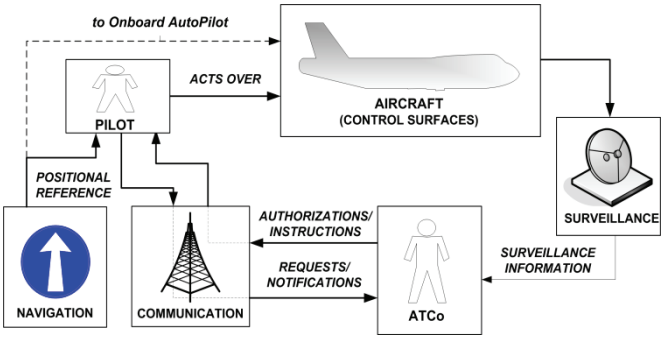



Fig. 3. The ATC system architecture

The parameters considered for each ATC system element are:

- Regarding *Airspace*, it is illustrated in Fig. 4. The route structure was considered in a Cartesian (non-spherical) plan $((X,Y)\in(R^+)^2)$ containing unidirectional routes, A and B, at the same flight level (altitude) and converging at a Θ angle ($\Theta=45^o$) to the notification point $O_z$ (with well-known coordinates $\mathbf{O_z}=(X_{NOT(z)},Y_{NOT(z)})$). Pairs of aircrafts, $A_1$ and $A_2$, were planned to conduct flights adhering to routes A and B, respectively, at the same flight level and at the same velocity (280kts). In those flights, the minima separation planned between aircrafts $A_1$ and $A_2$ ($D_{min}$) occurs when $A_1$ aircraft reaches the notification point $O_z$ ($A_2$ following $A_1$).

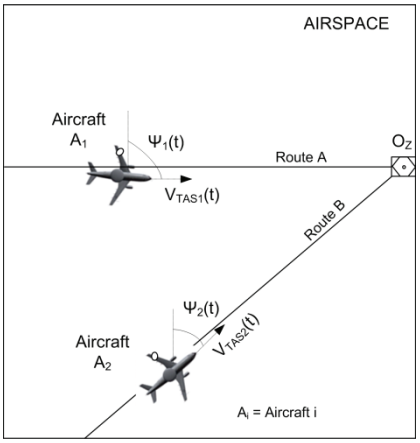



Fig. 4. The Airspace element

- Regarding *Aircraft*, three main flight parameters were considered for each one (i=1,2): position ($P_i(t)$), Velocity ($V_{TAS,i}(t)$) and Heading ($\psi_i(t)$). Then, the state of each aircraft at time t is defined as $A_i(t)=\{P_i(t),V_{TAS,i}(t),\psi_i(t)\}$. The aircrafts were responsible for following their own flight plans, controlling their $\psi_i(t)$ using onboard navigation systems (*Autopilot* and *Flight Management Computer* (FMC)) and obeying the ATCo instructions (reconfiguring *autopilot* with instructed resolution heading ($\Psi_{ATCoi}$), in case of resolution manoeuvres.

- Regarding *Air Traffic Controller* (ATCo), its *availability to detect conflicts* ($A_{ATCo}$) and the time spent to make a *decision* ($T_{decision}$), as well as the *resolution communication* ($T_{comm}$) to aircrafts involved in the conflict, were taken into consideration. A conflict resolution process (*detection-resolution-communication*) begins after ATCo observes, at least, "$N_{ConfATCo}$" successive aircraft target pairs on the surveillance screen with separation lower than $D_{ATCo_min}$ (minima separation standard). A conflict resolution process is concluded when heading the resolution communication to each aircraft pilot ($\Psi_{ATCo1}$ and $\Psi_{ATCo2}$). All times involved were represented by statistical variables with Normal distribution ($N(\mu,\sigma^2)$).
- Regarding *Pilots*, the time spent by pilots to receive and acknowledge the resolution heading ($\Psi_{ATCoi}$) communicated by the ATCo and to re-configure its value at the aircraft autopilot were considered. This time was represented by the statistical variable $T_{comm}$ at the ATCo model.
- Regarding *Navigation*, the aircraft position was obtained by its own GNNS equipment onboard. The considered parameter was the *GNSS accuracy* ($\varepsilon_{GNSS()}$~N(0.0;44.1m) per dimension.
- Regarding *Surveillance*, the aircraft positions applied to ATCo was obtained by the onboard GNSS receiver (of navigation system) and transmitted/received by the ADS-B system. The considered statistical parameters were *scan rate* ($\tau_{scan,i}$~14s@95%) and *screen presentation latency* ($\tau_{lat,i}$~1.2s@95%), both with exponential distributions.

The model in FSPN of each ATC system element is:

***Aircraft Model:*** Fig. 5 illustrates the FSPN model representing aircraft parameters $P_i(t)$ and $\psi_i(t)$ (one model to each aircraft). $V_{TAS,i}(t)$ is constant ($V_{TAS,i}$). Marking in place $K\psi$ ($\mu(K\psi)$) represents the heading in a $\psi_i=\alpha_i+(\mu(K\psi)-M).\Delta\psi$ form, where M is the number of heading steps ($\Delta\psi$). Firing time distribution of transitions $\tau_{\psi_I}$ and $\tau_{\psi_D}$ are given by $\tau_\psi=(\Delta\psi.||V_{TAS}||)/(g.\tan(\phi))$. Their firing events are enabled by guards $g(\tau_{\psi_I})$ and $g(\tau_{\psi_D})$, which are controlled by the aircraft navigation system. The aircraft position is given by continuous variables $P_i(t)=(Xi(t),Yi(t))$. The position dynamics were implemented by fluid arcs fxi, gxi, fyi and gyi, and their cardinalities were given by $B_{Xi}(\mu)=V_{TAS,i}.\text{sen}(\alpha_i+(\mu(K\psi_i)-M_i).\Delta\psi_i)$ and $B_{Yi}(\mu)=V_{TAS,i}.\cos(\alpha_i+(\mu(K\psi_i)-M_i).\Delta\psi_i)$.

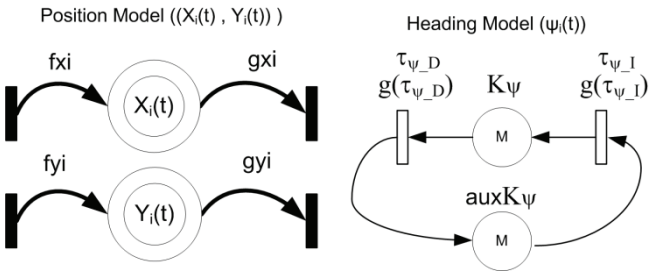



Fig. 5. Aircraft Model in FSPN

***ATCo Model:*** Fig. 6 illustrates the FSPN model representing the Air Traffic Controller (ATCo) parameters. Immediate transition "t_Conflict_occured" is enabled to fire by guard g(t_Conflict_occured) when $||P_{SUR,1}(t)-P_{SUR,2}(t)||<D_{ATCo_min}$ (guard dependent of Surveillance model). The guard g(t_NRes_conflict) enables "t_NRes_conflict" (transition

with higher priority than t_detected) to fire when NConfATCo successive conflicts does not occur.

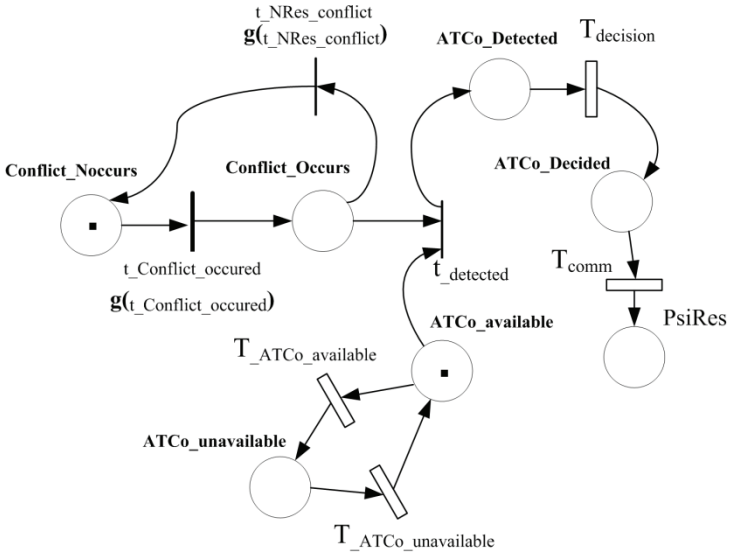



Fig. 6. ATCo Model in FSPN

***Airspace Model:*** Airspace was modelled by variables used as input to other model elements. In this way, the airspace model was represented by $\{X_{NOT(z)},Y_{NOT(z)},D_{min},\Theta,V_{TAS,i}\}_{[Airspace]}$.
***Navigation Model:*** the navigation system was modelled with 3 functionalities: i. *Obtaining aircraft position* (represented by GNSS navigation position ($P_{NAV}(t)$)); ii. *Managing flight plan* (represented by FMC, indicating to the autopilot the desired flight parameters ($(V_{TAS},\psi)_{desired}$) based on the flight plan parameters and on the $P_i(t)$); iii. *Aircraft monitoring and control* (represented by the autopilot, which receives $\psi_{desired}$ from the FMC and controls $\psi_i(t)$ according to the aircraft roll angle ($\phi$)). Their models, presented in Fig. 7, represent 3 states in dynamic aircrafts heading: "Maintaining Heading", "Decreasing Heading" and "Increasing Heading". Heading correction algorithm applies over the aircraft a roll angle value ($\phi$) based on $||\psi_i(t)-\psi_{desired}||$. Heading correction behaviour is represented by logical conditions at transition guards in navigation model, which defines its duration, when the number of marking in navigation places is not zero ($\mu(navigation\ places)\neq 0$) and, consequently, it defines the duration to be enabled by guards $g(\tau_{\psi_I})$ and $g(\tau_{\psi_D})$ for each aircraft heading models. Algorithms modelling this navigation functionality were implemented at $g(T_{_Inc_Head_i})$ and $g(T_{_Dec_Head_i})$, both calculated only when $\mu(Mant_Head_i)>0$. GNSS algorithm obtains $P_{NAV}(t_n)=(X(t)+\varepsilon x_{GNSS},Y(t)+\varepsilon y_{GNSS})=(X_{NAV}(t),Y_{NAV}(t))$, where $\varepsilon_{GNSS}$ is the position accuracy of the GNSS system.

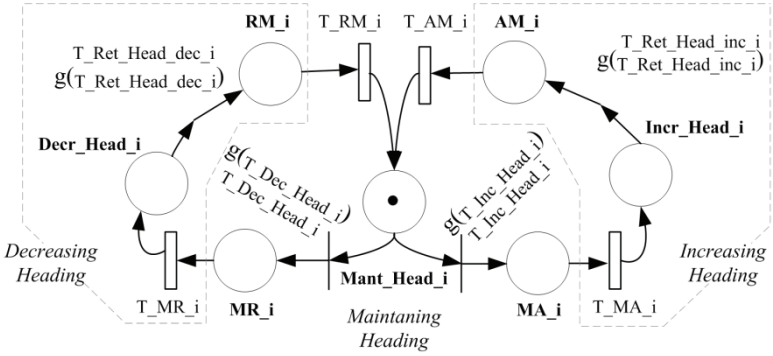



Fig. 7. Navigation in FSPN

***Surveillance Model*:** the FSPN model, illustrated in Fig. 8, represents the surveillance parameters mentioned before. $T_{SCAN}$ and $t_{sampling}$ firing events ($\uparrow T_{SCAN}$,$\uparrow t_{sampling}$) represent the process of sampling targets, where $P_{SUR,i}(\uparrow T_{SCAN},\uparrow t_{sampling})=(X_{SUR,i},Y_{SUR,1})$, i=1,2. $P_{SUR,i}(.)$ are presented to ATCo when $\uparrow t_{shows_ATCo}$, $T_{lat}$ times occur after $\uparrow T_{SCAN}$ and $\uparrow t_{sampling}$ events. The *firing time distributions* are stochastic, and there is one model embedded for each aircraft. The *source of information* ($P_{SUR,i}(t_n)=P_{NAV,i}(t_n)$) provides the same position data obtained by the navigation system).

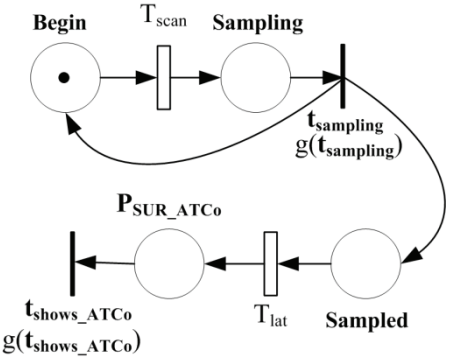



Fig. 8. Surveillance Model in FSPN

### 3.2 Defining metrics of interest by Reward Nets

In this risk assessment study, it was evaluated the ATC system ability to detect and solve the events of loss of minima separation between aircrafts that could affect air traffic safety (defined as "AIRPROX") based on its own elements. Toward it, two environment processes were observed: the real process (the real aircrafts states) and the monitored process (aircraft states observed by surveillance elements). The AIRPROX events were evaluated over the real process, verifying in which situations those events could occur due to the inefficiency of ATC elements and functionalities. In the original study (Vismari & Camargo Junior, 2008), a set of 6 metrics concerning AIRPROX were adopted. Some of them – related to the reward net concept – are presented next:

$T_{LMIN,i}$: exposure time to real separation losses lower than a value $L_{MIN}$.

$$T_{\text{Lmin_i}} = 1_{\{\|P_1(t) - P_2(t)\| < (\text{Lmin} + \text{"i"} \cdot 0{,}5\text{NM})\}} \text{, where i = 0, 1, 2 ... .} \quad (23)$$

$T_{PS_INT}$: exposure time to real separation losses lower than $D_{ATCo_min}$ per ATCo intervention.

$$T_{PS_INT} = \frac{1_{\{\|P_1(t) - P_2(t)\| < \text{Dmin_ATCo}\}}}{\#(\uparrow \text{t_detected})} \quad (24)$$

$N_{PS_Interv}$: number of conflicts presented in the surveillance screen per ATCo intervention.

$$N_{PS_Interv} = \frac{\#(\uparrow \text{"t_Conflict_occured"})}{\#(\uparrow \text{t_detected})} \quad (25)$$

An estimation to each adopted metrics was obtained by Discrete Event Simulation considering both the *functional conditions*, using sensitivity analysis at $D_{ATCo_min} \in [4.4; 5.8]$NM and the *degraded conditions of GNSS position accuracy* ($\varepsilon_{GNSS()}$) as well as the *ADS-B scan rate* ($\tau_{scan,i}$). In the simulation process, and for each parameter configuration, 10,000 aircraft approach scenarios, with 3105 seconds length each, were run. The results obtained in this case study, as well as the risk analysis performed on them, are illustrated in Vismari & Camargo Junior (2008).

## 4. Conclusions

This chapter demonstrated that the Fluid Stochastic Petri Nets (FSPN) is a good graphic-mathematical formalism for modeling and assessing the current paradigm of safety critical systems. In addition to its ability to formally represent systems both by its mathematical expressions and by its graphical formalism, the FSPN allows the modeling of complex hybrid systems - the continuous and discrete elements of which have high interdependence - and with stochastic temporal characteristics. Furthermore, the existence of the Reward Structures in the FSPN formalism allows assessing complex hybrid systems based on a variety of metrics, defined during the modeling process, allowing its use in various fields of application. The use of Discrete Event Simulation enables the analyst modeling transitions on FSPN with the widest range of probability distributions, not limited to the Markovian paradigm.

The problem found in the assessment framework proposed in this chapter, in particular in the solving method using discrete event simulation, is related to the conflict between "cost in processing time" and "confidence intervals of results". For example, in the case study, we modeled the system using 40 discrete places, approximately 5 fluid places, 80 discrete arcs, 10 fluid arcs, 50 discrete transitions and a huge number of guard functions. To obtain each metric in a specific system configuration, approximately 2.8 hours of computer processing were necessary in a 2.4GHz Intel Dual Xeon with 2.5 GB RAM, using a time interval of 0.5 seconds for the discretization of the continuous net, and using the SPNP - Stochastic Petri Net Package (Trivedi, 1999) as the FSPN computational tool. Even with the high processing cost , the use of the FSPN formalism is highly recommended for the proposed Assessment

Framework, especially when used togheter with Reward Nets, Discrete Event Simulation and Statistical Inference. Moreover, this high processing cost can motivate new researches focused on improving the performance of the simulation methods.
At last, the Assessment Framework proposed in this chapter was intended to be applied in safety assessment works. However, it could be applied to other kinds of systems to evaluate characteristics such as cost-benefit analysis, availability, reliability and others, depending on the the metrics adopted. To validate these new applications, however, it is necessary to continue this research study.

## 5. Acknowledgments.

The authors would like to thank Dr. Kishor Trivedi of Duke University for kindly granting us and providing all the support for his computational tool "Stochastic Petri Nets Package" (SPNP), and the Advanced Scientific Computing Laboratory of the University of São Paulo (LCCA/USP) for providing the computing infrastructure applied in the simulation part of this work.

## 6. References

Ahlström, K. & Torim, J. (2002). Future Architecture of Flight Control Systems. *IEEE AESS Systems Magazine,* Vol.17, Issue 12, (December, 2002) 21-27, ISSN 0885-8985.

Asarin, E., Maler, O. & Pnueli, A. (1995). Reachability Analysis of Dynamical Systems Having Piecewise-Constant Derivatives. *Theorical Computer Science,* Vol.138, Issue 1, (February, 1995) 33-35, ISSN 0304-3975.

Bobbio, A. (1988). System Modelling with Petri Nets, *Proceedings of the Ispra Course 1988*, pp. 103-143, ISBN 0-7923-0837-9, Madrid-Spain, September 1988, Kluwer, Spain.

Ciardo, G., Nicol, D.M. & Trivedi, K.S. (1999). Discrete-Event Simulation of Fluid Stochastic Petri Nets. *IEEE Transactions on Software Engineering,* Vol. 25, n. 2, (March/April, 1999) 207-217, ISSN 0098-5589.

Computer Science Department (2000). On the Simulation of Stochastic Petri Nets. *College of Willian and Mary. England. Available at* http://citeseer.ist.psu.edu/307709.htm l

German, R. (2000). *Performance Analysis of Communication Systems: Modelling with Non-Markovian Stochastic Petri Nets,* John Wiley & Sons, ISBN 0471492582, England

Gordon, G. (1978). *System Simulation,* Prentice-Hall, USA

Gribaudo, M. et al (2002). Model-Checking Based on Fluid Petri Nets for the Temperature Control System of the ICARO Co-generative Plant, *Proceedings of Safecomp 2002*, pp. 273-283, ISBN 3-540-44157-3, Catanya - Italy, September 2002, Springer, Heidelberg.

Gribaudo, M. et al (2003). Fluid Petri Nets and Hybrid model-checking: a comparative case study. *Reliability Engineering and System Safety,* Vol.81, n. 3, (September, 2003) 239-257, ISSN 0951-8320.

Hammett, R. (2003). Flight-Critical Distributed Systems: Design Considerations. *IEEE AESS Systems Magazine,* Vol.18, Issue 6, (June, 2003) 30-36, ISSN 0885-8985.

Heck, B.S. et al (2003). Software Technology for Implementing Reusable, Distributed Control Systems. *IEEE Control Systems Magazine,* Vol.23, Issue 1, (January, 2003) 21-35, ISSN 0272-1708.

Horton, G. et al (1998). Fluid Stochastic Petri Nets: Theory, applications and solution techiniques. *European Jornal of Operational Research,* Vol.105, Issue 1 (February, 1998) 184-201, ISSN 0377-2217.

ICAO (1999). *Manual of Technical Provisions for the Aeronautical Telecommunication Network (Doc.9705/AN956),* International Civil Aviation Organization, ISBN 92-9194-003-8, Montreal.

ICAO (2007). *Global Air Navigation Plan for CNS/ATM Systems (Doc.9750),* International Civil Aviation Organization, ISBN 9789291949304, Montreal.

Knight, J.C. (2002a). Safety Critical Systems: Challenges and Directions, *Proceedings of the 24rd International Conference on Software Engineering (ICSE 2002)*, pp. 547-550, Orlando - USA, May 2002, ACM.

Knight, J.C. (2002b). Software Challenges in Aviation Systems, *Proceedings of Safecomp 2002*, pp. 106-112, ISBN 3-540-44157-3, Catanya - Italy, September 2002, Springer, Heidelberg.

Kirner, T.G. (1997). Quality Requirements for Real-Time Safety-Critical Systems. *Control Engineering Practice,* Vol.5, n. 7, (July, 1997) 965-973, ISSN 0967-0661.

Labeau, P.E. et al (2000). Dynamic Reliability: towards an integrated platform for probabilistic risk assessment. *Reliability Engineering and System Safety,* Vol.68, n. 3, (June, 2000) 219-254, ISSN 0951-8320.

Leveson, N.G. & Stolzy, J.L. (1987). Safety Analysis Using Petri Nets. *IEEE Transactions on Software Engineering,* Vol.SE13, n. 3, (March, 1987) 386-397, ISSN 0098-5589.

Leveson, N.G. (1995). *Safeware: System, Safety and Computers,* Addison-Wesley Publishing Company, ISBN 0201119722, USA

Muppala, J.K., Ciardo, G. & Trivedi, K. (1994). Stochastic Reward Nets for Reliability Prediction. *Communications in Reliability, Maintainability and Serviceability: An International Journal,* Vol.1, n. 2, (March, 1994) 9-20.

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE,* Vol.77, n. 4, April, 1989) 541-579, ISSN 0018-9219.

Murray, R.M. et al (2003). Future Directions in Control in an Information-Rich World. *IEEE Control Systems Magazine,* Vol.23, Issue 2, (March, 2003) 20-33, ISSN 0272-1708.

Peterson, L.J. (1981). *Petri Net Theory and the Modelling of Systems,* Prentice-Hall, ISBN 0136619835, USA

Philippi, S. (2003). Analysis of fault tolerance and reliability in distributed real-time system architectures. *Reliability Engineering and System Safety,* Vol.82, n. 2, (November, 2003) 195-206, ISSN 0951-8320.

Ripley, B.D. (1987). *Stochastic Simulation,* Wiley & Sons, ISBN 978-0-470-00960, USA

Tipsuwan, Y. & Chow, M.Y. (2003). Control Methodologies in Networked Control Systems. *Control Engineering Practice,* Vol.11, n. 10, (October, 2003) 1099-1111, ISSN 0967-0661.

Trivedi, K. & Kulkarni, V.G. (1993). FSPN: Fluid Stochastic Petri nets, *Proceedings of the 1993 Petri Nets Conference*, Chicago-USA

Trivedi, K.S. (1999). *SPNP User's Manual Version 6.0,* Duke University, USA.

Tuffin, B., Chen, D.S. & Trivedi, K. (2001). Comparison of Hybrid Systems and Fluid Stochastic Petri Nets. *Discrete Event Dynamic Systems: Theory and Applications,* Vol.11, n. 1&2, (January/February, 2001) 77-95, ISSN 1573-7594.

Varaya, P. (1999). Design Simulation and Implementation of Hybrid Systems, *Proceedings of Aplication and Theory of Petri Nets 1999*, pp. 1-5, ISBN 3-540-66132-8, Williamsburg-USA, June 1999, Springer

Vismari, L.F. & Camargo Junior, J.B. (2008). An Absolute-Relative Risk Assessment Methodology Approach to Current Safety Critical Systems and its Application to the ADS-B based Air Traffic Control System, *Proceedings of 27th International Symposium on Reliable Distributed Systems*, pp. 95-104, ISBN 9780769534107, Napoli-Italy, October 2008, IEEE Computer Society, Los Alamitos

Wolter, K. (2000). Modelling Hybrid Systems with Fluid Stochastic Petri Nets, *Proceedings of International Conference on Automation on Mixed Process: Hybrid Dynamic Systems*, pp. 287-294, Dortmund - Germany, September 2000, Shaker Verlag, Dortmund.

# Modelling and Analysis of Traffic Light Control Systems Using Timed Coloured Petri nets

Yi-Sheng Huang[1] and Ta-Hsiang Chung [2]
[1]*Department of Electrical and Electronic, National Defense University, Taoyuan 335, Taiwan, R.O.C.*
[2] *Air Force Institute of Technology,Taiwan, R.O.C.*

## Abstract

An urban traffic network of signalized intersections can be suitably modeled as a discrete event system, in which the traffic light alternations are described by means of Timed Coloured Petri nets (TCPN). In this chapter, a module of basic traffic TCPN model with a signal timing plan for a day is successfully constructed. The traffic operations are ruled by the control logic of TCPN and an analysis method of place invariant is verified. Based on the basic TCPN model, some of complicated traffic signal models will be easily obtained. Moreover, a real-world supervisor of the urban traffic light system is implemented by the new methodology. Finally, an urban traffic light control systems with five intersections has been realized. Additionally, the performance of the supervisor of the urban traffic light system can be confirmed by the simulation results.

**Keywords:** Petri nets, traffic control, intelligent transportation systems.

## 1. Introduction

With the growing number of vehicles, the traffic congestion and transportation delay on urban arterials are increasingly worldwide. Therefore it is a practical importance to develop, verify and validate simple, yet powerful models that help in design and improve the safety and efficiency of transportation. It is a significant issue to control traffic lights in road-vehicle systems. The main reasons to the traffic signals are used to manage conflicting requirements for the use of road space – often at road junctions – by allocating right of way to different sets of mutually compatible traffic movements during distinct time intervals.
The traffic light control systems regulate, warn and guide transportation for the purpose of improving the safety and efficiency of pedestrians and vehicles. There are a lot of literatures to develop various strategies [1-10] and they are classified into two categories [11]: 1) Fixed-time strategies and 2) Traffic-response strategies. An urban traffic control in most of industrialized countries has been used fixed-time strategies to nowadays. In addition, the topic of traffic signal control can be separated into two categories [12]:

1) determine what signal-indication sequence in following order optimizes the system performance and 2) ascertaining how to implement the signal control logic. This chapter centralizes on the second category with a traffic signal timing plan that is a predetermined time.

Petri nets (PN) have been proven to be a powerful modeling tool for various kinds of discrete event systems [13-14], and its formalism provides a clear means for presenting simulation and control logic. Hence, the PN is applied in traffic control. In fact, traffic control has been accomplished by using Petri nets [15-18]. In [19] chooses Deterministic and Stochastic Petri Nets (DSPNs) as the modeling tools. The behavior of the pre-timed two-phase signal is depicted**.** The green periods, and the cycle lengths are predetermined and of fixed duration. The approach via programmable logic control (PLC) and Petri nets synthesis is proposed in [20]. Its traffic lights contain three-color lights, and each lane has a series light whose signal will be changed by the regular time intervals. And the model of traffic light is used by Petri nets. In [21] a signal timing plan is proposed by timed Petri net (TPN). The streams allowed us to proceed with the eight-phase of the signal timing plan. Recently, an urban traffic light controller using statecharts is proposed [22] which includes eight-phase, six-phase and two-phase.

As mentioned above, the models of traffic light systems only have one set of phase duration. Obviously, the timing plan of traffic light to be a leading role in urban traffic light systems. And it can determine the optimal splits and the optimal cycle time. It hints that there are many sets of phase duration varied with traffic flow in a day. As a result, the authors propose a new modeling methodology to tackle this difficult problem in this chapter. To sum up, a traffic light control system with multi-set of phase duration has been designed based on TCPN.

This chapter is arranged as follows. Section 2 briefly introduces basic definitions of TCPN that are related to this chapter. Section 3 depicts how to model the traffic lights by using TCPN. Additionally, the analysis of the TCPN models is presented in section 4. Finally, some conclusions are given in section 5.

## 2. The basic definitions of TCPN

We first introduce the new methodology which is based on a global clock. The global clock values represent the system model time where they may either be discrete or continuous. More precisely, each token has a time stamp on it. The time stamp describes the earliest model time at which the token can be moved by a binding element. Note here that one can define the time stamp unit as seconds, microseconds, or millennia and so on. It depends completely on the designer. Please note that the authors assume one time unit is equal to one second in this chapter and the definitions of TCPN are going to be presented here in a compact way and follow the original definitions of TCPN by [23-24].

Definition 1: A timed non-hierarchical CP-nets is a tuple TCPN = (CPN, R, $r_0$) such that:

1. CPN = (Σ, P, T, A, N, C, G, E, I) satisfying the requirements below:
   (1) Σ is a finite set of non empty types, called colour sets.
   (2) P is a finite set of places.
   (3) T is a finite set of transitions.
   (4) A is a finite set of arcs such that: $P \cap T = P \cap A = T \cap A = \emptyset$.
   (5) N is a node function. It is defined from A into $P \times T \cup T \times P$.

(6) C is a colour function. It is defined from P into Σ.
(7) G is a guard function. It is defined from T into expressions.
(8) E is an arc expression function. It is defined from A into expressions.
(9) I is an initialization function. It is defined from P into closed expressions that an expression is without variables.

2. R is a set of time values, also called time stamps. It is closed under + and containing 0.

3. $r_0$ is an element of R called the start time.

Definition 2: A binding of transition t is a function b defined on Var(t) which is called the set of variables of t. B(t) is denoted the set of all binding for t.

Definition 3: A binding element is a pair (t, b) where $t \in T$ and $b \in B(t)$. The set of all binding elements is denoted by BE.

Definition 4: A step Y is enabled in a marking M iff the following properties are satisfied:

$$\forall p \in P: \sum_{(t,b)\in Y} E(p,t) < b > \quad \leq M(p)$$

The expression evaluation E(p, t) <b> yields the multi-set of token colours, which are removed from p when t occurs with the binding b. For a time concept, the enabled definition is modified as follows:

A step Y is enabled in a state ($M_1$, $r_1$) at time $r_2$ iff the following properties are satisfied:

1. $\forall p \in P: \sum_{(t,b)\in Y} E(p,t) < b >_{r_2} \leq M_1(p)$

2. $r_1 \leq r_2$.

3. $r_2$ is the smallest element of R for which there exists a step satisfying above two restrictions.

Next, the definition is relation with place invariant.

Definition 5: For a non-hierarchical CP-net, a set of place weights with range $A \in \Sigma$ is a set of functions $W = \{W_p\}_{p\in P}$ such that $W_P \in [C(p)_{WS} \rightarrow A_{WS}]_L$ for all $p \in P$.

1. W is a place flow iff

$$\forall (t,b) \in BE: \sum_{p\in P} W_P(E(p,t) < b >) = \sum_{p\in P} W_P(E(p,t) < b >).$$

2. W determines a place invariant iff:

$$\forall M \in [M_0 >: \sum_{p\in P} W_P(M(p)) = \sum_{p\in P} W_P(M_0(p)) \cdot$$

Note that the invariant property in 2 is a dynamic property, while the flow property in 1 is a static property. It hints that the static property which can be checked without considering the set of all reachable markings.

## 3. Modelling of traffic light systems by using TCPN

One can see frequently the vehicle goes straight and turns right in an intersection with two-way roads. In this chapter, a traffic system with the two phases (i.e. shown in table I), called a basic traffic system, will be introduced in this section. The two phases consist of phase_NS and phase_EW. And the operations are described as follows. In phase_NS, the northbound and southbound vehicles are allowed to go straight and turn right. In phase_EW, the westbound and eastbound vehicles are allowed to go straight and turn right.

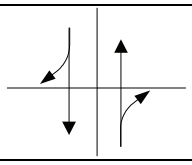
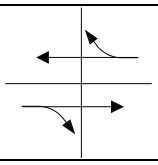
| Phase_*NS* | Phase_*EW* |
|---|---|
|  |  |

Table 1. The two phase transitions.

A schedule of the traffic signals is given in table II for this intersection. At first, the definitions of "period cycle" that represents one cycle of a period and "day cycle" that represents one cycle of a day are introduced. There are several periods in a day cycle and they represent the intervals which are continuous execution times. The duration of the phase_NS represents the time of the vehicles can go and pass the intersection in NS direction. Similarly, the duration of the phase_EW is the vehicle can pass through the intersection in EW direction. The duration of the two phases include both the duration of green and the duration of yellow. A cycle time is defined as the summation of the two durations. It is worthy to notice that there are with many repetition cycles in a period. For example, the numbers of repetition cycles are 30 in one hour if its cycle time is 120 seconds.

| Period | $t_1$ | $t_2$ | … | $t_i$ |
|---|---|---|---|---|
| Execution time | 0-$h_1$ | $h_1$-$h_2$ | … | $h_{i-1}$-$h_i$ |
| Phase_NS duration | $\tau_{11}$ | $\tau_{12}$ | … | $\tau_{1i}$ |
| Phase_EW duration | $\tau_{21}$ | $\tau_{22}$ | … | $\tau_{2i}$ |
| Cycle time | $ct_1$ | $ct_2$ | … | $ct_j$ |

Table 2. A schedule of traffic light signals.

Here, the chapter focuses on how to model the traffic lights using TCPN. It is an interesting work for the traffic light system model. It is worthy to notice that three TCPN models are constructed in Figure 1 by three type lines.

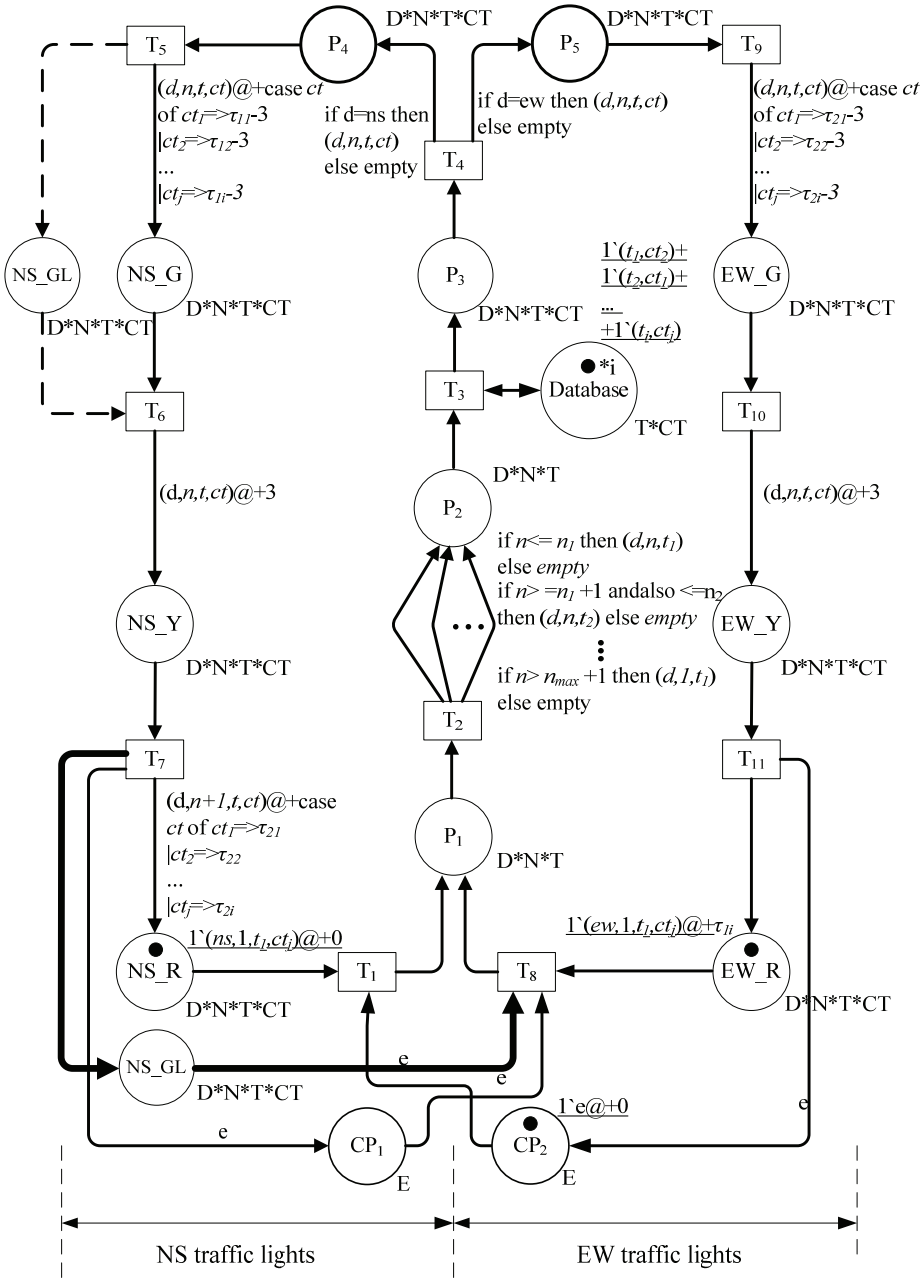



Fig. 1. An intersection TCPN model.

One is the basic traffic system model whose is modeled by normal lines. The other one is an extended model which is added with dash lines. Another one is also an extended model which is added with heavy lines. The two type extended models will be explained later. The basic traffic system model consists of two main parts: the left part which is called the NS traffic lights describes the states of both the northern and southern traffic lights; the right part which is called the EW traffic lights depicts the states of both the eastern and western traffic lights. More precisely, each part can be divided into two sub-models. One sub-model is for the signal indication and the other is for the signals' timing whose model is shared between the signal indication models. The duration of the yellow is assumed as 3 seconds and the lost time [11] are neglected in this chapter. The more detailed information is described as follows.

### 3.1. Signal indication models

Usually, the green light means the vehicles can go straight and turn right. And the vehicles do not permit turning left. In the pioneer works [17-20], this situation is hardly presented in detail. However, this case is easily modeled by the new methodology. A signal indication model can show the traffic light in red, yellow or green. The traffic lights are modeled as places EW_G, EW_Y, EW_R, NS_G, NS_Y, and NS_R (i.e. Figure 1). The places of the traffic lights are triggered by tokens in the TCPN model. Based on the rules of TCPN, the tokens are defined as 4-tuple (d, n, t, ct) with time stamp. The first element d represents the traffic lights in NS/EW. Thus it can be written as $d \in \{ns, ew\}$. The second one n (i.e. $n \in I^+$) represents the numbers of the repetition cycles. The element t stands for the period time, where $t \in I^+$. The final element ct represents the cycle time, where $ct \in I^+$. Here, $I^+$ is non-negative integer.

Places NS_G, NS_Y, NS_R, EW_G, EW_Y, EW_R, $CP_1$ and $CP_2$ belong to this sub-model. A token in place NS_G means the green lights turn on in NS traffic lights. And then place NS_Y (NS_R) with a token represents the yellow (red) lights turn on in NS traffic lights. Similarly, the places EW_G, EW_Y and EW_R with tokens means the green lights, yellow lights and red lights turn on in EW traffic lights, respectively. Places $CP_1$ and $CP_2$ play the guard roles, which guarantee only one direction is allowed to proceed for the vehicle streams. For example, when the green light turns on in NS, the red light turns on in EW. More specifically, the two places ensure the safety of the traffic. The transitions of the system model will be introduced in the following paragraph.

The transitions $T_5$ and $T_9$ stand for the green durations. Because the duration of the phase involves green duration and yellow duration (i.e. 3 seconds), the time inscription $\tau$-3 that deducts yellow duration from phase duration is a green duration. The transitions $T_6$ and $T_{10}$ represent the yellow duration (i.e. 3 seconds). The transition $T_7$ stands for the red durations in NS. And then the red duration in EW is controlled by the token of place $CP_1$. The physical means the red light of the traffic lights in EW is going on till the enabled binding element ($T_8$, e) is taken. Obviously, the red duration of traffic lights in EW is the $\tau_{1i}$ seconds. It reveals that the vehicle streams only proceed in either NS or EW.

Based on the basic sub-model, two types of extended signal indication models which involve a left turn arrow on green are obtained. The first one is added with the dashed lines in Figure 1 which describes the left turn arrow on green to be triggered with the right turn

arrow on green and go straight arrow on green at the same time. In other words, the three green lights are turned on concurrently. In this extended signal indication model, the authors add a place NS_GL to the basic traffic model. And it represents the left turn green light. The second one is added with heavy lines in Figure 1 which indicates only one of the three green lights (left turn green light, go straight green light, and right turn green light) goes on. It means that the left turn green light (NS_GL) is going on after the other two green lights (NS_G) turn off. For this purpose, the authors add a place NS_GL to the basic model. It also represents the left turn green light. It is worthy to notice that the two types of extended signal indication models in Figures 1 just only describe one part of NS traffic lights. The attached places and transitions can also be constructed the part of EW traffic lights. The extended signal indication model is useful in the multi-phase transitions.

### 3.2. Signal timing plan models

The sub-model of the signal timing plan model is the kernel of the TCPN model (i.e. Figure 1). It is the model's duty to assign the cycle times to the different periods. Especially, there are no durations and delay times in the sub-model. Indeed, it is used to estimate what is the cycle time of a period. In the sub-model, one has to count the numbers (i.e. $n_i$) of the repetition cycle. Once the number of repetition cycles meets the number $n_i$, the current period $t_i$ will enter next period $t_{i+1}$. And then a new cycle time will be given in terms of period $t_{i+1}$.

The definitions of the elements of tokens in the sub-model are the same as the signal indication model. It is worthy to notice that the component elements of token in this sub-model consists of two (i.e. (t, ct)), three (i.e. (d, n, t)), or four (i.e. (d, n, t, ct)) elements. And the signal indication model consists of four elements (i.e. (d, n, t, ct)). The more explicit expression describing as follows.

There are several arc expressions between transition $T_2$ and place $P_2$. The arc expressions determine when the current period goes to next period. The repetition cycle time in a "period cycle" is represented by $r_i$. And a values $n_i$ is determined from the formula, where $n_i = \sum_{i=1}^{n} r_i, n \in I^+$. For example, $n_2 = r_1 + r_2$. Once the number of repetition cycles is equal to $n_i$. The current period ($t_i$) will go to the next period ($t_{i+1}$). And then the current token (d, n, $t_i$) will change to (d, n, $t_{i+1}$). In addition, a maximum number of repetition cycles $n_{max}$ for the "day cycle" is required. Because the system model needs a mechanic to judge what is the day off. At the new day, the number $n_{max}$ is enforced to reset. It means the state is coming back to the initial state.

In summary, when a token is removed from $P_1$ to $P_2$, the token is with a $t_i$. Once the token is removed from $P_2$ to $P_3$, the token is now with a $ct_i$. Note that the $ct_i$ is offered by the Place Database. The function of Place Database is that the model can show both of the cycle time and the current period together. Therefore, we have to create a database of ($t_i$, $ct_j$) by tokens in Database. Meanwhile, the output token is changed and it is with four elements like as (d, n, $t_i$, $ct_i$).

## 4. Analysis of the urban traffic net model

A traffic light control system model must have correct and readable features. For example, the controller should not lock up (deadlock) due to some unexpected combination of actions, should not allow conflicting movements to have right of way simultaneously, should be able to serve all signal phases and return to some initial state. A major strength of TCPN is the availability of methods for analyzing the properties of the model. Those properties of TCPN model reveal weather the model is reliable or not.

There are three methods to analyze a TCPN model: 1) invariant method; 2) occurrence graphs method; and 3) simulation method [23]. In this section the basic traffic light control system model is analyzed by the occurrence graphs method and invariant method. In addition, a real-world urban traffic net which consists of three intersections is used to verify the model by simulation.

### 4.1. Invariant method

The basic ideal behind place invariants creates equations that are satisfied in all reachable marking [24]. In TCPN nets, the sets of removed tokens are not fully determined by the binding elements. Based on the rules of TCPN, a transition can be fired if the global time is great than or equal to the time stamp. It hints that the system acts according to the time stamps of the binding elements. In fact, the system models require only the time stamps to be small enough, instead of requiring them to have some exact time values. This means that linearity of weight functions is insufficient to guarantee that each flow determines an invariant. However, our traffic light TCPN model is predetermining time. Therefore, it is certainly to use invariants in analysis of TCPN models. The basic model (i.e. Figure 1) is used to analyze the place invariance. Based on the definition 5, we can obtain several equations from Figure 1. And the performance of the system should be verified by the equations. The detailed information is given as follows.

$$NS_G + NS_Y + NS_R + P_1 + P_2 + P_3 + P_4 = 1 \quad (1)$$

This invariant states that they can be only one token in any one of the places involved in (1). And it indicates that the firing sequence of the binding elements should be in order at NS traffic light. For example, the variations of traffic lights are red, green and yellow in turn. Note that places $P_1$, $P_2$, $P_3$, and $P_4$ are necessary control places. Similarly, the other invariant can be obtained, i.e., $EW_G+EW_Y+EW_R+P_1+P_2+P_3+P_5 =1$. This invariant asserts that, at any given time, there are only one token in the right-hand side of the display indication modules.

$$NS_G+NS_Y+NS_R+P_1+P_2+P_3+P_4 = EW_R, \text{ where } EW_R = 1 \quad (2)$$

This invariant depicts that if there is one token in place EW_R, then there must be a token in either place NS_G, NS_Y, NS_R, $P_1$, $P_2$, $P_3$ or $P_4$. It means that once a red signal goes on in the EW traffic lights, then a green, a yellow or a red signal turns on in the NS traffic lights.

In the other way, the invariant $EW_G+EW_Y+EW_R+P_1+P_2+P_3+P_5=NS_R$ asserts that if there is one token in place NS_R, then there must be a token in either place EW_G, EW_Y,

EW_R, $P_1$, $P_2$, $P_3$ or $P_5$. It shows that once a red signal is going on in the EW traffic lights, then a green, a yellow or a red turns on in the NS traffic lights.

$$NS_G+NS_Y+ EW_G+EW_Y+ P_1+P_2+P_3+P_4+P_5+CP_1+CP_2 =1 \quad (3)$$

As mentioned above, this invariant means that the direction of the vehicles movement is either in NS or EW. This invariant hints that the system model guarantees the safety of traffic.

The three equations (1), (2), and (3) show the key invariants in the TCPN model. And the three key invariants show the system model is with invariant.

### 4.2. Occurrence Graphs method

The basic idea behind occurrence graphs (OG) is to construct a graph, which is shown in Figure 2), containing a node for each reachable marking and an arc for each occurring binding element. And also it is intuitive in this approach to see that there is no possibility of deadlock in the system model.

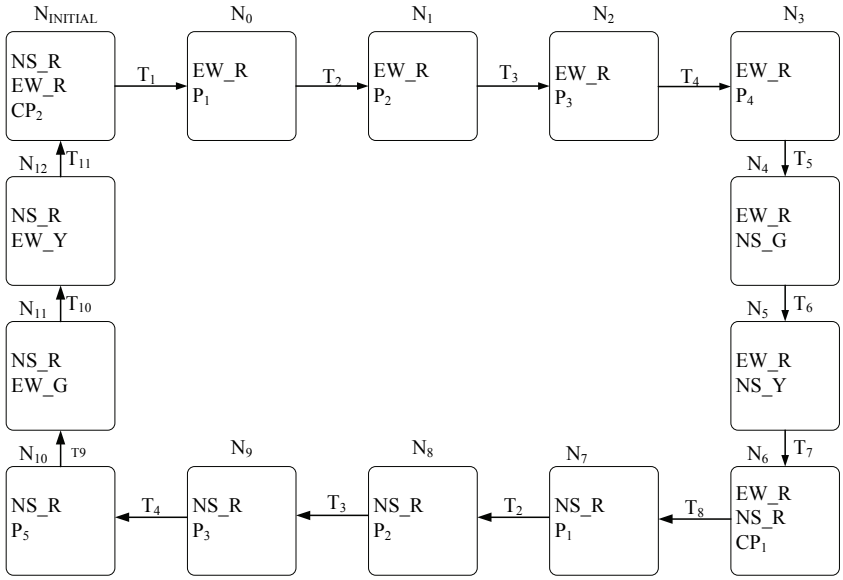



Fig. 2. The OG of the basic TCPN model.

It is worthy to notice that each node represents a marking, and the content of the marking is described by the text inscription of the node. And each arc represents the occurrence of a binding element, and the content of this binding element is described by the text attached to the arc. For example, the text EW_R and NS_G are used to describe the node $N_5$. This means that the red lights turn on in the EW traffic light. And the green lights turn on in the NS traffic lights. After a binding element $T_6$ firing, the marking will be changed to node $N_6$. It is worthy to notice that the initial marking (i.e., node $N_{INITIAL}$) shows all the red lights

turn on in a set of traffic lights. For the sake of simplicity, the place Database is not shown in the OG.

Based on this analysis, it can be concluded that: 1) there are no dead-end nodes in the OG, therefore the net is live; 2) the net is reversible because we can always find an occurring sequence that bring the system back to the initial marking.

### 4.3. The simulation results

Considering the simulation, a real-world urban traffic net is used to verify the model. The real traffic net consists of five intersections which are shown in Figure 3. Some notations are employed to represent the five intersections. For example, 1NS_G means the green light goes on in the NS traffic light which is placed at intersection $I_1$. Table 3 shows the information of the phase transitions in this real case. And also table 4 shows the signal time plan of traffic light model. The urban traffic net model will be constructed based on the basic traffic model. For convenience, the five intersections are divided into three types. One consists of intersection $I_1$, $I_4$ and $I_5$ whose models are the same as the basic model. The other one is with intersection $I_2$ whose model is obtained from the first extending model. Another one is with intersection $I_3$ whose model is also from the second extending model. The detailed information of the urban net model is described as follows.

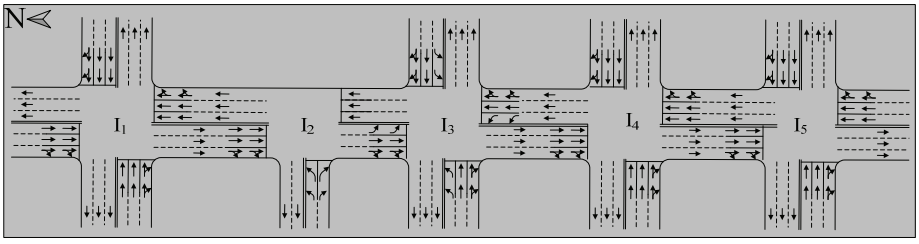



Fig. 3. A real-world urban traffic net.

| | Phase1 | Phase2 | Phase3 | Phase4 |
|---|---|---|---|---|
| I1 | | | | |
| I2 | | | | |
| I3 | | | | |
| I4 | | | | |
| I5 | | | | |

Table 3. The phase transitions of the traffic net.

| | Period | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ | $t_8$ | $t_9$ |
|---|---|---|---|---|---|---|---|---|---|---|
| | Execution time | 00:00 - 01:00 | 01:00 -05:00 | 05:00 -07:00 | 07:00 -09:00 | 09:00 -13:00 | 13:00- 16:30 | 16:30 -19:00 | 19:00 -23:00 | 23:00 -24:00 |
| I1 | Phase 1 | 60 | 46 | 60 | 110 | 110 | 110 | 110 | 95 | 60 |
| | Phase 2 | 60 | 44 | 60 | 90 | 90 | 90 | 90 | 85 | 60 |
| | Cycle time | 120 | 90 | 120 | 200 | 200 | 200 | 200 | 180 | 120 |
| I2 | Phase 1 | 80 | 60 | 80 | 140 | 140 | 140 | 140 | 120 | 80 |
| | Phase 2 | 40 | 30 | 40 | 60 | 60 | 60 | 60 | 60 | 40 |
| | Cycle time | 120 | 90 | 120 | 200 | 200 | 200 | 200 | 180 | 120 |
| I3 | Phase 1 | 55 | 30 | 55 | 85 | 85 | 85 | 85 | 80 | 55 |
| | Phase 2 | 10 | 10 | 10 | 15 | 15 | 15 | 15 | 15 | 10 |
| | Phase 3 | 45 | 40 | 45 | 85 | 85 | 85 | 85 | 70 | 45 |
| | Phase 4 | 10 | 10 | 10 | 15 | 15 | 15 | 15 | 15 | 10 |
| | Cycle time | 120 | 90 | 120 | 200 | 200 | 200 | 200 | 180 | 120 |
| I4 | Phase 1 | 60 | 45 | 60 | 100 | 100 | 100 | 100 | 90 | 60 |
| | Phase 2 | 60 | 45 | 60 | 100 | 100 | 100 | 100 | 90 | 60 |
| | Cycle time | 120 | 90 | 120 | 200 | 200 | 200 | 200 | 180 | 120 |
| I5 | Phase 1 | 90 | 60 | 90 | 160 | 160 | 160 | 160 | 140 | 90 |
| | Phase 2 | 30 | 30 | 30 | 40 | 40 | 40 | 40 | 40 | 30 |
| | Cycle time | 120 | 90 | 120 | 200 | 200 | 200 | 200 | 180 | 120 |

Table 4. The schedule of the signal timing plan.

(1) Modeling of intersection $I_1$, $I_4$, and $I_5$ (type I)

The model of the type I is the same as basic traffic system's model. Hence, the module of basic traffic system could be applied directly to the type I. Form the schedule of the signal timing plan, there are four kinds of cycle time are given. For fitting the basic models, the notations $ct_1 = 90$, $ct_2 = 120$, $ct_3 = 180$ and $ct_4 = 200$ are assigned. And then the values of $r_i$ and $n_i$ have to be determined. The intersection $I_1$ is used to illustrate how to obtain the values $r_i$ and $n_i$. Table 4 shows the execution time and the cycle time for every period of the intersection $I_1$. From the table, the numbers of the repetition (i.e. $r_i$) is easily to count. For example, the numbers of the repetition of $r_1$ is 30 for the period $t_1$ in the intersection $I_1$. As a result, all the numbers of the repetition (i.e. $r_i$) are obtained and are shown in table 5. Based on table 5, the values $n_i$ which is shown in table 6 is determined from the formula, where $n_i = \sum_{i=1}^{n} r_i, n \le 9$. For example, $n_2 = r_1 + r_2 = 30 + 160 = 190$. It is worthy to notice

that the values of $r_i$ and $n_i$ in table 5 can be applied to intersections $I_2$ and $I_3$ since the three intersections have the same cycle time in a period.

Finally, three TCPN models of type I will be derived if the values $n_i$, $\tau_{1i}$, $\tau_{2i}$, and $ct_j$ are put into the original model (i.e. Figure 1). And then the three models (i.e. $I_1$, $I_4$ and $I_5$) are given in Figure 4, 5 and 6, respectively.

| | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ | $r_7$ | $r_8$ | $r_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $r_i$ | 30 | 160 | 60 | 36 | 72 | 63 | 45 | 80 | 30 |

Table 5. The values of $r_i$..

| | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | $n_6$ | $n_7$ | $n_8$ | $n_9$ |
|---|---|---|---|---|---|---|---|---|---|
| $n_i$ | 30 | 190 | 250 | 286 | 358 | 421 | 466 | 546 | 576 |

Table 6. The values of $n_i$.

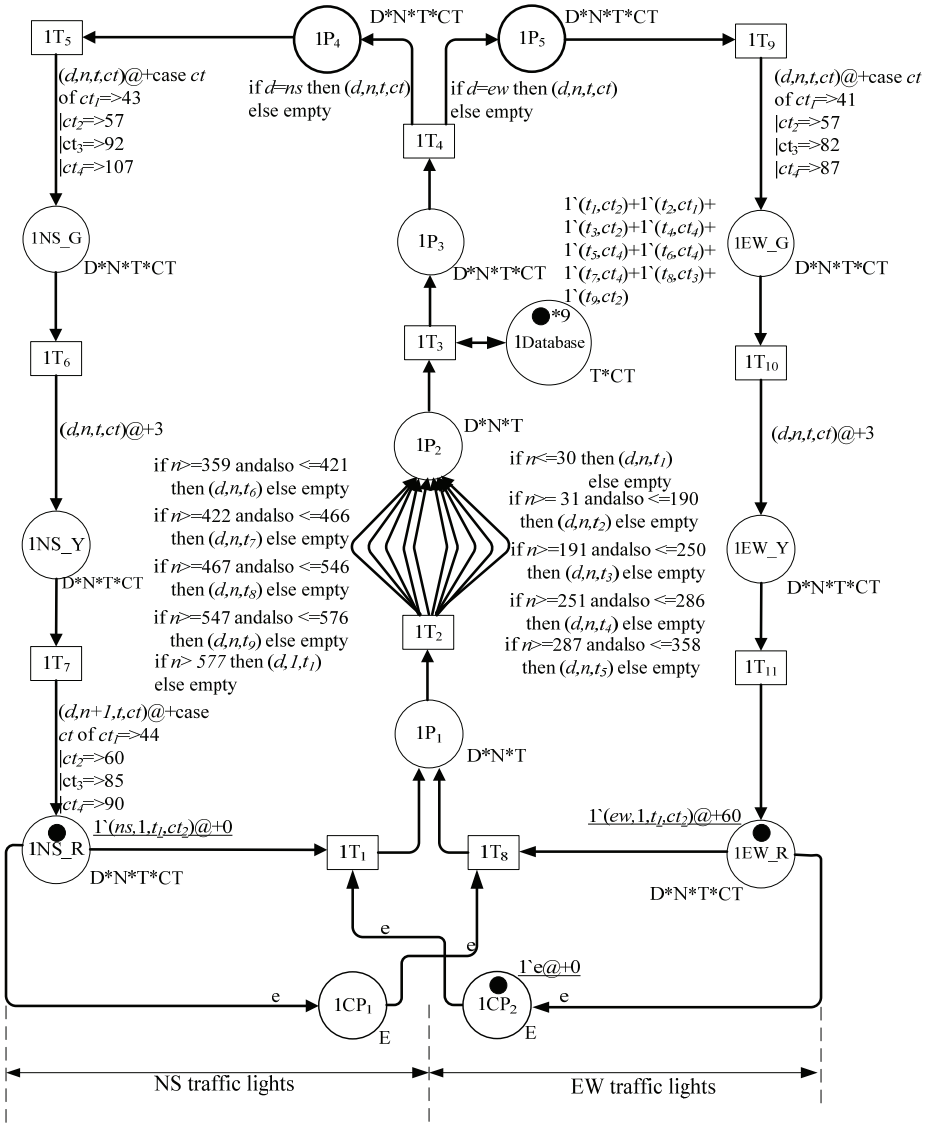



Fig. 4. The TCPN model of the intersection $I_1$.

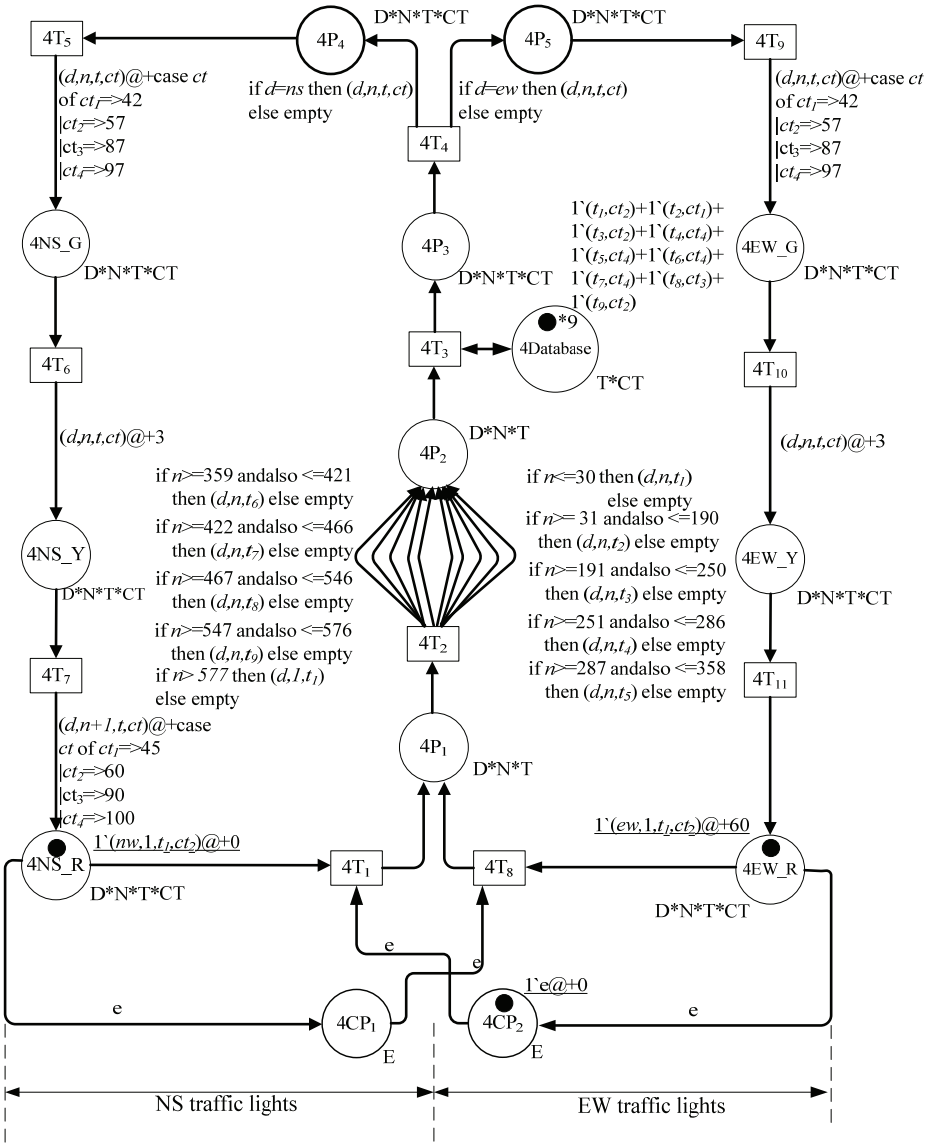



Fig. 5. The TCPN model of the intersection $I_4$.

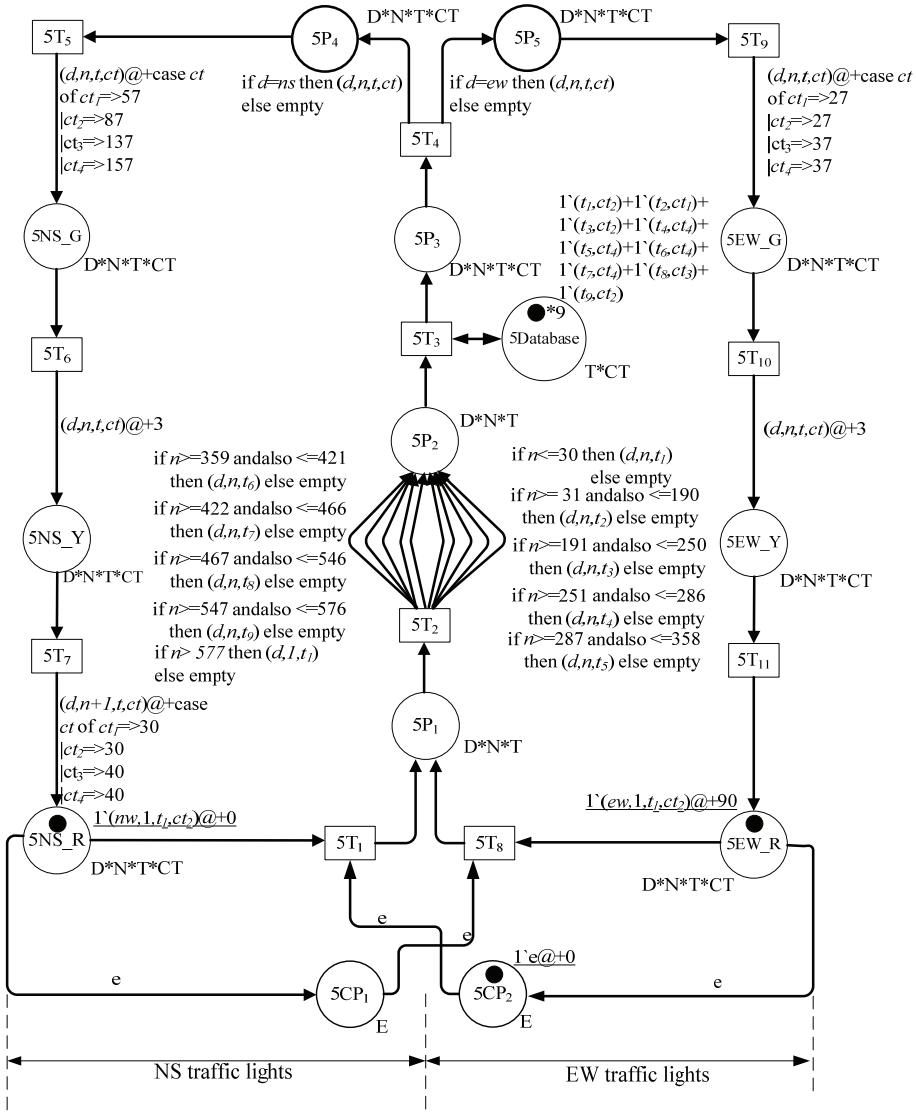



Fig. 6. The TCPN model of the intersection $I_5$.

Based on the Place-invariant method, the three models should be verified by the following equations.

Intersection $I_1$ :

$$1NS_G+1NS_Y+1NS_R+1P_1+1P_2+1P_3+1P_4 = 1 \quad (4)$$

$$1EW_G+1EW_Y+1EW_R+1P_1+1P_2+1P_3+1P_4 = 1 \quad (5)$$

$$1NS_G+1NS_Y+1NS_R+1P_1+1P_2+1P_3+1P_4 = 1EW_R, \text{ where } 1EW_R = 1 \quad (6)$$

$$1EW_G+1EW_Y+1EW_R+1P_1+1P_2+1P_3+1P_4 = 1NS_R, \text{ where } 1NS_R = 1 \quad (7)$$

$$1NS_G+1NS_Y+1EW_G+1EW_Y+1P_1+1P_2+1P_3+1P_4+1P_5+1CP_1+1CP_2 = 1 \quad (8)$$

Intersection $I_4$：

$$4NS_G+4NS_Y+4NS_R+4P_1+4P_2+4P_3+4P_4 = 1 \quad (9)$$

$$4EW_G+4EW_Y+4EW_R+4P_1+4P_2+4P_3+4P_4 = 1 \quad (10)$$

$$4NS_G+4NS_Y+4NS_R+4P_1+4P_2+4P_3+4P_4 = 4EW_R, \text{ where } 4EW_R = 1 \quad (11)$$

$$4EW_G+4EW_Y+4EW_R+4P_1+4P_2+4P_3+4P_4 = 4NS_R, \text{ where } 4NS_R = 1 \quad (12)$$

$$4NS_G+4NS_Y+4EW_G+4EW_Y+4P_1+4P_2+4P_3+4P_4+4P_5+4CP_1+4CP_2 = 1 \quad (13)$$

Intersection $I_5$：

$$5NS_G+5NS_Y+5NS_R+5P_1+5P_2+5P_3+5P_4 = 1 \quad (14)$$

$$5EW_G+5EW_Y+5EW_R+5P_1+5P_2+5P_3+5P_4 = 1 \quad (15)$$

$$5NS_G+5NS_Y+5NS_R+5P_1+5P_2+5P_3+5P_4 = 5EW_R, \text{ where } 5EW_R = 1 \quad (16)$$

$$5EW_G+5EW_Y+5EW_R+5P_1+5P_2+5P_3+5P_4 = 5NS_R, \text{ where } 5NS_R = 1 \quad (17)$$

$$5NS_G+5NS_Y+5EW_G+5EW_Y+5P_1+5P_2+5P_3+5P_4+5P_5+5CP_1+5CP_2 = 1 \quad (18)$$

(2) Modeling of intersection $I_2$ (type II)

A T-type intersection is usually presented in a real-world urban traffic system. A feature of the T-type intersection is that one way of the vehicle stream can not be allowed to go straight. For this reason, a left turn arrow on green is needed to present in the traffic light system. As a result, the function of the left turn green light can be presented when a place NS_GL is added on the original model. It is interesting that the completed TCPN model of the T-type intersection (i.e. Figure 7) can be derived after a minor revision of the first extended traffic system model.

Form the definition of the phases (i.e. table 3), a right turn movement is absent in northern traffic light (i.e. phase 1 of $I_2$). For convenience, i.e., in the extended model (i.e. Figure 7), the place NS_G is used to represent the vehicle stream can be allowed to go straight in the northbound traffic light and the other way can be allowed to go straight and turn right. In addition, the vehicle stream can not be allowed to go straight in the eastbound (i.e. phase 2 of $I_2$). This situation is different form the phase of the basic model (i.e. phase 2 of $I_1$). By the same reason, the place E_G is used to represent the vehicle stream can be allowed to turn right and turn left in the eastern traffic light. Notice that the place EW_G/ EW_GL is instead by place E_G/E_GL in the new extended TCPN model.

Finally, the TCPN model of the T-type will be derived if the values $n_i$, $\tau_{1i}$, $\tau_{2i}$, and $ct_j$ are put into the original model. And then the model is constructed and is shown in Figure 7.

Based on the Place-invariant method, the model of intersection $I_2$ should be verified by the following equations：

$$2NS_G+2NS_Y+2NS_R+2P_1+2P_2+2P_3+2P_4 = 1 \quad (19)$$

$$2W_G\ (2W_GL) +2W_Y+2W_R+2P_1+2P_2+2P_3+2P_4 = 1 \quad (20)$$

$$2NS_G+2NS_Y+2NS_R+2P_1+2P_2+2P_3+2P_4 = 2W_R, \text{ where } 2W_R = 1 \quad (21)$$

$$2W_G\ (2W_GL) +2W_Y+2W_R+2P_1+2P_2+2P_3+2P_4 = 2NS_R, \text{ where } 2NS_R = 1 \quad (22)$$

$$2NS_G+2NS_Y+2W_G\ (2W_GL) +2W_Y+2P_1+2P_2+2P_3+2P_4+2P_5+2CP_1+2CP_2=1 \quad (23)$$

(3) Modeling of intersection $I_3$ ( type III)

From the table 3, the phases of intersection $I_3$ is with four phases. Obviously, one of the phases is with a left turn movement of intersection $I_3$. The TCPN model of the intersection $I_3$ can be easily obtained due to the phase is already included in the second extended model. As a result, the TCPN model of the intersection $I_3$ can be constructed if the relations values are put into the original model. And the model is constructed and is shown in Figure 8.

Based on the Place-invariant method, the model of intersection $I_3$ should be verified by the following equations：

$$3NS_G+3NS_Y+3NS_R+3P_1+3P_2+3P_3+3P_4 = 1 \quad (24)$$

$$3EW_G+3EW_Y+3EWS_R +3P_1+3P_2+3P_3+3P_4 = 1 \quad (25)$$

$$3NS_G+3NS_Y+3NS_R+3P_1+3P_2+3P_3+3P_4 = 3EW_R, \text{ where } 3EW_R = 1 \quad (26)$$

$$3EW_G+3EW_Y+3EW_R+3P_1+3P_2+3P_3+3P_4 = 3NS_R, \text{ where } 3NS_R = 1 \quad (27)$$

$$3NS_G+3NS_Y+3NS_GL+3EW_G+3EW_Y+3EW_GL +3P_1+3P_2+3P_3+3P_4+3P_5 +3CP_1 +3CP_2 = 1 \quad (28)$$

The TCPN models are implemented and simulated by the CPN tools [25]. Aim of the simulation is to observe the relation between cycle time and execution time in each period. Based on the schedule of the signal timing plan (i.e. table 4), the global time of the supervisor is set 86400 time units (i.e. one time unit is equal to one second) for the nine periods (i.e. $t_1$, $t_2$, …, and $t_9$) in the CPN tools. In summary, the simulation results are the same as the predetermining time (i.e. table 4) and the cycle time of each period is consistent with table 4. Moreover, the traffic performance can be confirmed by the simulation results.

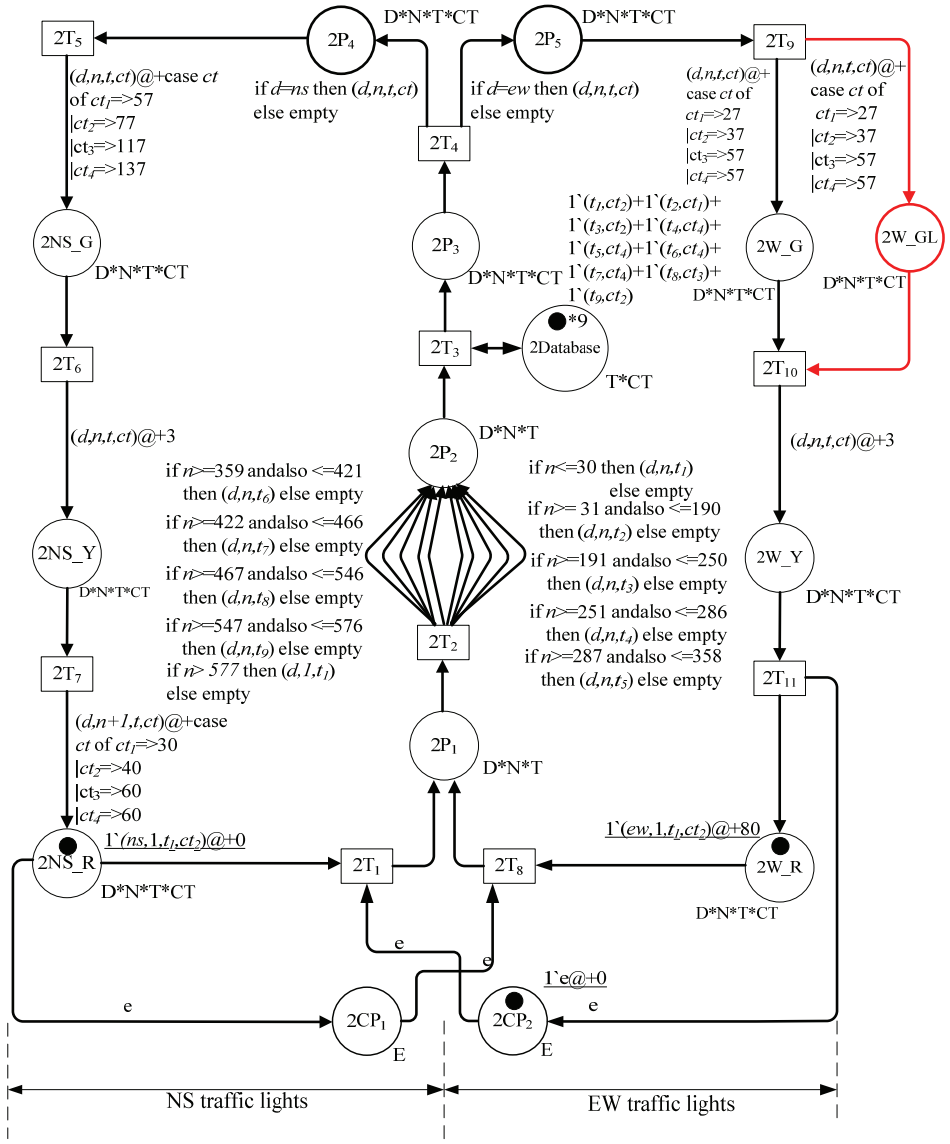



Fig. 7. The TCPN model of the intersection $I_2$.

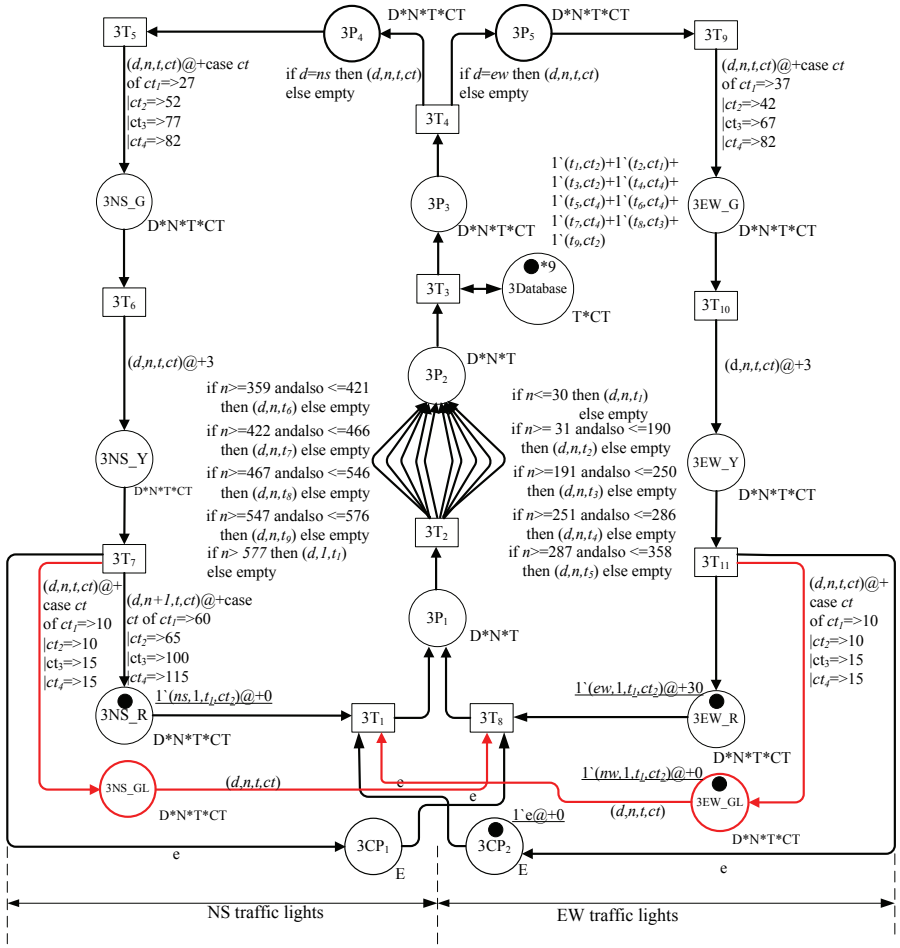



Fig. 8. The TCPN model of the intersection $I_3$.

## 6. Conclusion

This chapter presents the modeling, analysis and implementation of an urban traffic lights system using TCPN models. Especially, this chapter also proposed the module of basic traffic light system model which can assist in designing the extended models. Based on the operational flow of the traffic lights systems, the authors derive the associated TCPN model by looking into the schedule of the signal timing plan of the traffic systems. The advantage of the proposed approach is the clear presentation of the system behavior and readiness for implementation. To summarize, this chapter has the following contributions.

This chapter has demonstrated how to use TCPN to model the traffic lights of the urban network. And then the applications of TCPN to urban traffic lights have been realized.

Structural analysis of TCPN models was performed.

The traffic systems with signal timing plan for a day is successful to convert TCPN models. These examples are helpful to us to obtain a TCPN model for a complex urban traffic lights system.
The authors believe that using TCPN to model traffic light systems will become more important in this field due to the increasing demands in many features of the traffic light systems.

## 7. Acknowledgements

This work was supported by the National Science Council of Taiwan, R.O.C. under Grant NSC 96-2221-E-606-029.

## 8. References

[1]Allsop, R. E.: 'SIGSET: A computer program for calculating traffic capacity of signal-controlled road junctions', Traffic Eng. Control, 1971, 12, pp. 58-60.
[2]Allsop, R. E.: 'SIGCAP: A computer program for assessing the traffic capacity of signal-controlled road junctions', Traffic Eng. Control, 1976, 17, pp. 338-341.
[3]Little, J. D. C.: 'The synchronization of traffic signals by mixed integer-linear-programming', Oper. Res., 1966, 14, pp. 568-594.
[4]Little, J. D. C., Kelson, M. D., and Gartner, N. H.: 'MAXBAND: A Program for Setting Signals on Arteries and Triangular Networks', Transp. Res. Record, 1981, 795, pp. 40-46.
[5]Li, M. T., and Gan, A. C.: 'Signal timing optimization for oversaturated networks using TRANSYT-7F', Transp. Res. Record, 1999, 1683, pp. 118-126.
[6]Robertson, D.I.: 'TRANSYT method for area traffic control', Traffic Eng. Control, 1969, 10, pp. 276-281.
[7]Hunt, P. B., Robertson, D. L., and Bretherton, R. D.: 'The SCOOT on-line traffic signal optimization technique' Traffic Eng. Control, 1982, 23, pp. 190-192.
[8]Gartner, N. H.: 'OPAC: A demand-responsive strategy for traffic signal control', Transp. Res. Record, 1983, 906, pp. 75-81.
[9]Lim, J. H., Hwang, S. H., Suh, I. H. and Bien, Z.: 'Hierarchical optimal control of oversaturated urban traffic networks', International Journal of Control, 1981, 33, pp. 727-737.
[10]Davison, E. J., Ozguner U.: 'Decentralized control of traffic networks', IEEE Trans. Automat. Contr., 1983, 28, pp. 677-688.
[11]Papageorgiou M., Diakaki, C., Dinopoulou, V., Kotsialos, A., Wang, Y.: 'Review of Road Traffic Control Strategies', Proc. IEEE, Dec. 2003, pp. 2043-2067.
[12]List G. F., and Cetin M.: 'Modeling traffic signal control using Petri nets', IEEE Trans. on Intelligent Transportation Systems, 2004, 5, pp. 177- 187.
[13]Murata T.: 'Petri Nets: Properties, Analysis and Applications', Proc. IEEE, April, 1989, pp. 541-580, IEEE.
[14]Peterson J. L.: 'Petri net theory and the modeling of systems' (Prentice Hall, 1981)
[15]Febbraro A. D., Giglio D. and Sacco N.: 'Urban Traffic Control Structure Based on Hybrid Petri Nets', IEEE Trans. on Intelligent Transportation Systems, 2004, 5, pp. 224-237.

[16]Tolba C., Thomas P., ElMoudni, A., and Lefebvre, D.: 'Performances evaluation of the traffic control in a single crossroad by Petri nets', Proceedings of IEEE Emerging Technologies and Factory Automation, Lisbon, Portugal, Sep. 2003, pp. 157-160.

[17]Tzes, A., Seongho, K., and McShane, W. R.: 'Application of Petri Networks to Transportation Network Modeling', IEEE Transactions on Vehicular Technology, 1996, 45, pp. 391-400.

[18]Wang, J., Jin, C. and Deng, Y.: 'Performance analysis of traffic networks based on stochastic timed Petri net models', Proceedings of IEEE Engineering of Complex Computer Systems, Las Vegas, NE, U.S., Oct. 1999, pp.77-85.

[19]Wang, H., List, G.F., and DiCesare, F.: 'Modeling and evaluation of traffic signal control using timed Petri nets', Proceedings of IEEE Systems, Man and Cybernetics, Le Touquet, France, 1993, pp. 180-185.

[20]Lin, L., Nan, T., Xiangyang, M. and Fubing, S.: 'Implementation of Traffic Lights Control Based on Petri Nets', Proceedings of IEEE Intelligent Transportation Systems, Shanghai, China, 2003, pp.1087-1090.

[21]Dotoli, M., Fanti, M.P. and Lacobellis, G.: 'Validation of an Urban Traffic Network Model using Colored Timed Petri nets', IEEE Int. Conf. on. Systems, Man and Cybernetics, Waikoloa, HI, U.S., 2005, pp. 1347-1352.

[22]Huang, Y. S.: 'Design of Traffic Light Control System Using Statecharts', Comp. J., 2006, 49, 634-649.

[23]Jensen K.: Coloured Petri Nets. vol. 1 (Springer-Verlag, 1997)

[24]Jensen K.: Coloured Petri Nets. vol. 2 (Springer-Verlag, 1995)

[25] http://wiki.daimi.au.dk/cpntools/cpntools.wiki, Computer Tool for Coloured Petri Nets.

# 28

# Traffic Network Control Based on Hybrid System Modeling

Youngwoo Kim
*Nagoya University, Nagoya*
*Japan*

## 1. Introduction

With the increasing number of automobiles and complication of traffic network, the traffic flow control becomes one of significant economic and social issues in urban life. Many researchers have been involved in related researches in order to alleviate traffic congestion. From viewpoint of modeling, the existing scenarios can be categorized into the following two approaches:

(A1) Microscopic approach; and

(A2) Macroscopic approach.

The basic idea of Microscopic approach (A1) (2)is that the behavior of each vehicle is affected by neighboring vehicles, and the entire traffic flow is represented as statistical occurrences. The Cellular Automaton (CA) based model (3) (4) and (11) is widely known idea to represent the behavior of each vehicle. In the CA model, the road is discretized into many small cells. Each cell can be either empty or occupied by only one vehicle. The behavior of each vehicle in each cell is specified by the geometrical relationship with other vehicles together with some stochastic parameters. Although many simulation results based on these microscopic models showed high similarity to the measured real data, these approaches are not suitable for the large-scale traffic network modeling and its traffic light controller design. This is because they require enormous computational efforts to find all vehicles' behavior. Furthermore, the precise information on initial positions and speeds of all vehicles are usually not available in advance.
On the other hand, it has been a common strategy in the macroscopic approach (A2) (9) that the designer uses a fluid approximation model where the behavior of traffic flow is regarded as a continuous fluid with density $k(x,t)$ and volume $q(x,t)$ at location $x$ and time $t$. In this case, $k(x,t)$ and $q(x,t)$ must satisfy the following law of mass conservation;

$$\frac{\partial k(x,t)}{\partial t} + \frac{\partial q(x,t)}{\partial x} = 0. \tag{1}$$

Also, some relationship among $q$, $k$ and $v$, which are usually described by

$$q(x,t) = k(x,t)v(x,t), \tag{2}$$

is introduced together with the appropriate model of the $v(x,t)$, where $v(x,t)$ denotes the velocity of the traffic flow. By incorporating these two equations, the macroscopic behavior of

the traffic flow is uniquely decided. This model, however, is applicable only when the density of the traffic flow $k(x,t)$ is continuous. Although this model expresses well the behavior of the flow on the freeway, it is unlikely that this model can be applied to the urban traffic network which involves many discontinuities of the density coming from the existence of intersections controlled by traffic signals. In order to treat the discontinuity of the density in the macroscopic model, the idea of 'shock wave', which represents the progress of the boundary of two neighboring different density area, has been introduced in literature (6) (5) (7) (8). Although these approaches included judicious use of theoretical ideas for the flow dynamics, it is not straightforward to exploit them for the design of real-time traffic signal control since the flow model results in complicated nonlinear dynamics.

This paper presents a new method for the real-time traffic network control based on an integrated Hybrid Dynamical System (HDS) framework. The proposed method characterizes its synthetic modeling description. The information on geometrical traffic network is modeled by using Hybrid Petri Net (HPN), whereas the information on the behavior of traffic flow is modeled by means of Mixed Logical Dynamical Systems (MLDS) description. The former allows us to easily apply our method to complicated and wide range of traffic network due to its graphical understanding and algebraic manipulability. The latter allows us to represent physical features governing the dynamics of traffic flow and control mechanism for traffic congestion control employing the model predictive control policy (13).

Note that current traffic flow away from the signaler affects future traffic flow behavior. Through the model predictive control policy, we can construct the decentralized controller in a manner that each traffic outflow from the intersection or crosswalk is controlled and the information is shared with neighboring traffic controllers. A large-scale centralized traffic network controller is not appropriate because of the increased computational effort, synchronization in information processes and so on. In this case, the decentralized controller with model predictive control policy could be a realistic method.

In order to control large-scale traffic network with nonlinear dynamics, we formulate the traffic network control system based on the Mixed Integer NonLinear Programming (MINLP) problem. Generally, it is difficult to find the global optimal solution to the nonlinear programming problem. However, if the problem can be recast to the convex programming problem, the global optimal solution is easily found by applying an efficient method such as Steepest Descent Method (SDM). We use in this paper general performance criteria for traffic network control and show that although the problem contains non-convex constraint functions as a whole, the generated sub-problems are always included in the class of convex programming problem. In order to achieve high control performance of the traffic network with dynamically changing traffic flow, we adopt Model Predictive Control (MPC) policy. Note that MLDS formulation often encounters multiplication of two decision variables, and that without modification, it cannot be directly applied to MPC scheme. One way to avoid the multiplication is to introduce a new auxiliary variable to represent it. And then it becomes a linear system formally. However, as we described before, the introduction of discrete variables causes substantial computational amounts. A new method for this type of control problem is proposed. Although the system representation is nonlinear, MPC policy is successfully applied by means of the proposed Branch and Bound strategy.

After verification of the solution optimality, PWARX classifier is applied which describes a nonlinear feedback control law of the traffic control system. This implies we don't need a time-consuming searching process of a solver such as a Branch-and-Bound algorithm to solve

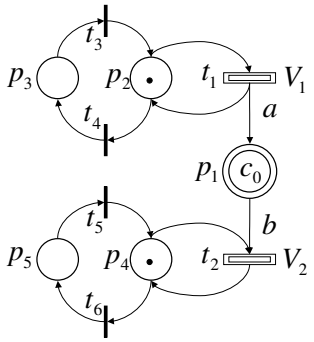



Fig. 1. Example of Hybrid Petri Net model

a mixed integer nonlinear programming (MINLP) problem, and furthermore the exactly same solutions are obtained in a very short time.

The problem we address in this paper is a special classification problem where the output $y$ is a 0-1 binary variable, and very good classification performance is desirable even with very large number of the introduced clusters. If we plot the observational data in a same cluster in the $x$-$y$ space, it will show always zero inclination, since we have a binary output, i.e., all components of $\theta$, $a$ and $b$ except for $f$ will be zeros. This implies we need consideration for a binary output. A new performance criterion is presented in this paper to consider not only a covariance of $\theta$, but also a covariance of $y$. The proposed method is a hierarchical classification procedure, where the cluster splitting process is introduced to the cluster with the worst classification performance (which includes 0-1 mixed values of $y$). The cluster splitting process is follows by the piecewise fitting process to compute the cluster guard and dynamics, and the cluster updating process to find new center points of the clusters. The usefulness of the proposed method is verified through some numerical experiments.

## 2. Modeling of Traffic Flow Control System (TFCS) based on HPN

The Traffic Flow Control System (TFCS) is the collective entity of traffic network, traffic flow and traffic signals. Although some of them have been fully considered by the previous studies, most of the previous studies did not simultaneously consider all of them. In this section, the HPN model is developed, which provides both graphical and algebraic descriptions for the TFCS.

### 2.1 Representation of TFCS as HPN

HPN is one of the useful tools to model and visualize the system behavior with both continuous and discrete variables. HPN is a structure of $N = (P, T, I_+, I_-, C, D)$. The set of places $P$ is partitioned into a subset of discrete places $P_d$ and a subset of continuous places $P_c$. The set of transition $T$ is partitioned into a subset of discrete transitions $T_d$ and a subset of continuous transitions $T_c$. The incidence matrix of the net is defined as $I(p,t) = I_-(p,t) - I_+(p,t)$, where $I_+(p,t)$ and $I_-(p,t)$ are the forward and backward incidence relationships between the transition $t$ and the place $p$ which follows and precedes the transition. We denote the preset (postset) of transition t as ${}^\bullet t$ ($t^\bullet$) and its restriction to continuous or discrete places as ${}^{(d)} t =$

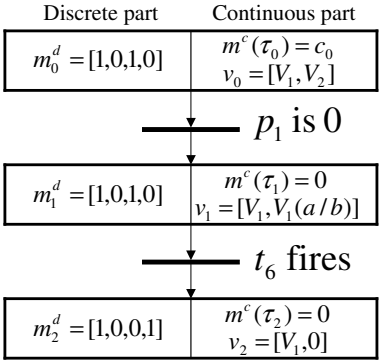



Fig. 2. Phase diagram of Hybrid Petri Net model

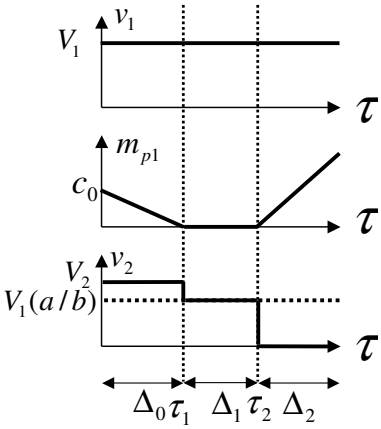



Fig. 3. Behavior of Hybrid Petri Net model

$^\bullet t \cap P_d$ or $^{(c)}\ t = {}^\bullet t \cap P_c$. Similar notation may be used for presets and postsets of places. The function $C$ and $D$ specify the firing speeds associated to the continuous transitions and the timing associated to the (timed) discrete transitions. For any continuous transition $t_i$, we let $C(t_i) = (v_i, V_i)$, where $v_i$ and $V_i$ represent the minimum and maximum firing speed of transition $t_i$. We associate to the timed discrete transition its firing delay, where the firing delay is short enough and the state is preserved until next sampling instant. The acquisition of firing sequence of the discrete transition at every sampling instant is applied to a variety of scheduling and control problems. The marking $M = [M_C|M_D]$ has both continuous ($m$ dimension) and discrete ($n$ dimension) parts.

Consider a simple example of First-Order Hybrid Petri Net model, Fig.1, where the control switch is represented with two discrete transitions and two discrete places connected to the continuous transition. In Fig.1, $p_1$ is the continuous place with the initial marking $m_c(\tau_0) = m_{p1} = c_0$, and $p_2$, $p_3$, $p_4$ and $p_5$ are the discrete places with the initial marking $m_d(\tau_0) = [m_{p2}, m_{p3}, m_{p4}, m_{p5}] = [1, 0, 1, 0]$. We assume $V_1 a < V_2 b$, where $V_1$ and $V_2$ are firing speed of $t_1$ and

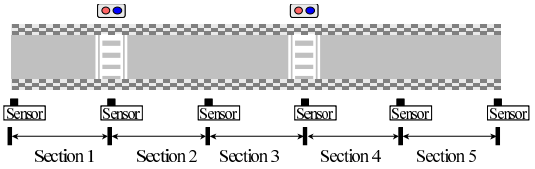



Fig. 4. Traffic network

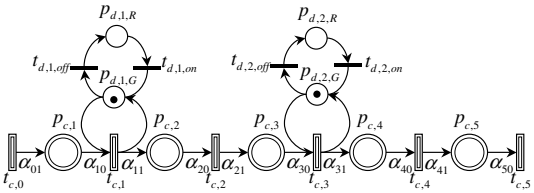



Fig. 5. Hybrid Petri Net model of traffic network

$t_2$ respectively, $a$ and $b$ are the arc weights given by the incidence relationship. The behavior is illustrated in Fig.2 and Fig.3.

Figure 5 shows the HPN model for the road of Fig.4. In Fig.5, each section $i$ of $l_i$-meters long constitutes the straight road, and two traffic lights are installed at the point of crosswalk. $p_c \in P_c$ represents each section of the road, and has maximum capacity (maximum number of vehicles). Also, $p_d \in P_d$ represents the traffic signal where green signal is indicated by an existence of a token. Note that each signal is supposed to have only two states 'go (green)' or 'stop (red)' for simplicity.$T$ is the set of continuous transitions which represent the boundary of two successive sections. $q_j(\tau)$ is the firing speeds assigned to transition $t_j \in T$ at time $\tau$. $q_j(\tau)$ represents the number of vehicles passing through the boundary per time unit of two successive sections(measuring position) at time $\tau$. The sensors to capture the number of the vehicles are supposed to be installed at every boundary of the section as show in Fig.4. The element of $I(p,t)$ is always 0 or $\alpha_{ij}$. $\alpha_{ij}$ is the number of traffic lanes in each section. Finally, $M^0$ is specified as the initial marking of the place $p \in P$. The net dynamics of HPN is represented by a simple first order differential equation for each continuous place $p_{c_i} \in P_c$ as follows:

if $p_{d,k} = {}^\bullet t_j$ is not null,

$$\frac{dm_{C,i}(\tau)}{dt} = \sum_{t_j \in p_{c_i}{}^\bullet \cup {}^\bullet p_{c_i}} I(p_{c_i}, t_j) \cdot q_j(\tau) \cdot m_{D,k}(\tau), \tag{3}$$

otherwise,

$$\frac{dm_{C,i}(\tau)}{dt} = \sum_{t_j \in p_{c_i}{}^\bullet \cup {}^\bullet p_{c_i}} I(p_{c_i}, t_j) \cdot q_j(\tau), \tag{4}$$

where $m_{C,i}(\tau)$ is the marking for the place $p_{c_i} (\in P_c)$ at time $\tau$, and $m_{D,k}(\tau)$ is the marking for the place $p_{d_k} (\in P_d)$. The equation (3) is transformed to its discrete-time version supposing

that $q_j(\tau)$ is constant during two successive sampling instants as follows:

$$m_{C,i}((\kappa+1)T_s) = m_{C,i}(\kappa T_s) + \sum_{t_j \in p_{c_i}{}^{\bullet} \cup {}^{\bullet} p_{c_i}} I(p_{c_i}, t_j) \cdot q_j(\kappa T_s) \cdot m_{D,j}(\kappa T_s) \cdot T_s. \quad (5)$$

where $\kappa$ is sampling index, and $T_s$ is sampling period.
Note that the transition $t$ is *enabled* at the sampling instant $\kappa T_s$ if the marking of its preceding discrete place $p_{d_j} \in P_d$ satisfies $m_{D,j}(\kappa) \geq I_+(p_{d_j}, t)$. Also if $t$ does not have any input (discrete) place, $t$ is always *enabled*.

### 2.2 Definition of flow $q_i$

In order to derive the flow behavior, the relationship among $q_i(\tau)$, $k_i(\tau)$ and $v_i(\tau)$ must be specified. One of the simple ideas is to use the well-known model

$$q_i(\tau) = \frac{(k_i(\tau) + k_{i+1}(\tau))}{2} \frac{v_i(\tau) + v_{i+1}(\tau)}{2} \quad (6)$$

supposing that the density $k_i(\tau)$ and $k_{i+1}(\tau)$, and average velocity $v_i(\tau)$ and $v_{i+1}(\tau)$ of the flow in $i$ and $(i+1)$th sections are almost identical. Then, by incorporating the velocity model

$$v_i(\tau) = v_{f_i} \cdot \left(1 - \frac{k_i(\tau)}{k_{jam}}\right), \quad (7)$$

with (6), the flow dynamics can be uniquely defined. Here, $k_{jam}$ is the density in which the vehicles on the roadway are spaced at minimum intervals (traffic-jammed), and $v_{f_i}$ is the maximum speed, that is, the velocity of the vehicle when no other vehicle exists in the same section.
If there exists no abrupt change in the density on the road, this model is expected to work well. However, in the urban traffic network, this is not the case due to the existence of the intersections controlled by the traffic signals. In order to treat the discontinuities of the density among neighboring sections (i.e. neighboring continuous places), the idea of 'shock wave'(10) is introduced as follows. We consider the case as shown in Fig.6 where the traffic density of $i$th section is lower than that of $(i+1)$th section in which the boundary of density difference designated by the dotted line is moving forward. Here, the movement of this boundary is called shock wave and the moving velocity of the shock wave $c_i(\tau)$ depends on the densities and average velocities of $i$th and $(i+1)$th sections as follows:

$$c_i(\tau) = \frac{v_i(\tau)k_i(\tau) - v_{i+1}(\tau)k_{i+1}(\tau)}{k_i(\tau) - k_{i+1}(\tau)}. \quad (8)$$

The traffic situation can be categorized into the following four types taking into account the density and shock wave.

(C1) $k_i(\tau) < k_{i+1}(\tau)$, and $c_i(\tau) > 0$,

(C2) $k_i(\tau) < k_{i+1}(\tau)$, and $c_i(\tau) \leq 0$,

(C3) $k_i(\tau) > k_{i+1}(\tau)$,

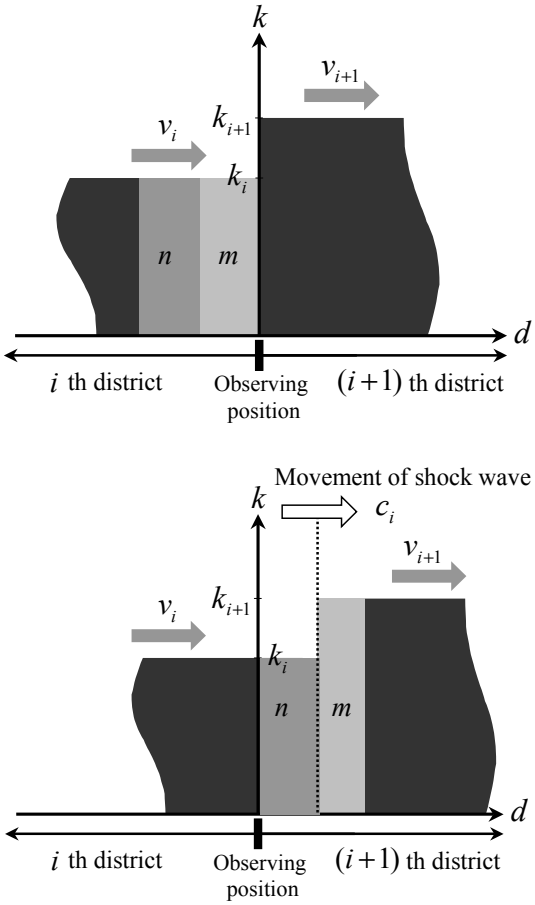



Fig. 6. Movement of shock wave in the case of $k_i(\tau) < k_{i+1}(\tau)$ and $c_i(\tau) > 0$

(C4) $k_i(\tau) = k_{i+1}(\tau)$ (no shock wave).

Firstly, in both cases of (C1) and (C2) where $k_i(\tau)$ is smaller than $k_{i+1}(\tau)$, the vehicles passing through the density boundary (dotted line) reduce their speeds. The movement of the shock wave is illustrated in Fig.6 ($c_i(\tau) > 0$) and Fig.7 ($c_i(\tau) \leq 0$). In Fig.6 and Fig.7, the 'measuring position' implies the position where transition $t_i$ is assigned. Since the traffic flow $q_i(\tau)$ represents the numbers of vehicles passing through the measuring position per unit time, in the case of (C1), it can be represented by $n + m$ in Fig.6, where $n$ and $m$ represent the area of the corresponding rectangular, i.e. the product of the $v_i(\tau)$ and $k_i(\tau)$. Similarly, in the case of (C2), $q_i(\tau)$ can be represented by $m$ in Fig.7.
These considerations lead to the following models:

in the case of (C1)

$$q_i(\tau) = v_i(\tau)k_i(\tau) \tag{9}$$
$$= v_{f_i}\left(1 - \frac{k_i(\tau)}{k_{jam}}\right)k_i(\tau), \tag{10}$$

in the case of (C2)

$$q_i(\tau) = v_{i+1}(\tau)k_{i+1}(\tau) \tag{11}$$
$$= v_{f_{i+1}}\left(1 - \frac{k_{i+1}(\tau)}{k_{jam}}\right)k_{i+1}(\tau). \tag{12}$$

In the cases of (C3) and (C4) where $k_i(\tau)$ is greater than $k_{i+1}(\tau)$, the vehicles passing through the density boundary come to accelerate. In this case, the flow can be well approximated by taking into account the average density of neighboring two sections. This is intuitively because the difference of the traffic density is going down. Then in the cases of (C3) and (C4), the traffic flow can be formulated as follows:

in the cases of (C3) and (C4),

$$q_i(\tau) = \left(\frac{k_i(\tau) + k_{i+1}(\tau)}{2}\right)v_f(\tau)\left(1 - \frac{k_i(\tau) + k_{i+1}(\tau)}{2k_{jam}}\right). \tag{13}$$

As the results, the flow model (9) $\sim$ (13) taking into account the discontinuity of the density can be summarized as follows:

$$q_i(\tau) = \begin{cases} \left(\frac{k_i(\tau)+k_{i+1}(\tau)}{2}\right)v_f\left(1 - \frac{k_i(\tau)+k_{i+1}(\tau)}{2k_{jam}}\right) \\ \quad if\ k_i(\tau) \geq k_{i+1}(\tau) \\ v_{f_i}\left(1 - \frac{k_i(\tau)}{k_{jam}}\right)k_i(\tau) \\ \quad if\ k_i(\tau) < k_{i+1}(\tau) \text{ and } c(\tau) > 0 \\ v_{f_{i+1}}\left(1 - \frac{k_{i+1}(\tau)}{k_{jam}}\right)k_{i+1}(\tau) \\ \quad if\ k_i(\tau) < k_{i+1}(\tau) \text{ and } c(\tau) \leq 0 \end{cases}. \tag{14}$$

Figure 8 shows the HPN model of the $i$th intersection, where the notation for other than southwardly entrance lane is omitted. In Fig.8, $l_{j,E}$, $l_{j,W}$, $l_{j,S}$ and $l_{j,N}$ are the length of the corresponding districts, and the numbers of the vehicles in the districts are obtained as for example $p_{c,j_{IS}}(\tau) = k_{j_{IS}}(\tau) \cdot l_{j,S}$. The vehicles in $p_{c,j_{IS}}$ are assumed to have the probability $\zeta_{j,SW}$, $\zeta_{j,SN}$, and $\zeta_{j,SE}$ to proceed into the district corresponding to $p_{c,j_{OW}}$, $p_{c,j_{ON}}$, and $p_{c,j_{OE}}$ as follows,

$$k_{j_{SW}}(\tau) = k_{j_{IS}}(\tau)\zeta_{j,SW}(\tau), \tag{15}$$
$$k_{j_{SN}}(\tau) = k_{j_{IS}}(\tau)\zeta_{j,SN}(\tau), \tag{16}$$
$$k_{j_{SE}}(\tau) = k_{j_{IS}}(\tau)\zeta_{j,SE}(\tau). \tag{17}$$

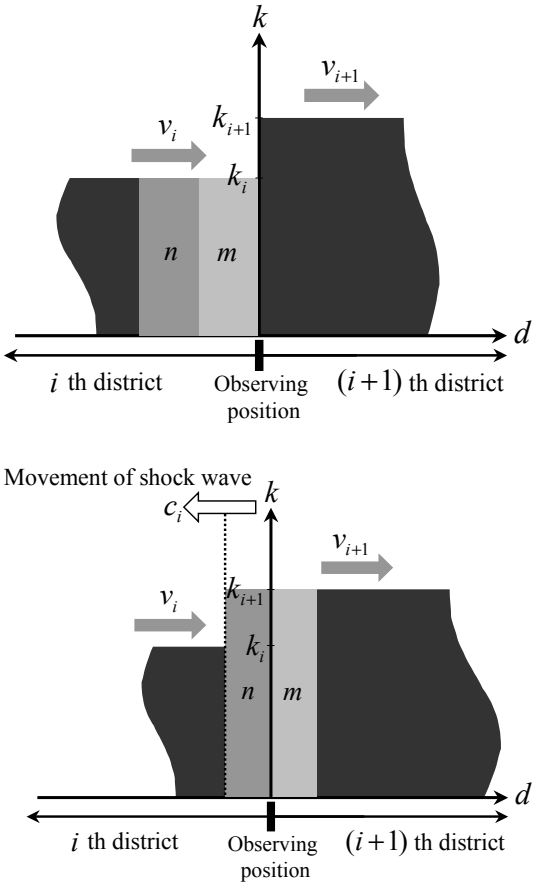



Fig. 7. Movement of shock wave in the case of $k_i(\tau) < k_{i+1}(\tau)$ and $c_i(\tau) \leq 0$

Note that these probabilities are determined by the traffic network structure, and satisfy at $\tau$,

$$0 \leq \zeta_{j,SW}(\tau) \leq 1, \tag{18}$$
$$0 \leq \zeta_{j,SN}(\tau) \leq 1, \tag{19}$$
$$0 \leq \zeta_{j,SE}(\tau) \leq 1, \tag{20}$$
$$\zeta_{j,SW}(\tau) + \zeta_{j,SN}(\tau) + \zeta_{j,SE}(\tau) = 1. \tag{21}$$

Therefore, the traffic flows of the three directions are represented with

$$q\left(k_{j_{SN}}(\tau), k_{j_{ON}}(\tau)\right), \tag{22}$$
$$q\left(k_{j_{SW}}(\tau), k_{j_{OW}}(\tau)\right), \tag{23}$$
$$q\left(k_{j_{SE}}(\tau), k_{j_{OS}}(\tau)\right). \tag{24}$$

Note that the mutual exclusion of the same traffic light with the intersecting road is represented in the Fig.8.

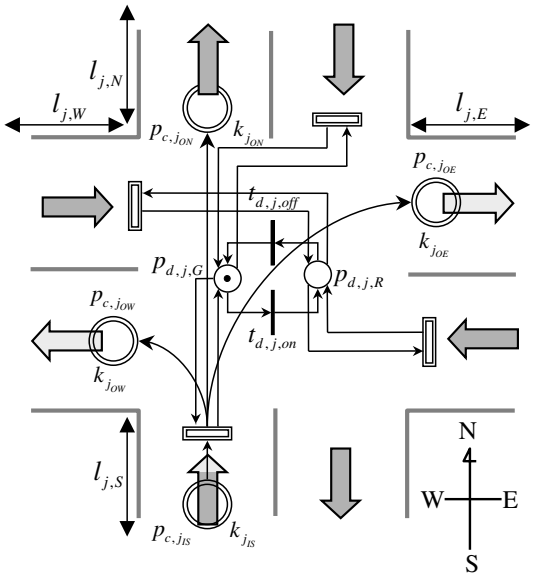



Fig. 8. Hybrid Petri Net model of intersection

### 2.3 Derived flow model

In this subsection, we confirm the effectiveness of the proposed traffic flow model developed in the previous subsection by comparing it with the microscopic model. The usefulness of Cellular Automaton (CA) in representing the traffic flow behavior was investigated in (3). Some of well-known traffic flow simulators such as TRANSIMS and MICROSIM are based on CA model.

The essential property of CA is characterized by its lattice structure where each cell represents a small section on the road. Each cell may include one vehicle or not. The evolution of CA is described by some rules which describe the evolution of the state of each cell depending on the states of its adjacent cells.

The evolution of the state of each cell in CA model can be expressed by

$$n_j(\tau+1) = n_j^{in}(\tau)(1-n_j(\tau)) - n_j^{out}(\tau), \tag{25}$$

where $n_j(\tau)$ is the state of cell $j$ which represents the occupation by the vehicle ($n_j(\tau) = 0$ implies that the $j$th cell is empty, and $n_j(\tau) = 1$ implies that a vehicle is present in the $j$th cell at $\tau$). $n_j^{in}(\tau)$ represents the state of the cell from which the vehicle moves to the $j$th cell, and $n_j^{out}(\tau)$ indicates the state of the destination cell leaving from the $j$th cell. In order to find $n_j^{in}(\tau)$ and $n_j^{out}(\tau)$, some rules are adopted as follows:

Step 1, Acceleration rule: All vehicles, that have not reached at the speed of maximum speed $v_f$, accelerate its speed $v_{\langle j\rangle}(\tau)$ by one unit velocity $v_{unit}$ as follows:

$$v_{\langle j\rangle}(\tau+\Delta\tau) \equiv v_{\langle j\rangle}(\tau) + v_{unit}. \tag{26}$$

Step 2, Safety distance rule: If a vehicle has $e$ empty cells in front of it, then the velocity at the next time instant $v_{\langle j \rangle}(\tau + \Delta\tau)$ is restricted as follows:

$$v_{\langle j \rangle}(\tau + \Delta\tau) \equiv \min\{e, v_{\langle j \rangle}(\tau + \Delta\tau)\}. \tag{27}$$

Step 3, Randomization rule: With probability $p$, the velocity is reduced by one unit velocity as follows:

$$v_{\langle j \rangle}(\tau + \Delta\tau) \equiv v_{\langle j \rangle}(\tau + \Delta\tau) - p \cdot v_{unit}. \tag{28}$$

Figure 9 shows the behavior of traffic flow obtained by applying the CA model to the two successive sections which is 450[m] long. The parameters used in the simulation are as follows: computational interval $\Delta\tau$ is 1 [sec], each cell in the CA is assigned to 4.5 [m]-long interval on the road, maximum speed $v_f$ is 5 (cells/$\Delta\tau$), which is equivalent to 81 [Km/h] (=4.5[m/cell] $\cdot$ 5 [cells/$\Delta\tau$] $\cdot$ 3600[sec]/1000). The left figure of Fig.9 shows the obtained relationship among normalized flow $q_i(\tau)$ and densities $k_i(\tau)$ and $k_{i+1}(\tau)$. The right small figure is the abstracted illustration of the real behavior.
First of all, we look at the behavior along the edge $a$ in the right figure which implies the case that the traffic signal is changed from red to green. At the point of $k_i(\tau) = 0$ and $k_{i+1}(\tau) = 0$, the traffic flow $q_i(\tau)$ becomes zero since there is no vehicle in both $i$th and $(i+1)$th section. Then, $q_i(\tau)$ is proportionally increased as $k_i(\tau)$ increases, and reaches the saturation point ($k_i(\tau) = 0.9$). Next, we look at the behavior along the edge $b$ which implies that the $i$th section is fully occupied. In this case, the maximum flow is measured until the density of the $(i+1)$th section is reduced by 50% (i.e. $k_{i+1}(\tau) = 0.5$), and after that the flow goes down according to the increase of $k_{i+1}(\tau)$. Although CA model consists of quite simple procedures, it can show quite natural traffic flow behavior.
On the other hand, Fig.10 shows the behavior in case of using HPN where the proposed flow model given by (14) is embedded. We can see that Fig.10 shows the similar characteristics to Fig.9, especially, the saturation characteristic is well represented despite of the use of macroscopic model. As another simple modeling strategy, we consider the case that the average of two $k_i(\tau)$ and $k_{i+1}(\tau)$ are used to decide the flow $q_i(\tau)$ (i.e. use (13) ) for all cases. Figure 11 shows the behavior in case of using HPN where the flow model is supposed to be given by (13) for all cases. Although the $q_i(\tau)$ shows similar characteristics in the region of $k_i(\tau) \geq k_{i+1}(\tau)$, at the point of $k_i(\tau) = 0$ and $k_{i+1}(\tau) = k_{jam}$, $q_i(\tau)$ takes its maximum value. This obviously contradicts to the natural flow behavior.
Before concluding this subsection, it is worthwhile to compare the computational amount. In case of using CA, it took 140 seconds to construct the traffic flow dynamics using Athlon XP 2400 and Windows 2000, while only 0.06 seconds in case of using HPN and (14).

## 3. Model Predictive Control of Traffic Network Control based on MLDS description

The Receding Horizon Control (RHC) or Model Predictive Control (MPC) is one of well - known paradigms for optimizing the systems with constraints and uncertainties. In RHC paradigms, the solutions are elements of finite dimensional vector spaces, and finite-horizon optimization is carried out in order to provide stability or performance analysis. However, the application of RHC has been mainly restricted to the system with sufficiently long sampling interval, since finite-horizon optimization is computationally demanding.
This chapter firstly formulate the traffic flow model developed in chapter 2 in the form of MLDS description coupled with RHC strategy, where wide range of traffic flow is considered.

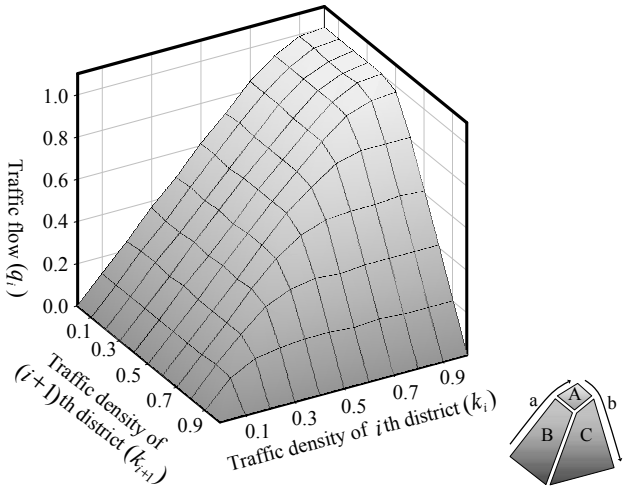



Fig. 9. Traffic flow behavior obtained from CA model

This formulation is recast to the canonical form of 0-1 Mixed Integer Linear Programming (MILP) problem to optimize its behavior and a new Branch and Bound (B&B) based algorithm is presented in order to abate computational cost of MILP problem.

### 3.1 MLDS representation of TCCS based on Piece-Wise Affine (PWA) linearization of traffic flow

Since TCCS is the hybrid dynamical system including both continuous traffic flow dynamics and discrete aspects for traffic light signal control, some algebraic formulation, which handles both continuous and discrete behaviors, must be introduced. The MLDS description has been developed to describe such class of systems considering some constraints shown in the form of inequalities and can be combined with powerful search engine such as Mixed Integer Linear Programming (MILP).
The MLDS (12) description can be formalized as following.

$$\begin{aligned} \boldsymbol{x}(\tau+1) &= \boldsymbol{A_\tau x}(\tau) + \boldsymbol{B_{1\tau}u}(\tau) \\ &\quad + \boldsymbol{B_{2\tau}}\delta(\tau) + \boldsymbol{B_{3\tau}z}(\tau) \qquad (29) \\ \boldsymbol{y}(\tau) &= \boldsymbol{C_\tau x}(\tau) + \boldsymbol{D_{1\tau}u}(\tau) \\ &\quad + \boldsymbol{D_{2\tau}}\delta(t) + \boldsymbol{D_{3\tau}}(\tau) \qquad (30) \\ \boldsymbol{E_{2\tau}}\delta(\tau) &+ \boldsymbol{E_{3\tau}z}(\tau) \leq \\ &\quad \boldsymbol{E_{1\tau}u}(\tau) + \boldsymbol{E_{4\tau}x}(\tau) + \boldsymbol{E_{5\tau}} \qquad (31) \end{aligned}$$

In MLDS formulation, (29), (30) and (31) are state equation, output equation and constraint inequality, respectively, where $x$, $y$ and $u$ are the state, output and input variable, whose components are constituted by continuous and/or 0-1 binary variables, $\delta(\tau) \in \{0,1\}$ and $z(\tau) \in \Re$

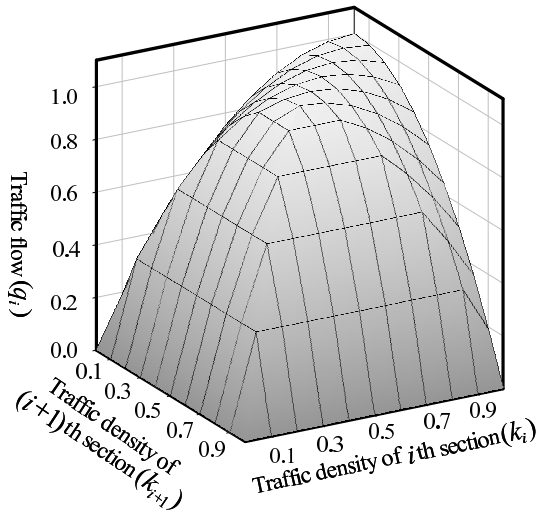



Fig. 10. Traffic flow behavior obtained from the proposed traffic flow model

represent auxiliary logical and continuous variables. By introducing the constraint inequality of (31), non-linear constraints as (14) can be transformed to the computationally tractable Piece-Wise Affine (PWA) forms.
The traffic flow of Fig. 9 can be approximated as the right figure of Fig. 9 which consists of three planes as follows,

Plane A: The traffic flow $q_i$ is saturated ($k_i(\tau) \leq a$ and $k_{i+1}(\tau) < (k_{jam} - a)$)

Plane B: The traffic flow $q_i$ is mainly affected by the quantity of traffic density $k_i(\tau)$ ($k_i(\tau) < a$ and $k_i(\tau) + k_{i+1} < k_{jam}$)

Plane C: The traffic flow $q_i$ is mainly affected by the quantity of traffic density $k_{i+1}(\tau)$ ($k_{i+1}(\tau) \leq k_{jam} - a$ and $k_i(\tau) + k_{i+1} \leq k_{jam}$)

where $a$ is the threshold value to describe saturation characteristic of traffic flow that if $k_i(\tau) > a$ and/or $k_{i+1}(\tau) < k_{jam} - a$, the value of $q_i(\tau)$ hovers at its maximum value $q_{max}$.

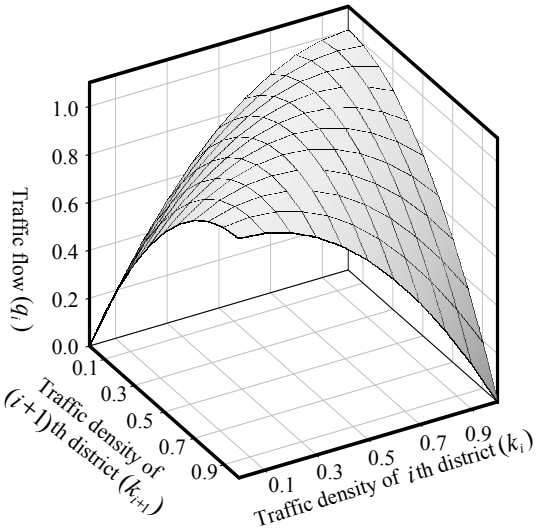



Fig. 11. Traffic flow behavior obtained by averaging $k_i$ and $k_{i+1}$

Fig.12 shows three planes partitioned by introducing three auxiliary variables $\delta_{P,i,1}(\tau)$, $\delta_{P,i,2}(\tau)$ and $\delta_{P,i,3}(\tau)$ which are defined as follows,

$$
\begin{aligned}
&[\delta_{P,i,1}(\tau) = 1] \\
&\quad \leftrightarrow \left\{ \begin{array}{lcl} k_i(\tau) & \geq & a \\ k_{i+1}(\tau) & \leq & k_{jam} - a \end{array} \right. && (32) \\
&[\delta_{P,i,2}(\tau) = 1] \\
&\quad \leftrightarrow \left\{ \begin{array}{lcl} k_i(\tau) & \leq & a - \varepsilon \\ k_i(\tau) + k_{i+1}(\tau) & \leq & k_{jam} \end{array} \right. && (33) \\
&[\delta_{P,i,3}(\tau) = 1] \\
&\quad \leftrightarrow \left\{ \begin{array}{lcl} k_{i+1}(\tau) & \geq & k_{jam} - a + \varepsilon \\ k_i(\tau) + k_{i+1}(\tau) & \geq & k_{jam} + \varepsilon \end{array} \right. && (34) \\
&\delta_{P,i,1}(\tau) + \delta_{P,i,2}(\tau) + \delta_{P,i,3}(\tau) = 1 && (35)
\end{aligned}
$$

where $\varepsilon$ is small tolerance to consider equality sign.
Therefore, the traffic flow $q_i(\tau)$ can be rewritten in a compact form as follows

$$
\begin{aligned}
q_i(\tau) \quad &= \quad q_{max}\delta_{P,i,1}(\tau) + \frac{q_{max}k_i(\tau)}{a}\delta_{P,i,2}(\tau) \\
&\qquad + \frac{q_{max}(1 - k_{i+1}(\tau))}{a}\delta_{P,i,3}(\tau) && (36) \\
\sum_{i=1}^{3} \quad &\delta_{P,i,j}(\tau) = 1
\end{aligned}
$$

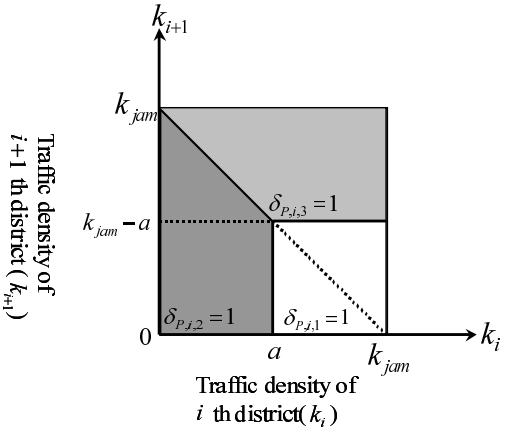



Fig. 12. Assignation of planes by introducing auxiliary variables

where $0 \leq k_i(\tau) \leq k_{jam}$, $0 \leq k_{i+1} \leq k_{jam}$ $(= 1)$, $q_{max}$ is the maximum value of traffic flow. Figure xxx shows the piece-wise affine (PWA) dynamics of the traffic flow model developed in the previous chapter where $a = 0.3$ and $q_{max} = 1$.
The equations (32) to (34) can be generalized as (37) and (38), and transformed to inequality as (39) The equations (32) and (34) can be generalized as (37) and (38), and transformed to inequality as (39)

$$[\delta_{P,i,j}(\tau) = 1] \quad \leftrightarrow \quad \left[\begin{bmatrix} k_i \\ k_{i+1} \end{bmatrix} \in \ell_j\right] \tag{37}$$

$$\ell_j \quad = \quad \left\{\begin{bmatrix} k_i \\ k_{i+1} \end{bmatrix} : \boldsymbol{S_j k_i}(\tau) \leq \boldsymbol{T_j}\right\} \tag{38}$$

where $\boldsymbol{k_i}(\tau) = [k_i(\tau) k_{i+1}(\tau)]^T$ and $\boldsymbol{S_j}$ and $\boldsymbol{T_j}$ are the matrices with suitable dimensions which satisfy

$$\boldsymbol{S_j k_i}(\tau) \quad - \quad \boldsymbol{T_j} \leq \boldsymbol{M_j}^*[1 - \delta_{P,i,j}(\tau)], \tag{39}$$

$$\boldsymbol{M_j^*} \quad \stackrel{\triangle}{=} \quad \max_{\boldsymbol{k_i} \in \ell_j} \boldsymbol{S_j k_i(\tau)} - \boldsymbol{T_j}. \tag{40}$$

The traffic flow $q_i(\tau)$ of (37) is the relationship between $\boldsymbol{k_i}(\tau)$ and $\boldsymbol{\delta_{P,i}}(\tau) = [\delta_{P,i,1}(\tau)\ \delta_{P,i,2}(\tau)\ \delta_{P,i,3}(\tau)]$ which can be rewritten as follows,

$$q_i(\tau) \quad = \quad f(\boldsymbol{\delta_{P,i}}(\tau), \boldsymbol{k_{i+1}}(\tau)) \tag{41}$$

$$\quad = \quad \sum_{j=1}^{3} (\boldsymbol{F_i^j}(\tau)\boldsymbol{k_i}(\tau) + \boldsymbol{H_i^j})\delta_{P,i,j}(\tau) \tag{42}$$

where $\boldsymbol{\delta_{P,i}} = [\delta_{P,i,1}, \delta_{P,i,2}, \delta_{P,i,3}]'$. In these equations, each pair of $\boldsymbol{F_i^j}$ and $\boldsymbol{H_i^j}$ represents the corresponding domain of Fig. 12 as follows,

$$\begin{aligned}
\boldsymbol{F_i^1} &= [\ 0 \quad 0\ ] && (43)\\
\boldsymbol{H_i^1} &= q_{max} && (44)\\
\boldsymbol{F_i^2} &= [\ \tfrac{q_{max}}{a} \quad 0\ ] && (45)\\
\boldsymbol{H_i^2} &= 0 && (46)\\
\boldsymbol{F_i^3} &= [\ 0 \quad -\tfrac{q_{max}}{a}\ ] && (47)\\
\boldsymbol{H_i^3} &= \frac{q_{max}}{a} && (48)
\end{aligned}$$

The traffic flow $\boldsymbol{z_i}(\tau) = [z_{i,1}(\tau)\ z_{i,2}(\tau)\ z_{i,3}(\tau)]$ in consideration of the binary input $u_i(\tau) \in \{0,1\}$ for traffic light control can be represented by

$$\begin{aligned}
z_{i,j}(\tau) &\leq M_i u_i(\tau)\delta_{P,i,j}(\tau), && (49)\\
z_{i,j}(\tau) &\geq m_i u_i(\tau)\delta_{P,i,j}(\tau), && (50)\\
z_{i,j}(\tau) &\leq \boldsymbol{F_i^j k_i}(\tau) + \boldsymbol{H_i^j} \\
&\quad -m_i(1 - u_i(\tau)\delta_{P,i,j}(\tau)), && (51)\\
z_{i,j}(\tau) &\geq \boldsymbol{F_i^j k_i}(\tau) + \boldsymbol{H_i^j} \\
&\quad -M_i(1 - u_i(\tau)\delta_{P,i,j}(\tau)). && (52)
\end{aligned}$$

where $M_i$ and $m_i$ are respectively

$$\begin{aligned}
M_i &= \max_{\boldsymbol{k_i}(\tau)\in\ell_j} \left\{ \boldsymbol{F_i^j k_i}(\tau) + \boldsymbol{H_j} \right\}, && (53)\\
m_i &= \min_{\boldsymbol{k_i}(\tau)\in\ell_j} \left\{ \boldsymbol{F_i^j k_i}(\tau) + \boldsymbol{H_j} \right\}. && (54)
\end{aligned}$$

The product $u_i(\tau)\ \delta_{P,i,j}(\tau)$ can be replaced by an auxiliary logical variable $\delta_{M,i,j}(\tau) = u_i(\tau)\ \delta_{P,i,j}(\tau)$ in order to make it tractable to deal with MILP problem. Then this relationship can be equivalently represented as follows,

$$\begin{aligned}
-u_i(\tau) + \delta_{M,i,j}(\tau) &\leq 0, && (55)\\
-\delta_{P,i,j}(\tau) + \delta_{M,i,j}(\tau) &\leq 0, && (56)\\
u_i(\tau) + \delta_{P,i,j}(\tau) + \delta_{M,i,j}(\tau) &\leq 1. && (57)
\end{aligned}$$

Therefore, the MLDS description for the proposed system can be formalized as follows,

$$\begin{aligned}
\boldsymbol{x}(\kappa+1) &= \boldsymbol{A x}(\kappa) + \boldsymbol{B z}(\kappa), && (58)\\
\boldsymbol{z}(\kappa) &= \boldsymbol{C_1} diag(\boldsymbol{u}(\kappa))\boldsymbol{C_2}\boldsymbol{\delta}(\kappa), && (59)\\
\boldsymbol{E_2}\boldsymbol{\delta}(\kappa) &+ \boldsymbol{E_3 z}(\kappa) \\
&\leq \boldsymbol{E_1 u}(\kappa) + \boldsymbol{E_4 x}(\kappa) + \boldsymbol{E_5} && (60)
\end{aligned}$$

where the element $x_i(\kappa)$ of $\boldsymbol{x}(\kappa) \in \Re^{|P|}$, is marking of the place $p_{c_i}$ at the sampling instance $\kappa$, the element $u_i(\kappa)(\in \{0,1\})$ of $\boldsymbol{u}(\kappa) \in Z^{|T|}$, is the signal of traffic light installed at $i$th district

and $\delta(\kappa)=[\delta_{\boldsymbol{P}}(\kappa),\ \delta_{\boldsymbol{M}}(\kappa)]^{'}$. Note that if there is no traffic light installed at $i$th district, $u_i(\kappa)$ is always set to 1. And $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C_1}$, $\boldsymbol{C_2}$, $\boldsymbol{E_1}$, $\boldsymbol{E_2}$, $\boldsymbol{E_3}$, $\boldsymbol{E_4}$ and $\boldsymbol{E_5}$ are the matrices with appropriate dimensions.

### 3.2 Model predictive control policy for traffic network control

The traffic system is large-scale dynamical system with uncertainty in the behavior of each car. In order to develop efficient traffic light control system, a wide range of traffic flow should be fully considered. In this subchapter, model predictive control policy for traffic light control is applied to the traffic flow model developed in the previous chapter. In RHC scheme, an input for next sampling period is decided based on the prediction for next several periods called the prediction horizon. This allows for the fact that the spatially changing dynamics of traffic flow are represented by temporal behavior over prediction horizon, since traffic flow can be considered as probabilistic time-series behavior.

The equation (58) can be modified, enumerating the state and input variables for the future periods as follows,

$$\begin{aligned}\boldsymbol{x}(\kappa+\lambda|\kappa) \quad &= \quad \boldsymbol{A^{\kappa}}x(\kappa) \\ &\quad + \sum_{\eta=0}^{\lambda-1}\{\boldsymbol{A^{\eta}}(\boldsymbol{BC_1}(diag(\boldsymbol{u}(\kappa+\lambda-1-\eta|\kappa))) \\ &\qquad\qquad \cdot \boldsymbol{C_2}\delta(\kappa+\lambda-1-\eta|\kappa))\}\end{aligned} \tag{61}$$

where $x(\kappa+\lambda|\kappa)$ denotes the predicted state vector at time $\kappa+\lambda$, obtained by applying the input sequence $u(\lambda|\kappa)=u(\kappa),\cdots,u(\kappa+\lambda)$ to (58) starting from the state $x(\lambda|\kappa)=x(\kappa)$.

Now we consider following requirements that usually appear in the traffic light control problems.

(R1) Maximizes traffic flow over entire traffic network.

(R2) Avoid frequent change of traffic signal.

(R3) Avoid concentration of traffic flow in a certain district.

These requirements can be realized by minimizing the following objective function.

$$\begin{aligned} & J(\boldsymbol{u}(\lambda|\kappa),\cdots,\boldsymbol{u}(\lambda+N_I|\kappa) \\ & \quad ,\boldsymbol{x}(\lambda|\kappa),\cdots,\boldsymbol{x}(\lambda+N_I|\kappa) \\ & \quad ,\boldsymbol{\delta}(\lambda|\kappa),\cdots,\boldsymbol{\delta}(\lambda+N_I|\kappa)) \\ = & \quad \sum_{\lambda=1}^{N}\Bigg\{-\sum_i w_{1,i}\Bigg\{\Bigg(\boldsymbol{\Theta_i}\begin{bmatrix} x_i(\lambda|\kappa)/l_i \\ x_{i+1}(\lambda|\kappa)/l_{i+1}\end{bmatrix} \\ & \quad +\boldsymbol{\Phi}\Bigg)^{'}\boldsymbol{\delta_{M,i}}(\lambda|\kappa)\Bigg\} \\ & \quad -\sum_i w_{2,i}\Big\{1-\big|u_i(\lambda|\kappa)-u_i(\lambda+1|\kappa)\big|\Big\} \\ & \quad +\sum_i w_{3,i}\left\{\left|\frac{x_i(\lambda|\kappa)}{l_i}-\frac{x_{i+1}(\lambda|\kappa)}{l_{i+1}}\right|\right\}\Bigg\}\end{aligned} \tag{62}$$

where

$$\boldsymbol{\Theta_i} = \begin{bmatrix} 0 & 0 \\ \frac{q_{max}}{a} & 0 \\ 0 & -\frac{q_{max}}{a} \end{bmatrix} \tag{63}$$

$$\boldsymbol{\Phi_i} = \begin{bmatrix} q_{max} \\ 0 \\ \frac{q_{max} k_i(\tau)}{a} \end{bmatrix} \tag{64}$$

and $w_{1,i}$, $w_{2,i}$ and $w_{3,i}$ are positive weight values for $i$th district which satisfy $w_{1,i} + w_{2,i} + w_{3,i} = 1$, and $0 \leq w_{1,i} \leq 1$, $0 \leq w_{2,i} \leq 1$ and $0 \leq w_{3,i} \leq 1$. In (62), the three terms correspond to the requirement (R1), (R2) and (R3) in order.
Therefore, the optimization problem can be formulated as follows:

$$\begin{gathered} find\ \boldsymbol{\delta}(\boldsymbol{\lambda}|\boldsymbol{\kappa}) = [\ \boldsymbol{\delta_P}(\boldsymbol{\lambda}|\boldsymbol{\kappa}), \boldsymbol{\delta_M}(\boldsymbol{\lambda}|\boldsymbol{\kappa})\ ]' \\ which\ minimizes\ (62) \\ subject\ to\ (32)\ \text{to}\ (61) \end{gathered}$$

The objective function (62) contains absolute functions, which are not directly trac table for MILP formulation. Therefore, these absolute functions are equivalently represented as follows:

$$\begin{aligned} & J(\boldsymbol{u}(\lambda|\kappa), \cdots, \boldsymbol{u}(\lambda + N_I|\kappa) \\ & \quad , \boldsymbol{x}(\lambda|\kappa), \cdots, \boldsymbol{x}(\lambda + N_I|\kappa) \\ & \quad , \boldsymbol{\delta}(\lambda|\kappa), \cdots, \boldsymbol{\delta}(\lambda + N_I|\kappa)) \\ = & \sum_{\lambda=1}^{N} \Bigg\{ - \sum_i w_{1,i} \Bigg\{ \left( \boldsymbol{\Theta_i} \begin{bmatrix} x_i(\lambda|\kappa)/l_i \\ x_{i+1}(\lambda|\kappa)/l_{i+1} \end{bmatrix} \right. \\ & \quad \left. + \boldsymbol{\Phi} \right)' \boldsymbol{\delta_{M,i}}(\lambda|\kappa) \Bigg\} \\ & \quad - \sum_i w_{2,i} \Big\{ 1 - \left( e^+_{u,i}(\lambda|\kappa) + e^-_{u,i}(\lambda|\kappa) \right) \Big\} \\ & \quad + \sum_i w_{3,i} \Big\{ \left( e^+_{x,i}(\lambda|\kappa) + e^-_{x,i}(\lambda|\kappa) \right) \Big\} \Bigg\} \end{aligned} \tag{65}$$

where

$$u_i(\lambda|\kappa) - u_i(\lambda + 1|\kappa) = e^+_{u,i}(\lambda|\kappa) - e^-_{u,i}(\lambda|\kappa), \tag{66}$$

$$\frac{x_i(\lambda|\kappa)}{l_i} - \frac{x_{i+1}(\lambda|\kappa)}{l_{i+1}} = e^+_{x,i}(\lambda|\kappa) - e^-_{x,i}(\lambda|\kappa), \tag{67}$$

$$e^+_{u,i}(\lambda|\kappa) \geq 0 \quad , \quad e^-_{u,i}(\lambda|\kappa) \geq 0, \tag{68}$$

$$e^+_{x,i}(\lambda|\kappa) \geq 0 \quad , \quad e^-_{x,i}(\lambda|\kappa) \geq 0. \tag{69}$$

The MLDS formulation coupled with RHC scheme can be transformed to the canonical form of 0-1 Mixed Integer Linear Programming (MILP) problem to find optimal solution for the objective function (65).
Note that the requirements (R1), (R2) and (R3) also can be realized by solving Mixed Integer Quadratic Programming (MIQP) problem instead of solving Mixed Integer Linear Programming (MILP) problem as in this paper. However, since RHC scheme is by nature computationally demanding as is witnessed by many applications, the computational effort is one of the key performance criteria. In this regard, this paper firstly handles MILP problem with the objective function of (65) that has faster procedure in solution method than conventional MIQP problems have. And next subchapter, this paper presents a new algorithm designed to reduce computational amount in 0-1 MILP problems.

## 4. Convexity Analysis

The problem we formulated in the previous section is recast to the convex programming problem in this subsection. The convex programming problem, where the constraint and objective functions are convex, has become quite popular recently for a number of reasons. Some of them are summarized as follows : (1) The global optimality is guaranteed for the obtained solution, (2) The attractive algorithm is easily applied, obtaining the solution with high speed due to the simple structure of the problem, and (3) The bounding process can be efficiently applied for the MINLP problem.

### 4.1 Convexity Analysis

In this subsection we first introduce the well-known performance criteria of traffic network control system and show they can be realized with convex functions. The following performance criteria are introduced in this paper: (1) maximization of traffic flow and (2) minimization of traffic density difference between neighboring districts. These criteria are numerically represented as follows,

$$f = \sum_{\eta=0}^{H-1} \sum_{i=0}^{N-1} z_i(\kappa+\eta), \tag{70}$$

and

$$f = \sum_{\eta=1}^{H} \sum_{i=0}^{M-1} |x_i(\kappa+\eta|\kappa) - x_{i+1}(\kappa+\eta|\kappa)|, \tag{71}$$

where $H$ is the predictive horizon and $N$ and $M$ are the dimension of $\boldsymbol{x}(\kappa)$ and $\boldsymbol{z}(\kappa)$, respectively.
In order to verify the convexity of (70), we first show the traffic flow dynamics with three modes are convex functions at each mode, and show that these dynamics at each mode are continuous to the neighboring ones. By using this continuity, the overall dynamics of the traffic flow is proven to be convex.

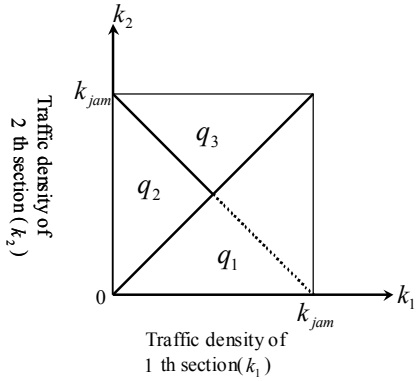



Fig. 13. Assignation of traffic flow mode

Consider Fig.(13), where each mode of traffic flow is assigned. Since the Hessian matrices of $q_1(\frac{x_1}{l_1}, \frac{x_2}{l_2})$, $q_2(\frac{x_1}{l_1}, \frac{x_2}{l_2})$, and $q_3(\frac{x_1}{l_1}, \frac{x_2}{l_2})$ are nonsingular as follows,

$$\begin{aligned} \nabla^2 q_1(\boldsymbol{x}) &= \left[\frac{\partial^2 q_1(\boldsymbol{x})}{\partial x_1 \partial x_2}\right] \\ &= \begin{bmatrix} \frac{v_f}{2x_{jam}} & \frac{v_f}{2x_{jam}} \\ \frac{v_f}{2x_{jam}} & \frac{v_f}{2x_{jam}} \end{bmatrix} \geq 0, \end{aligned} \tag{72}$$

$$\nabla^2 q_2(\boldsymbol{x}) = \left[\frac{\partial^2 q_2(\boldsymbol{x})}{\partial x_1 \partial x_2}\right] = \begin{bmatrix} \frac{v_f}{2x_{jam}} & 0 \\ 0 & 0 \end{bmatrix} \geq 0, \tag{73}$$

and

$$\begin{aligned} \nabla^2 q_3(\boldsymbol{x}) &= \left[\frac{\partial^2 q_3(\boldsymbol{x})}{\partial x_1 \partial x_2}\right] \\ &= \begin{bmatrix} 0 & 0 \\ 0 & \frac{v_f}{2x_{jam}} \end{bmatrix} \geq 0, \end{aligned} \tag{74}$$

they are convex at each mode.
In order to show the convexity of the overall dynamics of the traffic flow, we use following lemma :

**Lemma 1** The neighboring two closed convex dynamics $D_1(\Psi=(\psi_1, \psi_2, \cdots, \psi_n))$ and $D_2(\Psi)$ are convex if they are continuous at the boundary point $(\hat{\psi_1}, \hat{\psi_2}, \cdots, \hat{\psi_n}) \in \Theta$ $(\Theta = \overline{D_1(\Psi)} \bigcap \overline{D_2(\Psi) \backslash D_1(\Psi)})$ and satisfy that

if for $\forall\, i$, $\gamma$ and $\mu$

$$\nabla_\gamma D_1(\Psi)\Big|_{\psi_i=\hat{\psi_i}} \leq (\geq) \nabla_\gamma D_2(\Psi)\Big|_{\psi_i=\hat{\psi_i}} \tag{75}$$

then

$$\nabla^2_{\mu,\mu} D_1(\Psi)\Big|_{\psi_i=\hat{\psi}_i} \leq (\geq) \nabla^2_{\mu,\mu} D_2(\Psi)\Big|_{\psi_i=\hat{\psi}_i} \tag{76}$$

where overline denote the closure of the set, $1 \leq i, \gamma, \mu \leq n$, $\nabla_\gamma D$ is the $\gamma$th element of $\nabla D$, and $\nabla^2_{\mu,\mu} D$ is the $(\mu, \mu)$th element of the matrix $\nabla^2 D$.

The continuity at the boundary is easily confirmed by letting $k_1(\tau) = k_2(\tau) = k(\tau)$ as follows,

$$\begin{aligned} q_1(k_1(\tau), k_2(\tau)) &= q_2(k_1(\tau), k_2(\tau)) && (77)\\ &= q_3(k_1(\tau), k_2(\tau)) && (78)\\ &= k(\tau) v_f \Big(1 - \frac{k(\tau)}{k_{jam}}\Big). && (79) \end{aligned}$$

Lastly, with following eqs. (80) to (83),

$$\begin{aligned} \nabla q_1(\boldsymbol{x})\Big|_{x_1=\hat{x_1}} &= \left[\frac{v_f}{k_j}k - \frac{v_f}{2}, \frac{v_f}{k_j}k - \frac{v_f}{2}\right] && (80)\\ \nabla^2 q_1(\boldsymbol{x})\Big|_{x_1=\hat{x_1}} &= \begin{bmatrix} \frac{v_f}{2x_{jam}} & \frac{v_f}{2x_{jam}} \\ \frac{v_f}{2x_{jam}} & \frac{v_f}{2x_{jam}} \end{bmatrix} && (81)\\ \nabla q_2(\boldsymbol{x})\Big|_{x_1=\hat{x_1}} &= \left[2v_f \frac{k}{k_{jam}} - v_f, 0\right] && (82)\\ \nabla^2 q_2(\boldsymbol{x})\Big|_{x_1=\hat{x_1}} &= \begin{bmatrix} \frac{v_f}{2x_{jam}} & 0 \\ 0 & 0 \end{bmatrix}, && (83) \end{aligned}$$

the convexity condition of lemma 1 was satisfied, since

$$\nabla_1 q_1(x) \leq \nabla_1 q_2(x) \tag{84}$$

in pair with

$$\nabla^2_1 q_1(x) \leq \nabla^2_1 q_2(x). \tag{85}$$

In the same way,

$$\nabla_1 q_2(x) \leq \nabla_1 q_3(x) \quad , \quad \nabla_1 q_1(x) \leq \nabla_1 q_3(x) \tag{86}$$

are satisfied , paired together with

$$\nabla^2_1 q_2(x) \leq \nabla^2_1 q_3(x) \quad , \quad \nabla^2_1 q_1(x) \leq \nabla^2_1 q_3(x). \tag{87}$$

Therefore, the convexity of overall dynamics are confirmed.
Note that although $z$ is the multiplication of $q$ and $u$, the performance criteria (70) is a convex function. This is because $u$ is the vector whose elements $u_i \in \{0, 1\}$ are binary variables, if $u_i = 1$, $z_i$ remains as it stands now, otherwise the term $z_i$ is dropped off from the performance

criterion. And (71) is also a convex function, since $|x_1 - x_2|$ can be transformed to $(e_x^+ + e_x^-)$, minimizing $e_x^+ + e_x^-$ with the conditions of

$$e_x^+ \geq 0 \tag{88}$$
$$e_x^- \geq 0 \tag{89}$$
$$e_x^+ - e_x^- = x_1 - x_2 \tag{90}$$

where $e_x^+$ and $e_x^-$ are equivalently

$$e_x^+ = \frac{(x_1 - x_2) + |x_1 - x_2|}{2} \tag{91}$$
$$e_x^- = \frac{-(x_1 - x_2) + |x_1 - x_2|}{2}. \tag{92}$$

Since all the constraints are described in the form of Eq.(30), the problems (70) and (71) are included in the class of the convex programming problem.

### 4.2 Convex Programming

The efficient method such as Penalty Method (PM) can be easily applied to the convex programming problem with performance scheme as follows,

$$\text{minimize} \qquad F(\boldsymbol{x}, r) = f(\boldsymbol{x}) + r\boldsymbol{P}(\boldsymbol{x}) \tag{93}$$
$$\boldsymbol{P}(\boldsymbol{x}) \qquad \begin{cases} = 0, & \boldsymbol{x} \in X \\ > 0, & \boldsymbol{x} \notin X \end{cases} \tag{94}$$

where $f(\boldsymbol{x})$ is the convex performance criterion of the original problem, $r(> 0)$ is the cost coefficient which increases as iteration $l$ increases, $X$ is the convex set, and $P$ is the continuous penalty function satisfying Eq.(94).
This function can be constructed as follows.

**Step 1** Describe the solution space in the following form: $\boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{W}$.

**Step 2** Define active constraints as the set of constraints which fulfill $\boldsymbol{G_i}\boldsymbol{x} = \boldsymbol{W_i}$, and inactive constraints as the set which fulfills $\boldsymbol{G_i}\boldsymbol{x} < \boldsymbol{W_i}$. Here, $\boldsymbol{G_i}$ and $\boldsymbol{W_i}$ are the $i$th raw of the matrix $\boldsymbol{G}$ and $\boldsymbol{W}$, respectively. The active set $\Sigma(\boldsymbol{x})$ is the set of indices of the active constraints, that is, $\Sigma(\boldsymbol{x}) = \{i \in \{1, \cdots, q\} | \boldsymbol{G_i}\boldsymbol{x} = \boldsymbol{W_i}\}$.

**Step 3** Define $p_i$ as follows :

$$p_i(\boldsymbol{x}) = \boldsymbol{N_i}\boldsymbol{x} + \boldsymbol{e} \tag{95}$$

where $\boldsymbol{x} \in \Re^n$, $\boldsymbol{N} \in \Re^n$, $|\boldsymbol{N}| = 1$ is the unit normal vector to the line $\boldsymbol{G_i}\boldsymbol{x} - \boldsymbol{W_i} = 0$, and $\boldsymbol{e} \in \Re^n$ is the vector which describes parallel translation from the origin. Note that $\boldsymbol{N}$ takes outward direction from the convex sets defined by the active constraints, that is $\boldsymbol{N}\boldsymbol{x} + \boldsymbol{e} \leq 0$ for the feasible solution $\boldsymbol{x}$.

Step 4 Obtain the distance $d_i$ as follows,

$$\textit{if} \quad \boldsymbol{G_i}\boldsymbol{x} - \boldsymbol{W_i} \leq 0, \textit{then} \quad d_i(\boldsymbol{x}) = 0$$
$$\textit{otherwise} \quad d_i(\boldsymbol{x}) = |p_i(\boldsymbol{x})|$$

Therefore the condition (94) can be translated as follows,

$$\boldsymbol{P}(\boldsymbol{x}) \quad = \quad \begin{cases} \textit{if } (\mathrm{Y} = \phi) & \boldsymbol{P} = 0 \\ \textit{otherwise} & \boldsymbol{P} = min[\mathrm{Y}] \end{cases} \tag{96}$$

where Y is the set of $d_i(x)$, ($i \in \Sigma$) which is not equal to 0.
The penalty algorithm is implemented as follows,

**Step 1** Select initial point $\boldsymbol{x}^{l_O}$ $(= \boldsymbol{x}^{l_I})$ and $r^1$, and set $l_I \equiv 1$ and $l_O \equiv 1$.

**Step 2** If $r^{l_O} P(\boldsymbol{x}^{l_O}) < \epsilon$, terminate the algorithm. Otherwise, set $r^{l_O+1} \equiv cr^{l_O}$, $l_O \equiv l_O + 1$ and $\boldsymbol{x}^{l_I} \equiv \boldsymbol{x}^{l_O}$.

**Step 3** If $||\nabla f(\boldsymbol{x})|| < \epsilon$, set $x^{l_O} \equiv x^{l_I}$ and go to Step 2. Otherwise, go to Step 4.

**Step 4** Find the steepest descent direction, $\boldsymbol{d}$ $(= -\nabla^T f(\boldsymbol{x}^{l_I}))$.

**Step 5** Find the step width $\alpha^{l_I}$, do $\boldsymbol{x}^{l_I+1} = \boldsymbol{x}^{l_I} + \alpha^{l_I}\boldsymbol{d}^{l_I}$, and set $l_I \equiv l_I + 1$. And go to Step 3.

Here, $\epsilon$ is small tolerance, and $c$ and $\alpha^{l_I}$ are heuristically obtained.
If we can select the feasible initial solution, the optimal solution would be found in a short time. In this paper, the existence of solution is verified as follows.

**Lemma 2** The range of $x_i(\kappa)$ where $1 \leq i \leq m$ is $0 \leq x_i(\kappa) \leq l_i k_{jam}$. If $x_i(\kappa+1)$ always exists within the range for all $i$ in the case of $0 \leq x_i(\kappa) \leq l_i k_{jam}$ for all $i$, the feasible solution $\boldsymbol{x}(\kappa+1)$ can be found.

**Proof :** Consider the following equation :

$$\begin{aligned} & l_i k_i(\kappa+1) - l_i k_i(\tau) \\ = \quad & -\boldsymbol{q}(k_{i-1}(\kappa), k_i(\kappa))T_s \\ & +\boldsymbol{q}(k_i(\kappa), k_{i+1}(\kappa))T_s. \end{aligned} \tag{97}$$

It is obvious that $x_i$ is within the range if and only if

$$l_i k_i(\tau) \quad \geq \quad -\boldsymbol{q}(k_i(\tau), k_{i+1}(\tau))T_s \tag{98}$$

$$l_i k_{jam} - l_i k_i(\tau) \quad \leq \quad \boldsymbol{q}(k_{i-1}(\tau), k_i(\tau))T_s. \tag{99}$$

By substituting $q$ of (98) to (14), following inequality is obtained from the both $k_i(\kappa) \geq k_{i+1}(\kappa)$ and $k_i(\kappa) < k_{i+1}(\kappa)$.

$$1 \quad \geq \quad \frac{v_f}{l_i}\left(1 - \frac{k_i}{k_{jam}}\right)T_s. \tag{100}$$

Since $\frac{v_f T_s}{l_i} \ll 1$, (98) can be easily confirmed. In the similar way, the condition (99) can be easily confirmed.

## 5. 0-1 Classification based on PWARX System

The MINLP based traffic network controller introduced in the previous chapter is generally known to require large computational effort. In this chapter, we propose a new controller design method for hybrid systems with a binary output. The proposed method develops a classification map of the modified PWARX system, which relates a binary output and all observational variables including past inputs and outputs. The output $y(\kappa)$ (which corresponds to the plant input $u_p(\kappa)$) is obtained by finding the corresponding cluster among the classification map, while in the conventional methods, the MINLP problems were solved at every sampling instant.

Figure describes a block diagram of the proposed controller design method, where the MINLP controller is constructed to control the traffic flow in each traffic intersection in a decentralized manner. The traffic inflows from the outside and outflow to the outside are closely affected by the traffic flows at the adjoining traffic intersections. In order to construct the classification map, we need history of inputs and output of the MINLP controller obtained by applying it to various situations of the network.

### 5.1 Classification problem of hybrid dynamics

The PWARX (Piece-Wise Auto Regressive eXogeneous) system is a well-formulated classification technique for a hybrid and nonlinear dynamics. The PWARX system contains the state vector $\boldsymbol{x}$ which consists of past inputs and past outputs of the system as follows

$$\boldsymbol{x}(\kappa) = [y'(\kappa-1), y'(\kappa-2), \cdots, y'(\kappa-n_a), \tag{101}$$

$$u'(\kappa-1), u'(\kappa-2), \cdots, u'(\kappa-n_b)]$$

and this vector is involved in one of the polyhedral convex regions defined by

$$\chi_i = \{\boldsymbol{x} | \boldsymbol{V}_i \boldsymbol{x}(\kappa) \leq \boldsymbol{W}_i\}. \tag{102}$$

The entire behavior of the state vector is represented in a piece-wise manner. The dynamics of each region is defined as follows

$$f_i(\boldsymbol{x}(\kappa)) = \boldsymbol{\theta}_i \boldsymbol{\rho}(\kappa) \tag{103}$$

where $\boldsymbol{\rho}(\kappa)$ is $[\boldsymbol{x}(\kappa), 1]'$, and $\boldsymbol{\theta}$ is the coefficient vector as follows.

$$\boldsymbol{\theta}_i = [a_{i,1}, \cdots, a_{i,n_a}, b'_{i,1}, \cdots, b'_{i,n_b}, f_i]' \tag{104}$$

The problem we address in this paper is a special classification problem where the output $y$ is a 0-1 binary variable, and very good classification performance is desirable even with very large number of the introduced clusters. If we plot the observational data in a pure (not mixed) cluster in the $\boldsymbol{x}$-$y(k)$ space, it will show always zero inclination, since we have a binary output, i.e., all the components of $\theta$, $a$ and $b$ expect for $f$ will be zeros.

For this type of clustering problem, the conventional PWARX system does not well reproduce the 0-1 output. Since it simultaneously obtains clusters and its (linear) dynamics applying the least squared method to each of the fixed number of clusters, the overall accuracy of their reproduced model is not so high. Furthermore they are very sensitive to the initialization concerning the number of clusters, center of initial clusters, and so on.

### 5.2 Classification based on PWARX system

The identification procedure of the hybrid dynamics using the PWARX system is described as follows.

Step 1 Set the number of clusters, $s$, centers of $s$ clusters, $\mu$, and the threshold value $\epsilon > 0$.

Step 2 Obtain the cluster $D_i$ of $\xi$ points which minimize the following performance criterion

$$J = \sum_{i=1}^{s} \sum_{\xi_j \in D_i} ||\xi_j - \mu_i||^2_{R_j^{-1}} \tag{105}$$

Step 3 Update the centers $\mu$ according to the following formula.

$$\tilde{\mu}_i = \frac{\sum_{j:\xi_j \in D_i} \xi_j w_j}{\sum_{j:\xi_j \in D_i} w_j} \tag{106}$$

If $max(||\tilde{\mu}_i - \mu_i||) < \epsilon$, exit, else set

$$\mu = \tilde{\mu}_i \tag{107}$$

and go to Step 2.

In Step 2, $R_j$ is defined as

$$R_j = \begin{bmatrix} V_j & 0 \\ 0 & Q_j \end{bmatrix} \tag{108}$$

where

$$V_j = \frac{S_j}{c - (n_a + n_b) + 1} (\Phi_j' \Phi_j)^{-1} \tag{109}$$

$$Q_j = \sum_{(x,y) \in C_j} (x - m_j)(x - m_j)' \tag{110}$$

$$\Phi_j = \begin{bmatrix} x_1 & x_2 & \cdots & x_c \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{111}$$

$$S_j = y'_{c_j} (I - \Phi_j (\Phi_j' \Phi_j)^{-1} \Phi_j') y_{c_j} \tag{112}$$

$$m_j = \frac{1}{c} \sum_{(x,y) \in C_j} \boldsymbol{x}, j = 1, \cdots, N \tag{113}$$

$$\xi_j = [(\theta_j)', m_j'] \tag{114}$$

$$w_j = \frac{1}{\sqrt{(2\pi)^{(2n_a + 2n_b + 1)} \det(R_i)}}. \tag{115}$$

$V_j$ is the empirical covariance matrix which measures the relevance criterion, $Q_j$ is the scatter matrix which measures the sparsity of data in the cluster $j$, $S_j$ is the sum of squared residuals, $C_j$ is the cluster in the $\boldsymbol{x}$ space, $x_j$ is the regressor vector belonging to $C_j$, $y_{c_j}$ is the output vector included in $C_j$.

The main difference of this method from the conventional K-means method is that based on the confidence level $w_j$, the proposed method assigns the vectors $\xi$ to the cluster $D_i$ in the parameter vector $\theta$-$\boldsymbol{x}$ space, while K-means assigns the data to the cluster $C_i$ in the state vector $\boldsymbol{x}$ space. This property serves for identification of $y$ that mixed clusters are suppressed being referred to the dynamics of $y$.

### 5.3 0-1 classification based on modified PWARX system

The desired outputs are continued by the same values, 0 or 1 in the $x$-$y$ space. All values except for the offset variable $f$ among parameters of $\theta$ will be zeros, i.e., the dynamics in $\theta$-$\boldsymbol{x}$ space will be almost same. Therefore in the conventional PWARX system, the regions with same dynamics were often considered to be included in the same cluster.
The proposed method described below is a hierarchical PWARX system for a 0-1 classification as follows.

Step 1 (*Initialization Process*) Set the number of clusters, $s$, the number of the splitting clusters, $s_r$, the cluster centers, $\mu_i$ ($i \in [1, s]$), the initial data group number $N$, the renew data group number $N'$ and the threshold values $\epsilon > 0$ and $\gamma > 0$. Using $K$-means, obtain small $N$ data groups so that neighboring data may be belonged to the same groups.

Step 2 (*Piecewise Fitting Process*) Obtain the cluster $D_i$ of $\xi$ points which minimizes the following performance criterion.

$$J_\chi = \sum_{i=1}^{s} \sum_{\xi_j \in D_i} ||\xi_j - \mu_i||^2_{R_j^{-1}} \tag{116}$$

Obtain the guard $\boldsymbol{V}_i$ and $\boldsymbol{W}_i$ by solving a quadratic problem for all $i$ and $i'$ which satisfy $1 \leq i \leq s$ and $1 \leq i' \leq s$ ($i \neq i'$) as follows.

$$\text{find} \qquad \boldsymbol{V}_{i,i'} \quad \text{and} \quad \boldsymbol{W}_{i,i'} \tag{117}$$

$$\text{minimize} \qquad \boldsymbol{V}_{i,i'} \boldsymbol{V}_{i,i'}^T \tag{118}$$

$$\text{subject to} \qquad \zeta_l (\boldsymbol{V}_{i,i'}^T \boldsymbol{x_l} + \boldsymbol{W}_{i,i'}) \geq 1 \tag{119}$$

where $l$ is the data number and $\zeta$ is defined as follows.

$$\zeta_l \quad = \quad \begin{cases} 1 & if \quad \xi(\boldsymbol{x_l}) \in D_i \\ -1 & if \quad \xi(\boldsymbol{x_l}) \in D_{i'} \end{cases} \tag{120}$$

Here $\xi(\boldsymbol{x})$ is the function which obtains the corresponding value of $\xi$ from $\boldsymbol{x}$, i.e., $\xi$ is a translation of $\boldsymbol{x}$ in the $\theta$-$\boldsymbol{x}$ space. Then $\boldsymbol{V}_i$ and $\boldsymbol{W}_i$ are obtained as follows.
$\boldsymbol{V}_i = [\boldsymbol{V}_{i,1}^T, \cdots, \boldsymbol{V}_{i,i-1}^T, \boldsymbol{V}_{i,i+1}^T, \cdots, \boldsymbol{V}_{i,s}^T]^T$ and $\boldsymbol{W}_i = [\boldsymbol{W}_{i,1}, \cdots, \boldsymbol{W}_{i,i-1}, \boldsymbol{W}_{i,i+1}, \cdots, \boldsymbol{W}_{i,s}]^T$.

Step 3 (*Cluster Updating Process*) Update the centers $\mu$ according to the following formula.

$$\tilde{\mu}_i = \frac{\sum_{j:\xi_j \in D_i} \xi_j w_j}{\sum_{j:\xi_j \in D_i} w_j} \tag{121}$$

If $max||\tilde{\mu}_i - \mu_i||) < \epsilon$, go to Step 4, otherwise set

$$\mu = \tilde{\mu}_i \tag{122}$$

and go to Step 2.

Step 4 (*Cluster Splitting Process*) Obtain $J_i$ for all $i \in [1, s]$ which is defined by

$$J_i = \sigma^2(y(\kappa)). \tag{123}$$

Step 4-1 For all $i \in [1, s]$, do the following. If $J_i \leq \gamma$, do the following

$$\chi = \chi - \chi_i \tag{124}$$

$$\chi_i = \{\boldsymbol{x} | \boldsymbol{V}_i \boldsymbol{x} \leq \boldsymbol{W}_i\} \tag{125}$$

otherwise set new centers of the $s_r$ clusters, $\mu_r$ in $D_i$ randomly, and do the following.

$$s = s + s_r \tag{126}$$

Here, $\sigma^2(y(\kappa))$ is the covariance of $y(\kappa)$ in the cluster $D_i$.

Step 4-2 Set $i_m$ as follows.

$$i_m = \arg \min_{i \in [1,s]} \sigma^2(y(\kappa)) \tag{127}$$

Step 4-3 If $J_{i_m} \leq \gamma$, terminate with success, otherwise, obtain $N'$ data group of the corresponding region of $D_{i_m}$ and go to Step 2.

Note that in Step 2, the maximum margin of the data point $\boldsymbol{x}$ from the hyper-plane $\boldsymbol{V}_{i,i'}\boldsymbol{x} + \boldsymbol{W}_{i,i'} \leq 0$ is proportional to $(\boldsymbol{V}_{i,i'}^T \boldsymbol{V}_{i,i'})^{-1}$ since letting the hyperplane which maximize the margin $\alpha$ from the data points $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$ as follows

$$\boldsymbol{V}\boldsymbol{x}^+ + \boldsymbol{W} = \alpha \tag{128}$$

and

$$\boldsymbol{V}\boldsymbol{x}^- + \boldsymbol{W} = -\alpha, \tag{129}$$

the maximal margin $\alpha_{MAX}$ is defined as follows

$$\alpha_{MAX} = \frac{1}{2}\Big(\frac{\boldsymbol{V}}{||\boldsymbol{V}||_2}\boldsymbol{x}^+ - \frac{\boldsymbol{V}}{||\boldsymbol{V}||_2}\boldsymbol{x}^-\Big) \tag{130}$$

$$= \frac{1}{2||\boldsymbol{V}||_2}(\boldsymbol{V}\boldsymbol{x}^+ - \boldsymbol{V}\boldsymbol{x}^-) \tag{131}$$

$$= \frac{\alpha}{||\boldsymbol{V}||_2} \tag{132}$$

Therefore by minimizing $(\boldsymbol{V}_{i,i'}^T \boldsymbol{V}_{i,i'})$, the margin can be maximized.

## 6. Numerical Experiments

### 6.1 Numerical Environments

In order to show the usefulness of our proposed method, we show, in this section, some results of the numerical experiments. We considered the traffic network of Fig. 14, where the square network with $1000 \times 1000$ [$m^2$] consists of 16 intersections and 112 districts, all with 2 lanes bi-directionally. Four controllers are applied to find optimal traffic light for the overall network. Each controller is assigned to the network with $500 \times 500$ [$m^2$]. We assume that from the outside of the network traffic flows of vehicles move into the network with random speeds, whereas the traffic flows inside the network, move from the network with the speed of infinity (no congestion arises and affects the traffic flow inside the network). The variables used in this paper are as follows; $\boldsymbol{x} \in \Re^{56}$, $\boldsymbol{q} \in \Re^{80}$, $\boldsymbol{\delta} \in \{0, 1\}^4$. We used (70) as a performance criterion. All results are obtained from simulations over 30 minutes, where the sampling interval $T_s$ is 10 [sec].

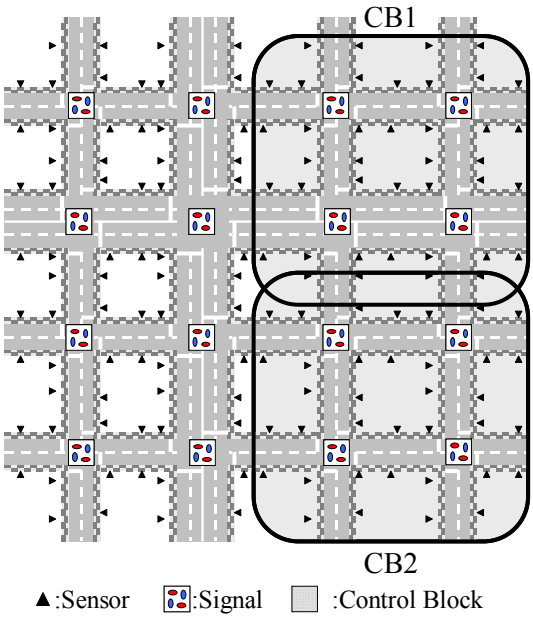



Fig. 14. Traffic network

| | No Control | $H = 1$ | $H = 2$ |
|---|---|---|---|
| A | 2724 | 2884 | 2913 |
| B | - | 3.1 | 370.4 |
| C | - | 1.2 | 14.6 |

Table 1. Numerical experimental result WRT $H$

### 6.2 Traffic Flow Control System for Traffic Network

We show the results obtained by applying our proposed methods in Table 1, where $H$ denotes the length of the prediction horizon, 'No Control' implies that the traffic light is changed at every 30 second, and

A: Number of cars passing through the boundary of every two consecutive districts,

B: Average computation time,

C: Average number of the generated sub-problems.

From the results in Table 1, we find that although the MPC with longer prediction horizon enables more vehicles to pass through the traffic network, the difference between the cases of $H = 1$ and $H = 2$ is not so remarkable. This implies that the proposed method can be applied to find semi-optimal solution for the real traffic control system with a proper selection of prediction horizon length.

| Length of $H$ | Proposed Method | | | Method of (1) | | |
|---|---|---|---|---|---|---|
| | A | B | C | A | B | C |
| 1 | 616 | 0.02 | 4 | 616 | 14.98 | 244 |
| 2 | 724 | 1.34 | 8 | 718 | 265.18 | 488 |
| 3 | 869 | 129.20 | 12 | 870 | 2688.6 | 732 |

Table 2. Comparison of the computational efforts

| | No Control | $H = 1$ | $H = 2$ |
|---|---|---|---|
| A | 5249 | 5660 | 5717 |
| B | - | 3.1 | 370.4 |
| C | - | 1.2 | 14.6 |

Table 3. Experimental result in case of no arterial road

### 6.3 Comparison of computational amount

In order to evaluate the computational amount of the presented method, we compare in Table 2 computational times obtained by applying our method and conventional method (1). We used Athlon XP 2400 and Windows 2000 for this experiments. Note that our method finds better solution with a shorter time. This is because the presented method does not approximate nonlinear dynamics(1) and solves non-linear programming problem, reformulating it to the convex programming problem. Furthermore, the presented refining process enables to avoid introduction of an enormous number of auxiliary variables.

### 6.4 TFCS for large-scale traffic network

In this subsection, the effectiveness of our method for large-scale traffic network control with the arterial roads is shown. If the traffic light controller is applied to the large-scale traffic network in a centralized manner, the computational amount would be fairly enormous. The presented method, as in Fig.14, designates the control block which groups some traffic lights in order that the feasible solutions may be obtained during the sampling interval. Fig. 14 illustrates that four control blocks (CB) constitute the entire traffic network where the sensory information at each boundary of CBs is shared for the control of both blocks. Note that two arterial roads are running north-south (second road from the left) and east-west (second road from the top), respectively. Table 3 and Table 4 show the obtained solutions by applying the presented method both in the case that there is no arterial roads and in the case that there are 2 arterial roads. In both numerical experiments, traffic densities at each road were set to exactly same value. The results in both cases show that the presented method has good solution in both cases. Note that our method has always better or equal solutions, compared with the cases of 'No Control'.

## 7. Classification Results

### 7.1 MINLP controller coupled with Model Predictive Control

The 5000 data sets obtained in the previous chapter are classified based on the 0-1 classification method. For this simulation, we set the number of initial clusters, $s$, to be 100 and whenever

| | No Control | $H = 1$ | $H = 2$ |
|---|---|---|---|
| A | 6060 | 6980 | 7185 |
| B | - | 3.6 | 250.4 |
| C | - | 1.3 | 10.4 |

Table 4. Experimental result in case of 2 arterial roads

| Step | Red | Blue | Mixed | Total |
|---|---|---|---|---|
| 1 | 7 | 5 | 38 | 50 |
| 2 | 27 | 29 | 32 | 88 |
| 3 | 48 | 46 | 26 | 120 |
| 4 | 64 | 63 | 19 | 146 |
| 5 | 72 | 75 | 18 | 165 |
| 6 | 87 | 81 | 15 | 183 |
| 7 | 95 | 92 | 11 | 198 |
| 8 | 102 | 98 | 9 | 209 |
| 9 | 110 | 103 | 5 | 218 |
| 10 | 114 | 107 | 2 | 223 |
| 11 | 116 | 109 | 0 | 225 |

Table 5. Stepwise Cluster Number (H=1)

we split the polyhedron defined by the guard $\boldsymbol{V}$ and $\boldsymbol{W}$ in the cluster splitting process, we split into two ($s_r$ =2).
We show the classification results in TABLE 5 - 8. In TABLE 5 and 7, "Red" and "Blue" imply the traffic signals of the clusters that if a data set is included in this cluster, the control input $u$ will represent this colors, while "Mixed" implies the clusters are not fully classified that Red and Blue signals are mixed in the cluster. The numbers of data in "Red", "Blue" and "Mixed" are shown in TABLE 6 and 8.
While the data shown in TABLE 5 and 6 are obtained by applying the MPC horizon $H$ = 1, the data shown in TABLE 7 and 8 are obtained by applying the MPC horizon $H$ = 3, respectively.

### 7.2 Comparison with conventional PWARX system

The conventional PWARX system is compared with our presented method. TABLE 9 and 10 compare the cluster number and the data number in the clusters. In TABLE 9 and 10, the conventional method is applied with the initial cluster number of 100, 200, 300, 400 and 500 respectively. Although most of data were well classified introducing a large number of clusters, 2.8 and 1.6 percents of the total data were not correctly classified. In contrast the presented method perfectly classified introducing relatively a small number of clusters.

## 8. Concluding remarks

In this paper we have presented a new design method for a traffic network hybrid feedback controller. Since the output of the traffic network controller is 0-1 binary signals, the output of the developed controller has been reproduced through 0-1 classifications of the PWARX systems. The developed PWARX classifier describes nonlinear feedback control laws of traffic

| Step | Red | Blue | Mixed |
|---|---|---|---|
| 1 | 188 | 238 | 2072 |
| 2 | 447 | 646 | 1405 |
| 3 | 670 | 801 | 1027 |
| 4 | 862 | 1013 | 623 |
| 5 | 952 | 1066 | 480 |
| 6 | 1063 | 1132 | 303 |
| 7 | 1110 | 1238 | 150 |
| 8 | 1131 | 1273 | 94 |
| 9 | 1149 | 1304 | 45 |
| 10 | 1169 | 1317 | 12 |
| 11 | 1171 | 1327 | 0 |

Table 6. Stepwise Data Number in the cluster(H=1)

| Step | Red | Blue | Mixed | Total |
|---|---|---|---|---|
| 1 | 5 | 8 | 37 | 50 |
| 2 | 23 | 27 | 37 | 87 |
| 3 | 45 | 46 | 33 | 124 |
| 4 | 66 | 66 | 25 | 157 |
| 5 | 81 | 82 | 19 | 182 |
| 6 | 92 | 93 | 16 | 201 |
| 7 | 105 | 105 | 7 | 217 |
| 8 | 112 | 108 | 4 | 224 |
| 9 | 116 | 112 | 0 | 228 |

Table 7. Stepwise Cluster Number (H=3)

| Step | Red | Blue | Mixed |
|---|---|---|---|
| 1 | 90 | 266 | 2142 |
| 2 | 356 | 570 | 1572 |
| 3 | 614 | 803 | 1081 |
| 4 | 794 | 999 | 705 |
| 5 | 953 | 1105 | 440 |
| 6 | 1010 | 1212 | 276 |
| 7 | 1058 | 1276 | 164 |
| 8 | 1138 | 1299 | 61 |
| 9 | 1175 | 1323 | 0 |

Table 8. Stepwise Data Number in the cluster(H=3)

| | Total | Red | Blue | Mixed |
|---|---|---|---|---|
| Proposed | 225 | 116 | 109 | 0 |
| Conventional | 100 | 30 | 32 | 38 |
| | 200 | 67 | 96 | 37 |
| | 300 | 127 | 135 | 38 |
| | 400 | 185 | 183 | 32 |
| | 500 | 228 | 255 | 17 |

Table 9. Comparison of cluster number (H=1)

| | Cluster Number | Data Number of Red Clusters | Blue | Mixed |
|---|---|---|---|---|
| Proposed | 225 | 1171 | 1327 | 0 |
| Conventional | 100 | 614 | 853 | 1031 |
| | 200 | 926 | 1113 | 459 |
| | 300 | 984 | 1180 | 334 |
| | 400 | 1014 | 1250 | 234 |
| | 500 | 1095 | 1263 | 140 |

Table 10. Comparison of data number in the cluster (H=1)

| | Total | Red | Blue | Mixed |
|---|---|---|---|---|
| Proposed | 228 | 116 | 112 | 0 |
| Conventional | 100 | 32 | 29 | 39 |
| | 200 | 78 | 83 | 39 |
| | 300 | 136 | 136 | 28 |
| | 400 | 186 | 184 | 30 |
| | 500 | 240 | 246 | 14 |

Table 11. Comparison of cluster number (H=3)

| | Cluster Number | Data Number of Red Clusters | Blue | Mixed |
|---|---|---|---|---|
| Proposed | 228 | 1175 | 1323 | 0 |
| Conventional | 100 | 609 | 715 | 1174 |
| | 200 | 830 | 1089 | 579 |
| | 300 | 1040 | 1245 | 213 |
| | 400 | 1047 | 1227 | 224 |
| | 500 | 1148 | 1266 | 84 |

Table 12. Comparison of data number in the cluster (H=3)

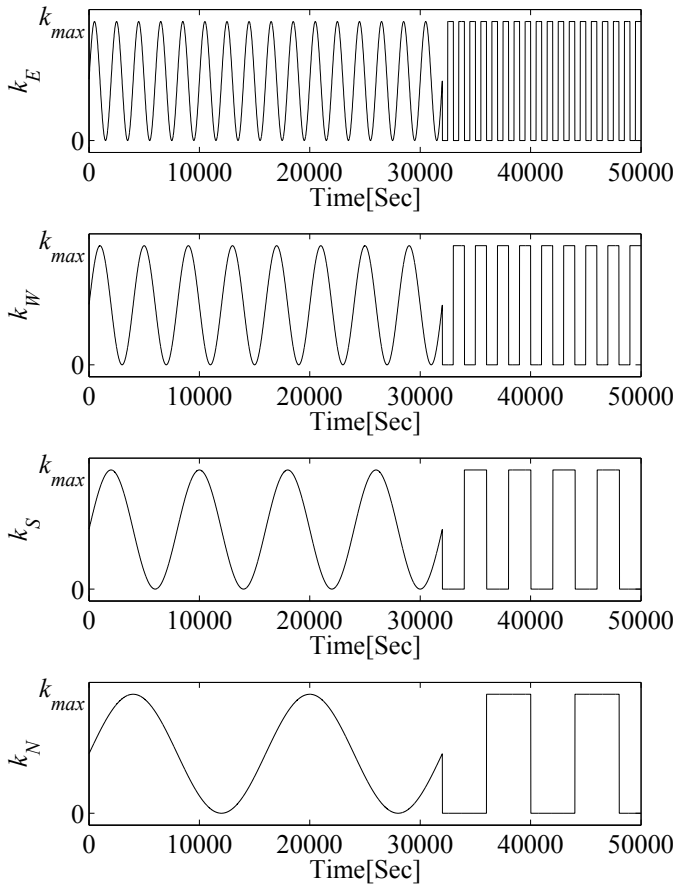



Fig. 15. Density of traffic flow

control systems. As we checked in chapter VII, very good solutions are obtained in a very short time, compared with the one obtained with the conventional MINLP controller.
In a classification problem considered in this paper, very good classification performance is required even with very large number of the introduced clusters. In our PWARX system formulation, we have adopted a new performance criterion related with the covariance of the control output. If a well-classified cluster is found, the cluster is separated from the classification map. If a bad-classified mixed cluster is found, the cluster is split into smaller $s_r$ pieces, and at the next iteration, this cluster is reclassified. The developed classification method has been applied to a traffic network control system, successfully reproducing the output of the conventional MINLP controller.

[Matrices in MLDS]

$$\boldsymbol{A} = \boldsymbol{I} \tag{133}$$

$$\boldsymbol{B} = \begin{bmatrix} -1 & 1 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & -1 & 1 & 0 & \ddots & \ddots & \vdots \\ 0 & -1 & 0 & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \tag{134}$$

$$\boldsymbol{E_1} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \boldsymbol{I} & -\boldsymbol{I} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Gamma_1} \end{bmatrix}^T \tag{135}$$

$$\boldsymbol{E_2} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\boldsymbol{I} & \boldsymbol{I} & \mathbf{0} \end{bmatrix}^T \tag{136}$$

$$\boldsymbol{E_4} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}^T \tag{137}$$

$$\boldsymbol{E_5} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \boldsymbol{\Lambda_x} & \mathbf{0} & \mathbf{0} & \boldsymbol{\Lambda_y} & \boldsymbol{\Gamma_5} \end{bmatrix}^T \tag{138}$$

where

$$\boldsymbol{\Lambda_x} = \begin{bmatrix} -x_{max} & -x_{max} & \cdots & -x_{max} \end{bmatrix} \tag{139}$$

$$\boldsymbol{\Lambda_z} = \begin{bmatrix} z_{max} & z_{max} & \cdots & z_{max} \end{bmatrix} \tag{140}$$

$$\boldsymbol{\Gamma_1} = \begin{bmatrix} \boldsymbol{\gamma_1} & 0 & \cdots & \cdots & 0 \\ 0 & \boldsymbol{\gamma_1} & 0 & \ddots & 0 \\ \vdots & 0 & \boldsymbol{\gamma_1} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \boldsymbol{\gamma_1} \end{bmatrix} \tag{141}$$

$$\boldsymbol{\Gamma_5} = \begin{bmatrix} \boldsymbol{\gamma_5} & \boldsymbol{\gamma_5} & \cdots & \boldsymbol{\gamma_5} \end{bmatrix}^T \tag{142}$$

$$\boldsymbol{\gamma_1} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix} \tag{143}$$

$$\boldsymbol{\gamma_5} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \tag{144}$$

$$\boldsymbol{u} = \begin{bmatrix} u_{E,1} \\ u_{W,1} \\ u_{N,1} \\ u_{S,1} \\ u_{E,2} \\ u_{W,2} \\ u_{N,2} \\ u_{S,2} \\ \vdots \\ u_{S,m} \end{bmatrix} \tag{145}$$

## 9. References

[1] Tatsuya Kato and YoungWoo Kim and Shigeru Okuma: Model Predictive Control of Traffic Flow Based on Hybrid System Modeling, ICCAS, pp.368-373 (2005)
[2] Hayakawa Hisao: Considering the traffic flow as the pulverulent body (in Japanese), Parity, Vol.13, No.5, pp.13–22 (1998)
[3] Yasuyoshi Kato: Traffic Flow Simulation by Cellular Automaton Method, JSIE, Vol.15, No.2, pp.242–250 (2000)
[4] K. Nagel and M. Schreckenberg. A: Cellular Automaton Model for Freeway Traffic, Journal de Physique I France, pp.2–2221 (1992)
[5] H. J. Payne: Models of freeway traffic and control in Mathematical Models of Public Systems, Simulation Council Proceedings Series, LaJolla, California,Vol.1, No.1, pp.51-61 (1971)
[6] M.J.Lighthill and G.B.Whitham: On kinematic waves II. A theory of traffic flow on long crowded roads, Proc. R. Soc. London Ser. A, Vol.229, pp.281 (1955)
[7] I. Prigogine and R. Herman: Kinetic Theory of Vehicular Traffic, Elsevier, New York (1971)
[8] M. Cremer and J. Ludwig: A fast simulation model for traffic flow on the basis of Boolean operations, Mathematics and Computers in Simulation, Vol.28 (1986, pp.297–303)
[9] Balduzzi, F.and Giua, A. and Menga, G: First-order hybrid Petri nets: a model for optimization and control., IEEE Trans. on Robotics and Automation, Vol.16, No.4, pp.382-399 (2000)
[10] Richard Haberman: Mathematical Models, Prentice-Hall (1977)
[11] Chaudahuri, P.P and others: Additive Cellular Automata -Theory and Applications, IEEE Computer Society Press (1997)
[12] A. Bemporad and M. Morari: "Control of systems integrating logic, dynamics, and constraints", Tech. Report AUT98-04, ETH, Automatica, Special issue on hybrid systems, Vol.35, No.3, pp.407-427 (1999)
[13] Camacho, E.F. and Bordons, C.: Model predictive control in the process industry, Springer-Verlag (1995)

# Using Petri Nets in the analysis of sequential automata models with direct applications on the transport systems with accumulation areas

Dan Ungureanu-Anghel
*Dep. of Automation and Applied Informatics,*
*"Politehnica" University of Timisoara*
*Faculty of Automation and Computers*

## 1. Introduction

The development of flexible manufacturing systems (FMS) led implicitly to the revaluation of place and significance of the transport systems. A FMS cannot be considered efficient if the afferent transport system is not itself an efficient one. In these circumstances, the importance of the transport system had risen, from the classical phase, in which this had a role preponderant to transport the pieces from one machine to another, to a phase in which the tasks of the transport system were extended (automatic sorting, the automatic selection of the destination for a piece, computing and choosing the optimal route, etc.), talking now about intelligent transport systems.

Transport systems with accumulation area (TSAA) belongs to the class of intelligent transport systems, being capable, based on some algorithms or on some functioning specifications (static – determined during design, or dynamic – which can be modified during the operation depending on the operator demands) to execute transfers of pieces or other transport elements (trolleys), according to the requirements. TSAA founds applicability in the most diverse places: automated warehouses, flexible fabrication lines, sorting systems, etc.

In specialized literature the field of industrial transport systems was not neglected. However, the majority of the authors approach these fields from the prospect of the analysis of the materials flux and of the allocated transport time (Chiang et.al 2008)(Ge et. al. 2008)(Polic&Jezernik 2008).

Other approaches related to transport systems can be found in the papers that deals with either the road or rail system (Li et al 2008) (Giua & Seatzu 2008).

The approach of industrial transport systems, within this chapter, it is done different from the way of approach existing in literature, once by direct concentration on a certain type of transport system (TSAA), and the second is done not by the analysis of the materials flux, but by the analysis the mode of administration at the lowest level of materials transport. Starting from the base structure of the TSAA it was identified a basic mechanical structure, called node, for which it was elaborated the adequate model of automatic type. After the

validation of this model, following the structural and behaviour analysis, it passed to the obtaining of generic models.
The higher complexity of these systems, make almost impossible the modelling based on differential equations, which led to the TSAA approach as a discrete event systems (DES).
For the modelling of the constitutive elements of TSAA, and of TSAA on the whole, considered as DES, there will be used models of untimed deterministic automatic type. Because of the fact that the structural and behaviour analysis of the larger models of automatic type is difficult, as well as the lack of dedicated tools, it led to the use of one methodology less common, but which has a very high efficiency. The used method of approach is based on the conversion of the models of automatic type in models of untimed labelled Petri Net type, the structural and behaviour analysis being made using the PetriNets Toolbox from Matlab.
In order that the obtained results from the analysis of the model of Petri Net type to be applicable (in fact to validate) to the model of automatic type it is necessary to fulfil the following condition: a model of Petri Net type obtained from a model of automatic type need to have the topology of state machine type. Only if this condition is complete satisfied the results can validate entirely the model of automatic type. This compulsoriness is imposed especially by the construction manner of the models of automatic type, respectively Petri Nets. Models of automatic type are based on the propagation of the model topology, while the models of Petri Net type are based on the propagation of the markings.
Within this chapter, is pursued the creation of a unified procedural and methodological background, for the conjunction between the models of untimed deterministic automatic type and the untimed label Petri Nets type with direct applications on the transport systems with the accumulation areas.
For the beginning, within the chapter is done a presentation of the theoretical aspects of the connections between the models of automatic type and the models of Petri Nets type connected to: theoretical notions concerning the relation between the models of automatic type and the models of Petri Nets type; the algorithm used for the conversion of the models of automatic type in models of Petri Net type; the validation conditions of the obtained results through the analysis of the models of Petri Net type equivalent to the basic models of automatic type.
Having as a base the theoretical considerations, the approach of the TSAA field is done in the first phase through a general presentation of TSAA and of their basic structures, continuing with the modelling of the constitutive elements of TSAA with the help of the models of automatic type, starting from the basic structure and ending with generic models. The structural and behaviour analysis of the established models of automatic type is realized by their conversion in models of untimed Petri Net type, on these models is being made the structural and behaviour analysis.
Some conclusions are presented at the end of the chapter.

## 2. Theoretical considerations connected to the models of automatic type, respectively Petri Nets

Both modelling methods, automata respectively Petri Nets, have at the base the using of states and transitions for the description of a system, it results that between them are similarities and differences (Cassandras & Lafortune 2001).

Modelling with automata respectively Petri Nets, as well as the model type choosing it is left to the modeller latitude (Murata 1989).
In this way, if it is desired the modelling of a DES only based on its external events, without having an interest in an explicit way on the hidden activities, and then a model of automatic type is satisfactory. If instead is desired a refinement of the internal operations then a model of Petri Net type is more favourable (Cassandras & Lafortune 2001) (Pastravanu 1997).
For the model of automatic type it was considered that these are of deterministic type. A deterministic automata is defined as follows:

***Definition 1. Untimed deterministic automata (Cassandras & Lafortune 2001)***

A deterministic automata, marked by G, is a sextuple

$$G = (X, E, \delta, \Gamma, x_0) \tag{1}$$

where:

- $X$ is the set of states;
- $\sum$ is a finite set of events associated with the transitions from G;
- $\delta : X \times \sum \to X$ is the transition function: $\delta(x, e) = y$ , means the appearance of a transition labelled through the event e in the state $x$, which has as a result the transition in the state y; in general, $\delta$ is a partial function on his domain of definition.
- $\Gamma : X \to 2^E$ is the active event function; $\Gamma(x)$ is the set of all events e for which $\delta(x, e)$ is defined and is called by the active event from $G$ accordingly to the active state $x$.
- $x_0$ is the initial state of the system.

For the model of Petri Net type it was considered that these are of labelled type and untimed. An untimed labelled Petri Net is defined as follows.

***Definition 2. Labelled Petri Net (Cassandras & Lafortune 2001)***

A labelled Petri Net is a weighted graph

$$PN = (P, T, F, W, \Sigma, l, M_0) \tag{2}$$

(net topology), where:

- P represents the finite multitude of positions, *with* $P = \{p_1, p_2, p_3, \ldots, p_n\}$;
- T represents the finite multitude of transitions, with $T = \{ t_1, t_2, t_3, \ldots, t_m\}$;
- $F \subseteq (P \times T) \bigcup (T \times P)$ represents the multitude of arcs from the positions to transitions and from transitions to positions, each arc being represented by $(p_i, t_j)$, respectively $(t_j, p_i)$, where $i, j \in N$;
- $W : A \to \{1,2,3...\}$ represents the balanced function of arcs;
- $\sum$ represents the set of events for the transitions labels;
- $l : T \to \sum$ represents the labelled transition function;
- $M_0 \in N^n$ represents the initial marking of the net .

As it was mentioned earlier, the automatons and the Petri Nets operate with states and transitions. In this way are possible transformations of the models of automatic type in models of Petri Net type and vice versa. The subject of synthesis of the models from one

way of representation in the other one is presented in detail in literature.(Pastravanu 1997)(Cortadella et al 1995)(Cortadella et al 1998)(Hellgren et al 2001)(Cassez & Roux 2004).. The algorithm used for the conversion of untimed deterministic automata $G=(X,\sum,\delta,\Gamma,x_0,X_m)$, into an untimed Petri Net $PN=(P,T,F,W,M_0)$ is:

***Step 1.*** The determination of the multitude of positions *P*
Each state from X is transformed into an equivalent position in *P*.

$$\forall x_i \in X : x_i \rightarrow p_i, p_i \in P \tag{3}$$

which means that the multitude *P* has exactly so much elements that are in the multitude *X*.
***Step 2.*** The determination of the transitions multitude T
For each pair of states from *X*, designated by (*x,x'*), with

$$x' = \delta(x,e), \quad e \in \Gamma(x) \tag{4}$$

it has associated a transition labelled $t_i \in T$ , in the Petri Net.
In the end result the multitude:

$$T = \bigcup \{t : (x,x'), x, x' \in X, x' = \delta(x,e), e \in \Gamma(x)\} \tag{5}$$

*Step 3*. The determination of the arc multitude F
It is attached the arcs $(p,t),(t,p') \in F$ all having the weight 1,*w(p,t)=1, w(t,p')=1* if exists the correspondence $(x,x') \rightarrow (p,p')$
With other words:

$$if\ (x,x') \rightarrow (p,p') : x' = \delta(x,e), e \in \Gamma(x)\ then \exists (p,t)\ and\ (t,p')\ p,p' \in P, t \in T \tag{6}$$

In the end, result the multitude:

$$F = \{(p,t) : p \in P, t \in T\} \bigcup \{(t,p') : p' \in P, t \in T\} \tag{7}$$

and

$$W : F \rightarrow \{1\} \tag{8}$$

*Step 4.* The determination of the event multitude $\sum$
The set of events is obtained as being:

$$\sum = E \tag{9}$$

*Step 5*. The determination of transition functions *l*. The label attached to the transition is the event *e* which led to change of the state *x* in *x'*.

For the transition $t \in T : (p,t),(t,p') \in F$ the event and function l and associated label are defined as:

$$l(t) = e,\ e \in \sum and\, x' = \delta(x,e) \tag{10}$$

*Step 6.* The determination of initial marking $M_0$
The initial marking $M_0$ is assigned accordingly to the initial state $x_0$.

The way of using the conversion algorithm from the model of untimed deterministic automatic type in the model of untimed labelled Petri Net type is exemplified through the next example.
*Example* . It is considered the untimed deterministic automata from figure 1.

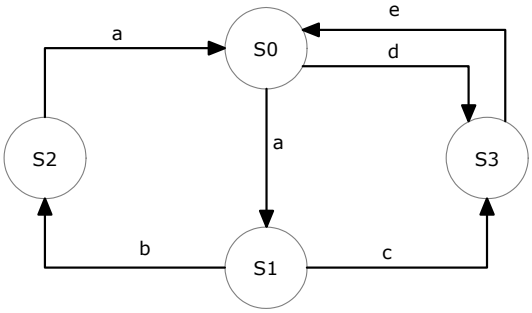



Fig. 1. The structure of the considered untimed deterministic automata

The considered automata formalized mathematical by: $AS = (X, E, \delta, \Gamma, x_0)$ where:

- The multitude of states: $X = \{S_0, S_1, S_2, S_3\}$
- The multitude of events: $E = \{a, b, c, d, e\}$
- The multitude of possible events and the transition functions of the states:

| | |
|---|---|
| $\Gamma(S_0) = \{a, d\}$ | $\delta(S_0, a) = S_1$<br>$\delta(S_0, d) = S_3$ |
| $\Gamma(S_1) = \{b, c\}$ | $\delta(S_1, b) = S_2$<br>$\delta(S_1, c) = S_3$ |
| $\Gamma(S_2) = \{a\}$ | $\delta(S_2, a) = S_0$ |
| $\Gamma(S_3) = \{e\}$ | $\delta(S_3, e) = S_0$ |

- The initial state: $x_0 = S_0$.

For obtaining the equivalent untimed labelled Petri Net is applied the presented algorithm:

*Step 1.* The determination of multitude of positions P:

$$S_i \to p_i;\ i = \overline{0,3}$$

from where results that $P = \{p_0, p_1, p_2, p_3\}$

*Step* 2. The determination of multitude of transitions *T*:

| State | States connection | The transition function | The associated transition |
|---|---|---|---|
| $S_0$ | $(S_0, S_1)$ | $\delta(S_0, a) = S_1$ | $t_1$ |
| | $(S_0, S_3)$ | $\delta(S_0, d) = S_3$ | $t_2$ |
| $S_1$ | $(S_1, S_2)$ | $\delta(S_1, b) = S_2$ | $t_3$ |
| | $(S_1, S_3)$ | $\delta(S_1, c) = S_3$ | $t_4$ |
| $S_2$ | $(S_2, S_0)$ | $\delta(S_2, a) = S_0$ | $t_5$ |
| $S_3$ | $(S_3, S_0)$ | $\delta(S_3, e) = S_0$ | $t_6$ |

From where results that $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$

*Step* 3. The determination of multitude of arcs F and of their shares

| Transition | States connection | The connection of equivalent positions | Arcs |
|---|---|---|---|
| $t_1$ | $(S_0, S_1)$ | $(p_0, p_1)$ | $(p_0, t_1), (t_1, p_1)$ |
| $t_2$ | $(S_0, S_3)$ | $(p_0, p_3)$ | $(p_0, t_2), (t_2, p_3)$ |
| $t_3$ | $(S_1, S_2)$ | $(p_2, p_3)$ | $(p_1, t_3), (t_3, p_2)$ |
| $t_4$ | $(S_1, S_3)$ | $(p_2, p_4)$ | $(p_1, t_4), (t_4, p_3)$ |
| $t_5$ | $(S_2, S_0)$ | $(p_3, p_1)$ | $(p_2, t_5), (t_5, p_0)$ |
| $t_6$ | $(S_3, S_0)$ | $(p_4, p_0)$ | $(p_3, t_6), (t_6, p_0)$ |

From where results that $F = \{(p_1, t_1), (p_1, t_2), (p_2, t_3), (p_2, t_4), (p_3, t_5), (p_4, t_6)\} \bigcup$
$\{(t_1, p_2), (t_2, p_4), (t_3, p_3), (t_4, p_4), (t_5, p_1), (t_6, p_1)$

The multitude of arcs shares:

$W = \{w(p_1, t_1) = 1, w(p_1, t_2) = 1, w(p_2, t_3) = 1, w(p_2, t_4) = 1, w(p_3, t_5) = 1, w(p_4, t_6) = 1,$
$w(t_1, p_2) = 1, w(t_2, p_4) = 1, w(t_3, p_3) = 1, w(t_4, p_4) = 1, w(t_5, p_1) = 1, w(t_6, p_1) = 1\}$

*Step* 4. The determination of multitude of events $\sum$

$$\sum = \{a, b, c, d, e\}$$

*Step* 5. The determination of transition functions *l*

| Transition | Connected arcs | Associated events |
|---|---|---|
| $t_1$ | $(p_1,t_1),(t_1,p_2)$ | $a$ |
| $t_2$ | $(p_1,t_2),(t_2,p_4)$ | $d$ |
| $t_3$ | $(p_2,t_3),(t_3,p_3)$ | $b$ |
| $t_4$ | $(p_2,t_4),(t_4,p_4)$ | $c$ |
| $t_5$ | $(p_3,t_5),(t_5,p_1)$ | $a$ |
| $t_6$ | $(p_4,t_6),(t_6,p_0)$ | $e$ |

*Step 6*. The determination of initial marking $M_0$

The initial state is $x_0 = \{S_0\}$ results the initial marking $M_0^T = [1,0,0,0]$

The graphical representation of the equivalent Petri Net, obtained after the application of the conversion algorithm, is presented in figure 2.

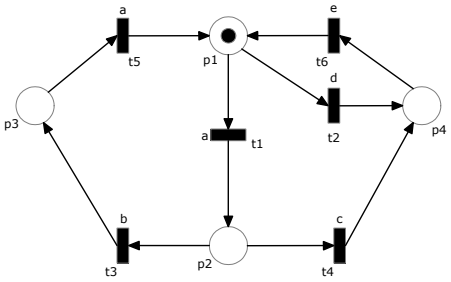



Fig. 2. The equivalent Petri Net

## 3. The principles of realization and functioning of the transport systems with accumulation areas

The industrial transport systems are of different types, depending on the effective mode in which is realized the transport. In this way exists horizontal transport systems or suspended transport systems. Each type of system has its own constructive and functional features, both of them belonging to the TSAA field. In figure 3 is presented a general view of TSAA.

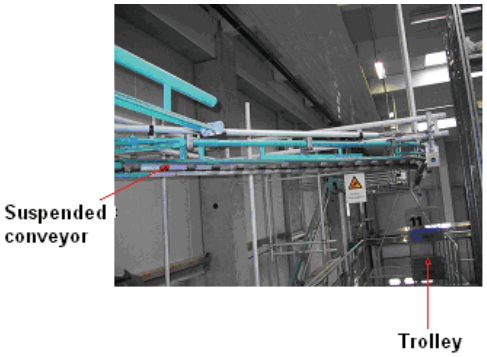



Fig. 4. Suspended transport system

Within the chapter it will be taken into consideration only the suspended transport systems, but the obtained results can be applied also to the horizontal transport systems. Constructive, these are based of the using of brush conveyors as a transport support and have as a main property the assurance of accumulation areas (jams) of the transported elements without being necessary the stopping of conveyor moving motors. The conveyor is moved by an electric motor connected through a mechanical reducer, situated at one of the conveyor extremities. This motor assures the moving of the conveyor with constant speed. The transport element used in the case of these transport systems is the trolley. Each trolley is identified in the system by a unique code bar. This code is used for the determination of the point in which is situated the trolley and for the computing of the route on which this has to follow.
In figure 4 is presented a general picture of the way in which the transport is performed inside a complete automated warehouse of exchange pieces.

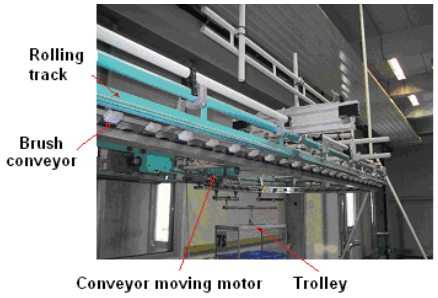



Fig. 5. General picture of a transport system

The main element which assures the stopping of a trolley depending on the used controlling algorithm is the stopper. In figure 5 is presented a stopper used in the case of these systems.

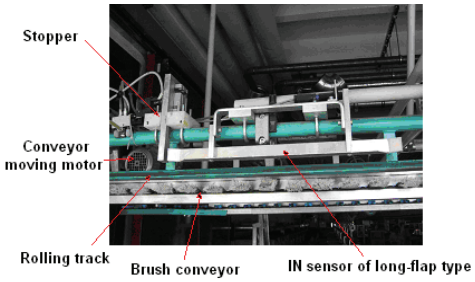



Fig. 6. Stopper and long-flap type sensor

In figure 5 is observed also the sensor which signalize the fact that in the front of the stopper there is a trolley.

## 4. Modelling the systems with accumulation areas

The basic idea in the modelling of such systems is that of assuring the modularization of the elements which makes the transfer of trolleys from one accumulation area into another (Ungureanu-Anghel 2006a)(Ungureanu-Anghel 2006b).
For the modelling of such systems it was established a basic structure (node of type 1 – an input an output), with the help of which can be obtained a general model regardless of its complexity. The observation which has to be made is connected to the fact that further will be no more references connected to the administration of the trolley flux and of their routes in the system, approaching only the issues connected to the modelling of nodes (Ungureanu-Anghel 2006a).

### 4.1 Node of type 1 – an input an output

The basic structure of such a node is presented in figure 6.

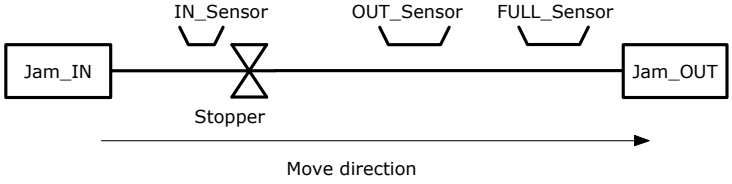



Fig. 7. Node of type 1. An input an output

As can be observed in the mentioned figure, such a node contains a single stopper, having an input and a single output. The sensors used have the following functions:

| | |
|---|---|
| IN_Sensor | The sensor from stopper. With the help of this sensor it is signalized the fact that a trolley is in the front of the stopper. |
| OUT_Sensor | The output sensor. With the help of this sensor is followed the moment in which a trolley has left the stopper area (of the node). |
| FULL_Sensor | Full sensor. With the help of this sensor it is checked the available space in the next jam. |

Jam_IN represents the input jam and Jam_OUT the output jam of the node. The moving motors were not presented because, as it was mentioned, they function continuous.
Before to present the functioning mode of the node of type 1, it is necessary to clearing up the mode of testing of FULL_Sensor.
Since the trolleys are in motion, the signal taken from FULL_Sensor is processed as it is presented in figure 7.

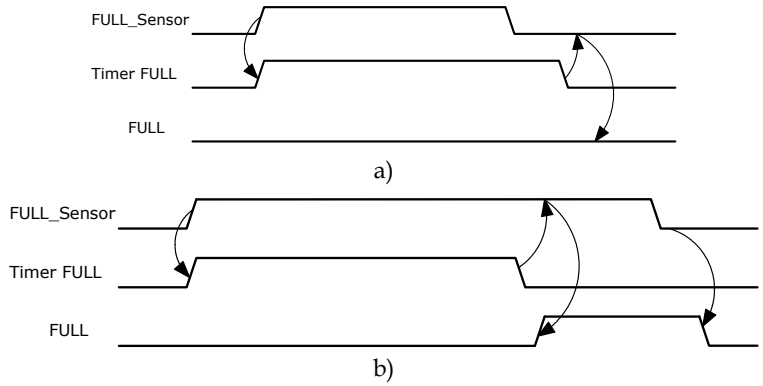



Fig. 8. The mode of processing the signal from FULL_Sensor. a) situation without detection FULL; b) situation of detection FULL

The FULL timer is used for verification with a certain delay of the signal from FULL_Sensor. If at the expiration of the timer the FULL_Sensor is inactive then the event FULL is inactive (figure 7.a), situation in which in Jam_OUT there is still space for at least a trolley. If at the expiration of the FULL timer the FULL_Sensor is active then the event FULL is set as being active, which corresponds to the situation Jam_OUT full. Deactivation of the FULL event is done in the moment in which FULL_Sensor becomes inactive (figure 7.b).
The functioning mode of the node of type 1 starts from the premise that the stopper is closed. A trolley situated in Jam_IN is transported by TEF and at one moment activates the IN_Sensor. The condition for moving the trolley in Jam_OUT area is given by: OUT_Sensor not to be active and FULL not to be active. In other words: in the area after the stopper doesn't have to be a trolley and in the Jam_OUT area has to be at least one free place. The mechanical structure of the node is thus realized so at the activation of OUT_Sensor the trolley has passed entirely by the stopper. The activation of OUT_Sensor has as an effect the closing of the stopper, realizing in this way the separation of two trolleys. A new trolley can be transferred from Jam_IN area in Jam_OUT area only after the OUT_Sensor is inactive, the previous trolley has left the area covered by the OUT_Sensor. For avoiding the uncontrollable situations, for example the stopper is opened and the trolley is jammed and it not touched the OUT_Sensor, are introduced two additional control timers named T1, respectively T2. T1 is used for verification of touching the sensor OUT_Sensor, while T2 for leaving of OUT_Sensor. If one of these timers expires before the touching, respectively the leaving of OUT_Sensor, it means that had appeared an abnormal state of functioning.
The functioning of the node of type 1 without error, described on the base of chronograms is presented in figure 8.

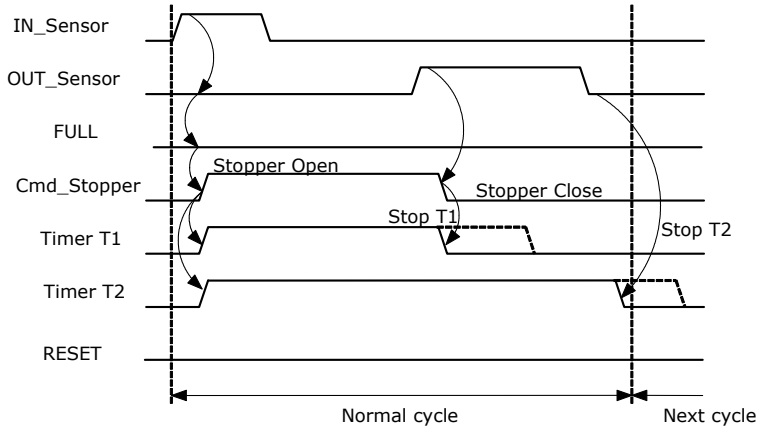



Fig. 9. The functioning chronograms of the node of type 1

The RESET event was introduced for giving the possibility of return in the WAIT state from the error states caused by the expiration of the timers T1 and T2.
The segments with doted line for the timers T1, respectively T2, represents the normal duration of functioning of the timers.

### 4.2 The modelling of the node of type 1 with the help of automata

The structure of the sequential automata taken into consideration for the modelling of the node of type 1 is presented in figure 9.
The meanings of the sequential automata states *AS_N1* are presented in the table 1.

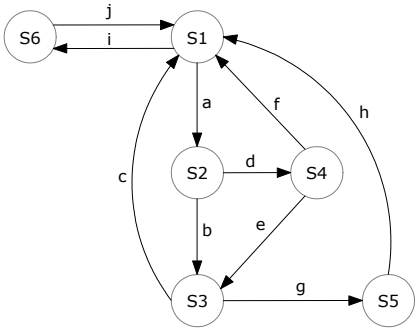



Fig. 10. The structure of the sequential automata corresponding to the node of type 1

| State | Comment |
|---|---|
| S1 | WAIT – The initial state of the system. In this state is waited that a trolley to reach the stopper (IN_Sensor = 1). Also in this state it is assured the transport of a trolley from the input sensor from the nod until the IN_Sensor. |
| S2 | OPEN – the state in which the stopper is open, assuring in this way the transfer of the trolley in the next jam. |
| S3 | CLOSE – state corresponding for closed stopper, but the trolley is still in node area, not being completely passed to the next jam. |
| S4 | ERROR 1 – error state which is installed if during the moving time of the trolley it has appeared a critical situation. T1 has expired |
| S5 | ERROR 2 – error state which is installed if during the moving time of the trolley it has appeared a critical situation. T2 has expired |
| S6 | ERROR 3 – error state which is installed if there is no trolley in IN_Sensor but OUT_ Sensor = 1 |

Table 1. The meanings of automata states AS_N1

The events under which take place the transitions between states are presented in table 2.
The net topology (result obtained with the help of PNT) is shown in figure 10.

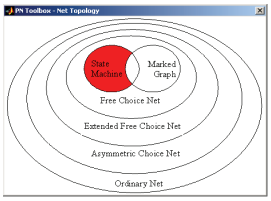



Fig. 12. The Petri Net topology validated by PNT
As can be seen, following the transformation of the model of automatic type in the Petri Net model, the net topology obtained is also of type automatic ("state machine"), which allows the analysis of the model of automatic type using the methods from the Petri Nets.

| Event | Comment |
|---|---|
| a | Event which assures the passing from state S1 in state S2.<br>Validated (active) based on the condition: IN_Sensor * !OUT_Sensor * !FULL. |
| b | Event which assures the passing from state S2 in state S3.<br>Validated (active) based on the condition: OUT_Sensor = 1. |
| c | Event which assures the passing from state S3 in state S1.<br>Validated (active) based on the condition: OUT_Sensor = 0. |
| d | Event which assures the passing from state S2 in state S4.<br>Validated (active) based on the condition: Timer T1 expirat . |
| e | Event which assures the passing from state S4 in state S3.<br>Validated (active) based on the condition: OUT_Sensor = 1. |
| f | Event which assures the passing from state S4 in state S1.<br>Validated (active) based on the condition: RESET = 1. |
| g | Event which assures the passing from state S3 in state S5.<br>Validated (active) based on the condition: Timer T2 expirat. |
| h | Event which assures the passing from state S5 in state S1.<br>Validated (active) based on the condition: OUT_Sensor + RESET = 1. |
| i | Event which assures the passing from state S1 in state S6.<br>Validated (active) based on the condition: !IN_Sensor * OUT_Sensor. |

Table 2. The meanings of the events corresponding to the automata AS_N1

The corresponding incidence matrices are:

The input incidence matrix:

$$A_I = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The output incidence matrix:

$$A_O = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Incidence matrix:

$$A = A_O - A_I = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

The corresponding cover tree is presented in figure 11.

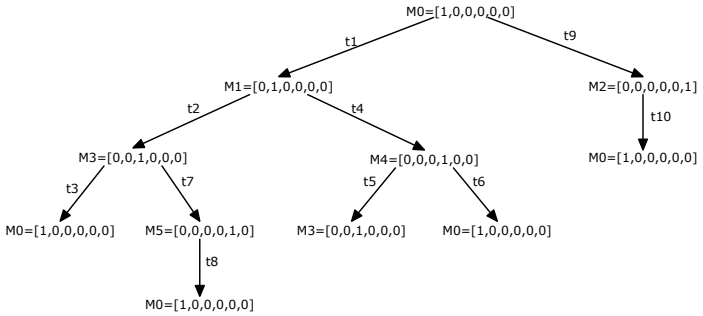



Fig. 13. The cover tree of Petri Net *PN_AS_N1*

Studying the behaviour properties of the Petri Net *PN_AS_N1*, based on the cover tree results:

- the net is *limited* (the symbol ω does not appear in the cover tree);
- the net is *safe* (the markings from all the cover tree nodes contains only „0" and „1");
- the net is *non-blocking* (all transitions have associated arcs in the cover tree);
- the net is *accessible* (any marking can be reached starting from $M_0$).

The corresponding cover graph is presented in figure 12.

The resulted structure for the cover graph confirms the fact that the model of Petri Net derived from the model of automatic type corresponding to the node of type 1 is *accessible*.

Concluding, it can be stated that the chosen model is *viable* from the behaviour point of view.

The analysis of the Petri Net *PN_AS_N1* from structural point of view it is done based on the analysis of the existence of the invariants of type *P* respectively *T*. To find out the number of invariants *P*, respectively *T*, it was applied the theorem for the determination of the invariants (Pastravanu 1997), which states that if the incidence matrix A (of dimension *nxm*) of a Petri Net, has the rank *r*, then: the net posses *m-r* base invariants P, and each invariant P of the net PN can be written as a linear combination of these; the net posses *n-r* base invariants T, and each invariant T of the net PN can be written as a combination of these.

In the considered case, the rank of the matrix A is: $rank\ A = 5$ .

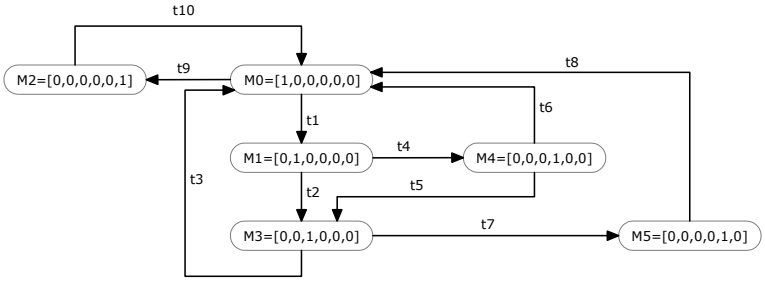



Fig. 14. The cover graph of Petri Net *PN_AS_N1*

Results that the Petri Net *PN_AS_N1* is covered by 1 invariant of type *P*. In a similar way, in the case of the invariants of type *T* results a number of 5 such invariants. The existence of

the invariants of type P, respectively T leads to the conclusion that *the Petri Net PN_AS_N1 is conservative and structural limited, respectively is consistent and repetitive* (Pastravanu 1997).
In the end can be *stated* that the chosen structure for a classic sequential automata corresponding to the node of type 1 is viable, the structure can be implemented, being sure that don't exists conditions of appearance of the blocking or appearance of uncontrollable situations.

### 4.3 The modelling of the node „one" input and „m" outputs with the help of automata

The base structure for such a node is presented in figure 13 (Ungureanu&Prostean 2007) (Ungureanu et al 2008).

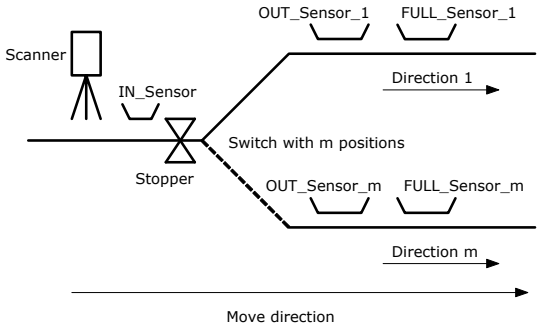



Fig. 15. The base structure of a node with one input and m outputs

Additional to the aspect of the node of type 1, is observed the appearance of a scanner and of a switch with *m* positions. With the help of the scanner, by reading the barcode from a trolley and following the list of destinations attached to the trolley, can be determined the direction on which has to be transferred the trolley and is commanded the corresponding switch positioning.
After the switch positioning everything is reduced to a node of type 1.
The position of the switch is not verified through some sensors, because if the switch is not accurate positioned it will appear an abnormal situation of functioning of no touching the corresponding output sensor.
The structure of the sequential automata considered for the modelling of such a node is presented in figure 14.

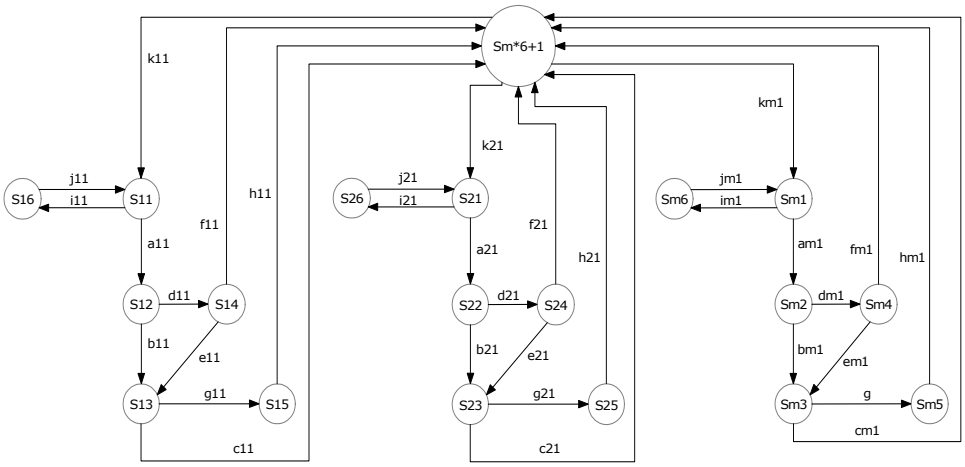



Fig. 16. The structure of the sequential automata corresponding to the node with *„one"* input and *„m"* outputs

The sequential automata of the node is realized by the synchronization of $m$ sequential automata *AS_N1*, one for each input line (stopper).
Mathematically, the automata *AS_Nm* corresponding to the node with one input and m outputs is described as follows:

- the multitude of states: $X = \bigcup_{i=1}^{m} \{Si1, Si2, Si3, Si4, Si5, Si6\} \cup \{S(m \cdot 6 + 1)\}$
- the multitude of events: $\Sigma = \bigcup_{i=1}^{m} \{ai1, bi1, ci1, di1, ei1, fi1, gi1, hi1, ii1, ji1, ki1\}$
- the multitude of possible events and the transition functions of the states:

$\Gamma(S) = \bigcup_{i=1}^{m} \Gamma(Si)$ where:

$\Gamma(Si1) = \{ai\}$ $\quad \delta(Si1, ai) = Si2$

$\Gamma(Si2) = \{bi, di\}$ $\quad \delta(Si2, bi) = Si3, \quad \delta(Si2, di) = Si4$

$\Gamma(Si3) = \{ci, fi\}$ $\quad \delta(Si3, ci) = S(4 \cdot m + 1), \quad \delta(Si3, fi) = Si4$

$\Gamma(Si4) = \{ei, gi\}$ $\quad \delta(Si4, ei) = Si3 \quad \delta(Si4, gi) = S(4 \cdot m + 1)$

$\Gamma(S(4 \cdot n + 1) = \bigcup_{i=1}^{m} \{hi\}$ $\quad \delta(S(4 \cdot m + 1), hi) = Si1$

- initial state: $x_0 = S(4 \cdot m + 1)$ .

The reasons linked to the transformation of the model of automatic type in the model of Petri Net type, presented in section 2, are valid also in the case of the node with one input and m outputs.
In figure 15 is presented the structure of the Petri Net associated to the sequential automata considered in the case of the node with one input and *m* outputs.

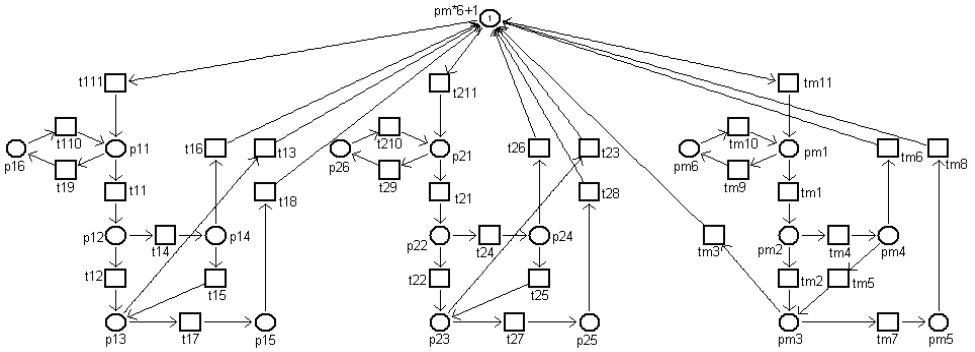



Fig. 17. The structure of the Petri Net associated to the sequential automata corresponding to the node with one input and m outputs

The net topology validated by PNT is presented in figure 16.

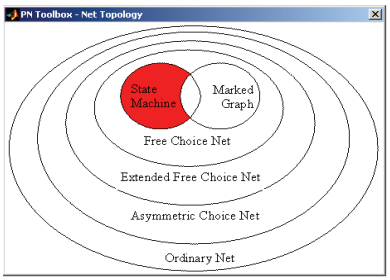



Fig. 18. The equivalent Petri Net topology validated by PNT

The obtained results confirms that after the effectuation of the conversion from the model of automatic type in the model of Petri Net type, the obtained Petri Net topology is also of automatic type („State machine").
The considered Petri Net is formalized by the quintuple:

$$PN_AS_Nm = (P, T, F, W, M_0)$$

where:

- $P = \bigcup_{i=1}^{m} \{p_{i1}, p_{i2}, p_{i3}, p_{i4}\} \cup \{p_{4m+1}\}$

- $T = \bigcup_{i=1}^{m} \{t_{i1}, t_{i2}, t_{i3}, t_{i4}, t_{i5}, t_{i6}, t_{i7}, t_{i8}\}$
- $F = \bigcup_{i=1}^{m} \{\{(p_{i1}, t_{i1}), (p_{i2}, t_{i2}), (p_{i2}, t_{i4}), (p_{i3}, t_{i6}), (p_{i3}, t_{i3}), (p_{i4}, t_{i5}), (p_{i4}, t_{i7}), (p_{4m+1}, t_{i8})\} \cup$
  $\{(ti_1, p_{i2}), (t_{i2}, p_{i3}), (t_{i3}, p_{4m+1}), (t_{i4}, p_{i4}), (t_{i5}, p_{i3}), (t_{i6}, p_{i4}), (t_{i7}, p_{4n+1}), (t_{i8}, p_{i1})\}\}$
- $W(p_{i1}, t_{i1}) = 1, W(p_{i2}, t_{i2}) = 1, W(p_{i2}, t_{i4}) = 1, W(p_{i3}, t_{i6}) = 1, W(p_{i3}, t_{i3}) = 1, W(p_{i4}, t_{i5}) = 1,$
  $W(p_{i4}, t_{i7}) = 1, W(p_{4m+1}, t_{i8}) = 1, W(t_{i1}, p_{i2}) = 1, W(t_{i2}, p_{i3}) = 1, W(t_{i3}, p_{4m+1}) = 1, W(t_{i4}, p_{i4}) = 1,$
  $W(t_{i5}, p_{i3}) = 1, W(t_{i6}, p_{i4}) = 1, W(t_{i7}, p_{4m+1}) = 1, W(t_{i8}, p_{i1}) = 1,$
- $M_0 = [0,0,0,0,0,0,0,0,0,0,0,0,...,1]^T$

The incidence matrices corresponding to the output direction *i*, where the columns are $p_{i1}$, $p_{i2}$, $p_{i3}$ , $p_{i4}$ , $p_{i5}$ and $p_{i6}$ and the lines: $t_{i1}$, $t_{i2}$, $t_{i3}$, $t_{i4}$, $t_{i5}$, $t_{i6}$, $t_{i7}$, $t_{i8}$ , $t_{i9}$, $t_{i10}$ and $t_{i11}$ are.

Input incidence matrix:

$$A_{Ii} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Output incidence matrix:

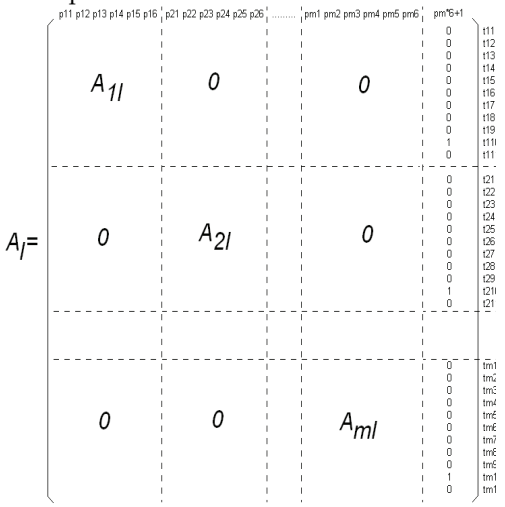
$$A_{Oi} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Incidence matrix:

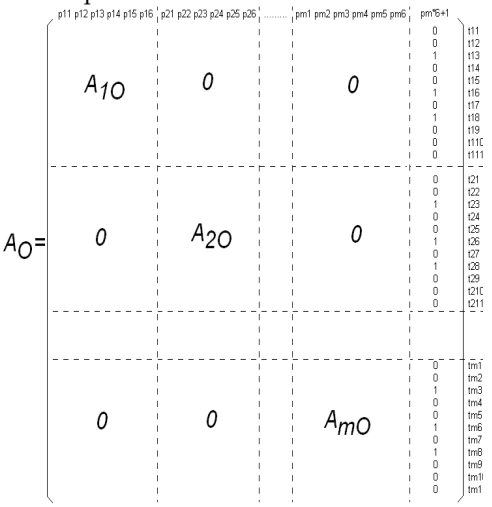
$$A_i = A_{Oi} - A_{Ii} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Having the matrices structure for a direction, the matrices corresponding to the node with *one* input (a stopper) and *m* outputs are:

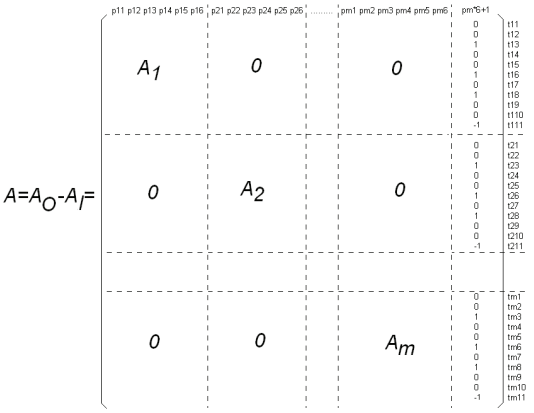
The input incidence matrix:




The output incidence matrix:




The incidence matrix:




The corresponding cover tree is presented in figure 17.

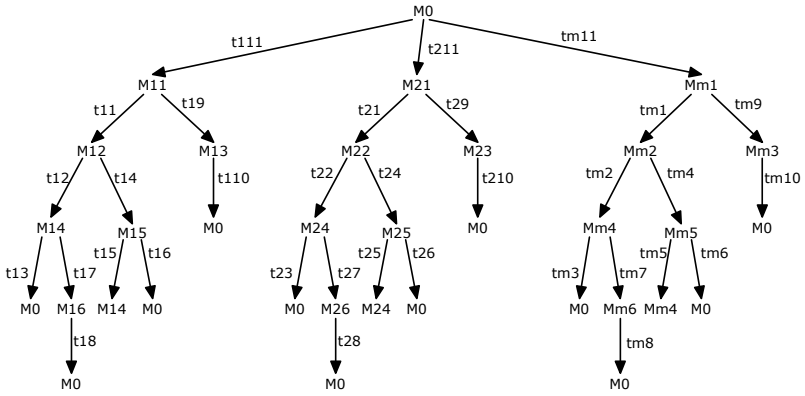



Fig. 19. The cover tree of the Petri Net *PN_AS_Nn*

Where:
M0 = [0,0,0,0,0,0,0,0,0,0,0,0,…,0,0,0,0,0,0,1],
M11 = [1,0,0,0,0,0,0,0,0,0,0,0,…,0,0,0,0,0,0,0]
M12 = [0,1,0,0,0,0,0,0,0,0,0,0,….,0,0,0,0,0,0,0]
M13 = [0,0,0,0,0,1,0,0,0,0,0,0,….,0,0,0,0,0,0,0]
M14 = [0,0,1,0,0,0,0,0,0,0,0,0,…..,0,0,0,0,0,0,0]
M15 = [0,0,0,1,0,0,0,0,0,0,0,0,…..,0,0,0,0,0,0,0]
M16 = [0,0,0,0,1,0,0,0,0,0,0,0,…..,0,0,0,0,0,0,0]
M21 = [0,0,0,0,0,0,1,0,0,0,0,0,…..,0,0,0,0,0,0,0]
M22 = [0,0,0,0,0,0,0,1,0,0,0,0,…..,0,0,0,0,0,0,0]
M23 = [0,0,0,0,0,0,0,0,0,0,0,1,…..,0,0,0,0,0,0,0]
M24 = [0,0,0,0,0,0,0,0,1,0,0,0,…..,0,0,0,0,0,0,0]
M25 = [0,0,0,0,0,0,0,0,0,1,0,0,…..,0,0,0,0,0,0,0]
M26 = [0,0,0,0,0,0,0,0,0,0,1,0,…..,0,0,0,0,0,0,0]
Mm1 = [1,0,0,0,0,0,0,0,0,0,0,0,….,1,0,0,0,0,0,0]
Mm2 = [0,1,0,0,0,0,0,0,0,0,0,0,….,0,1,0,0,0,0,0]
Mm3 = [0,0,0,0,0,1,0,0,0,0,0,0,….,0,0,0,0,0,1,0]
Mm4 = [0,0,1,0,0,0,0,0,0,0,0,0,…..,0,0,1,0,0,0,0]
Mm5 = [0,0,0,1,0,0,0,0,0,0,0,0,…..,0,0,0,1,0,0,0]
Mm6 = [0,0,0,0,1,0,0,0,0,0,0,0,…..,0,0,0,0,1,0,0]

Studying the behaviour properties of the Petri Net *PN_AS_Nm*, based on the cover tree results:

- the net is *limited* (the symbol ω does not appear in the cover tree);
- the net is *safe* (the markings from all the cover tree nodes contains only „0" and „1");
- the net is *non-blocking* (all transitions have associated arcs in the cover tree);
- the net is *accessible* (any marking can be reached starting from $M_0$).

Concluding, it can be stated that the chosen model is *viable* from the behaviour point of view.

The analysis of the Petri Net *PN_AS_Nm* from structural point of view it is done based on the analysis of the existence of invariants of type *P* respectively *T* (Pastravanu 1997).
The rank of the matrix A is in this case:

$$rank\ A = nr_out_node \cdot rankN_1$$

where:

- *nr_out_node* – represents the number of the outputs of the node;
- $rankN_1$ – represents the rank of the incidence matrix corresponding to the node of type 1 ($rankN_1$=4).

The adequate number of invariants P can be calculated with:

$$Nr_Inv_P = m - rank\ A = m - nr_out_node \cdot rankN_1$$

and the number of invariants *T* with:

$$Nr_Inv_T = n - rank\ A = n - nr_out_node \cdot rankN_1$$

### 4.4 The modelling of the node with n inputs and m outputs with the help of automata

The basic structure of such a node is presented in figure 18 (Ungureanu et al 2008).

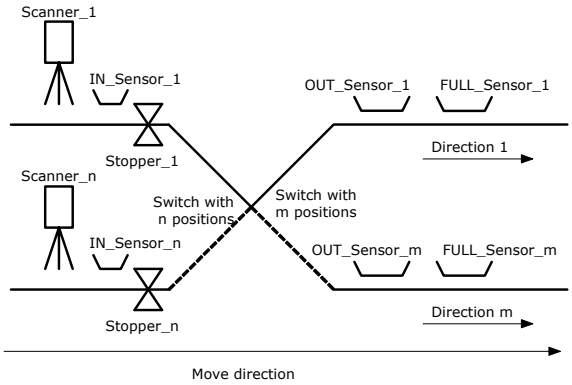



Fig. 20. The principle structure of a node with n inputs and m outputs

Towards the node with *one* input and *m* outputs, in the case of this node we have *n* inputs and a switch with *n* positions.
The condition of functioning for such a node is based on the mutual exclusion, namely a single trolley can be processed at a given moment. The administration of the trolleys from the input sensors can be realized in two ways: based on fixed priorities, respectively on FIFO principle.
Within this chapter is not concerned the mode of administration but strictly the mode of realization of the transfer of trolleys.
After the switch positioning with *n* positions, can be observed that all is resumed to a node with *one* input and *m* outputs.
The structure of the sequential automaton considered for the modelling of a node with *n* inputs and *m* outputs is presented in figure 19.

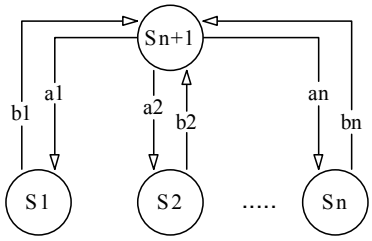



Fig. 21. The structure of the sequential automata corresponding to the node with *n* inputs and *m* outputs

In the above figure S1, S2 and Sn represent the automata which are modelling a node with *one* input and *m* outputs (are the models of type *AS_Nm*). The automata synchronization is realized through the additional state Sn+1.
The event *ai* corresponds to the validation of direction *i* after the barcode identification (result from the scanning operation). The event bi is generated by the automata *Si* at the end of the moving cycle.
The automata *AS_Nn-m* corresponding to the node with *n* inputs and *m* outputs is described as follows:

- the multitude of states:

$$X = \bigcup_{i=1}^{n} \{S_i\} \cup \{S_{n+1}\}$$

- the multitude of events:

$$\Sigma = \bigcup_{i=1}^{n} \{a_i, b_i\}$$

- the multitude of possible events and the transition functions of the states

$$\Gamma(S) = \bigcup_{i=1}^{m} \Gamma(Si)$$

where:

$$\Gamma(Si) = \{bi\} \qquad \delta(Si, bi) = S(n+1)$$
$$\Gamma(S(n+1)) = \{a1, a2, ..., an\} \qquad \delta(S(n+1), ai) = Si$$

- the initial state: $x_0 = S(n+1)$ .

The sub-models used for the modelling of states $S_i$, $i=1,...,n$, are defined identical with the ones presented in the case of the model are *AS_Nm*.

Applying the transformation algorithm of the automata in Petri Nets results:

- The multitude of positions:

$$P = \bigcup_{i=1}^{m} \{pi\} \cup \{p(n+1)\} ,$$

where

$$pi = \{Si\} \text{ and } p(n+1) = \{S(n+1)\};$$

- the multitude of transitions:
  - For $pi$ $(pi,p(n+1))$ $p(n+1)=\delta(pi,ti1)$ $ti1=\{bi\}$
  - For $p(n+1)$ $(p(n+1),pi)$ $pi=\delta(p(n+1),ti2)$ $ti2=\{ai\}$

The obtained Petri Net from the conversion operation is presented in figure 20.

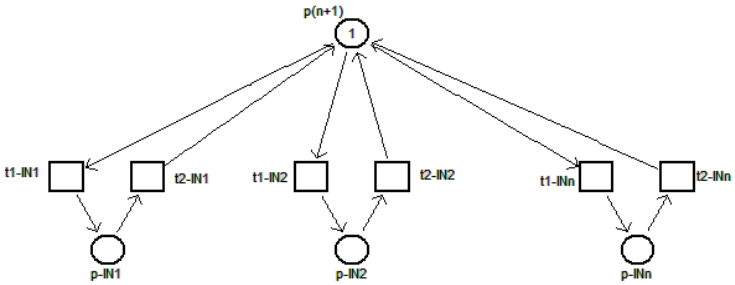



Fig. 22. The Petri Net corresponding to the modelling of the node with $n$ inputs and $m$ outputs derived from the sequential automata $AS_Nm$

The net topology validated by PNT is presented in figure 21.

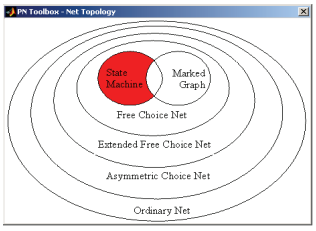



Fig. 23. The equivalent Petri Net topology validated by PNT

The Petri Net considered for the analysis of the sub-model is formalized by the quintuple:

$$PN_AS_Nn-m = (P,T,F,W,M_0)$$

where:

- $P = \bigcup_{i=1}^{n} \{p-IN_i\} \cup \{p_{n+1}\}$
- $T = \bigcup_{i=1}^{n} \{t1-IN_i, t2-IN_i\}$

- $F = \left\{ \bigcup_{i=1}^{n} \{\{(p - IN_i, t2 - IN_i)\} \cup \{(t1 - IN_i, p - IN_i\}\} \right\} \cup$
  $\left\{ \bigcup_{i=1}^{n} \{\{(p_{n+1}, t1 - IN_i)\} \cup \{(t2 - IN_i, p_{n+1})\}\} \right\}$
- $W(p - IN_i, t2 - IN_i) = 1, W(t1 - IN_i, p - IN_i) = 1,$
  $W(p_{n+1}, t1 - IN_i) = 1, W(t2 - IN_i, p_{n+1}) = 1$
- $M_0 = [0,0,.....,1]^T$

The incidence matrices are:

The input incidence matrix:

$$A_I = \begin{pmatrix} 0\ 0.....0\ 1 \\ 1\ 0.....0\ 0 \\ 0\ 0.....0\ 1 \\ 0\ 1.....0\ 0 \\ ..... \\ 0\ 0.....0\ 1 \\ 0\ 0.....1\ 0 \end{pmatrix}$$

The output incidence matrix:

$$A_O = \begin{pmatrix} 1\ 0.....0\ 0 \\ 0\ 0.....0\ 1 \\ 0\ 1.....0\ 0 \\ 0\ 0.....0\ 1 \\ ..... \\ 0\ 0.....1\ 0 \\ 0\ 0.....0\ 1 \end{pmatrix}$$

The incidence matrix:

$$A = A_O - A_I = \begin{pmatrix} 1 & 0..... & 0 & -1 \\ -1 & 0..... & 0 & 1 \\ 0 & 1..... & 0 & -1 \\ 0 & -1..... & 0 & 1 \\ ..... & & & \\ 0 & 0..... & 1 & -1 \\ 0 & 0..... & -1 & 1 \end{pmatrix}$$

The corresponding cover tree is presented in figure 22.

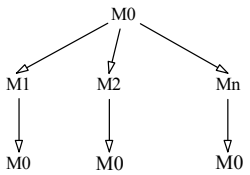



Fig. 24. The cover tree of the Petri Net $PN_{SUB}_AS_Nn\text{-}m$

Where:
M0=(0,0,....,0,1); M1=(1,0,....,0,0); M2=(0,1,....,0,0); Mn=(0,0,....,1,0);

Studying the behaviour properties of the Petri Net *PN_AS_Nn-m*, based on the cover tree results:

- the net is *limited* (the symbol ω does not appear in the cover tree);

- the net is *safe* (the markings from all the cover tree nodes contains only „0" and „1");
- the net is *non-blocking* (all transitions have associated arcs in the cover tree);
- the net is *accessible* (any marking can be reached starting from $M_0$).

The corresponding cover graph is presented in figure 23.

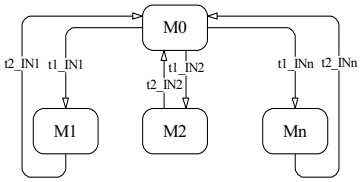



Fig. 25. The cover graph of the Petri Net *PN_AS_Nn-m*

The obtained structure for the cover tree confirms that the Petri Net, obtained with the help of sub-models corresponding to the sequential automaton of the node with $n$ inputs and $m$ outputs, is *accessible*.

Concluding, it can be stated that the chosen model is *viable* from the behaviour point of view.

The analysis of the Petri Net *PN_AS_Nn-m* from structural point of view is done based on the analysis of the existence of the invariants of type *P* respectively *T* (Pastravanu 1997).

The rank of the matrix A is:

$$rank\ A = n$$

where:

- $n$ – represents the number of node inputs;

The adequate number of invariants *P* is given by the relation:

$$Nr_Inv_P = nr_positions - rank\ A = n + 1 - n = 1$$

and the adequate number of invariants *T* is:

$$Nr_Inv_T = nr_transitions - rank\ A = n * 2 - n = n$$

## 5. Conclusions

Within this chapter it was followed the establishment of a unifying methodology for use of Petri Nets for the structural and behaviour analysis of the automata with direct applications on the transport systems with accumulation areas.

After the presentation of the conversion algorithm of an untimed deterministic automata into a untimed labelled Petri Net, with respecting the condition that the equivalent Petri Net topology to be the state machine, it was proceed to the using of this algorithm for the analysis of the constituent modules of the transport systems with accumulation areas.

The final aim of the chapter was the establishment of a general model of automatic type, for a mechanical structure with n inputs and m outputs, constituent part of a TSAA. To obtain this general model was started from a basic TSAA structure, called node with an input and an output, for which was developed the appropriate model of automatic type. Validation of

this model was achieved by conversion in model of Petri Net type, its structural and behaviour analysis validating the proposed model of automatic type.
As a simulation environment for the analysis of the equivalent Petri Nets was used PetriNets Toolbox from Matlab.
Having as a base the results obtained for the node with an input and an output, it has passed to the development of a model of automatic type for a node with an input and m outputs. Validation of the model was done using the same analysis method as in the case of the node with an input and an output.
In the end was elaborated a model of automatic type for a complex structure with n inputs and m outputs.
The obtained results following the analysis of this model validated entirely the proposed solution.
Using the general model for a node with n inputs and m outputs it can be implemented any types of nodes in a TSAA.
Also, the obtained results lead to the conclusion that the approaching way of the structural and behaviour analysis of untimed deterministic automata, by its conversion in the untimed labelled Petri Net, is viable, with the explanation the equivalent Petri Net topology has to be a state machine. In the future is intended the extending of the areas of applicability of the proposed solution as well as of the results obtained for TSAA for the road and rail transport systems.

## 6. References

C. G. Cassandras, S. Lafortune, „Introduction to Discrete Event Systems", Kluwer Academic Publishers, 2001, Boston

F. Cassez, O. Roux (2004), "From Petri Nets to Timed Automata", Published by Elsevier Science B.V., 2004

S.Y. Chiang, A. Hu, S. M. Meerkov(2008),"Lean Buffering in Serial Production Lines With Nonidentical Exponential Machines", IEEE Transaction on Automation Science and Engineering, Vol. 5, No. 2, April (2008), pp 298-306

J. Cortadella, M. Kishinevsky, L. Lavago, and A. Yakovlev (1995), " Synthesizing Petri nets from state-based models", Proc. International Conference Computer-Aided Design (ICCAD), 1995, www.isi.upc.edu/~jordicf/gavina/BIB/CONFERENCE.html

J. Cortadella, M. Kishinevsky, L. Lavagno, A. Yakovlev (1998), " Deriving Petri Nets from Finite Transition Systems", IEEE Transactions on Computers, Vol. 47, No. 8, August 1998

A. Hellgren, M. Fabian and B. Lennartson. On the Execution of Discrete Event Systems as Sequential Function Charts, *Proceedings of the 2001 IEEE Conference on Control Application*s, Mexico City, Mexico, September 2001.

M. Ge, Y. Xu, R. Du (2008), "An Intelligent Online Monitoring and Diagnostic System for Manufacturing Automation", IEEE Transaction on Automation Science and Engineering, Vol. 5, No. 1, January 2008, pp 127-139

A. Giua, C. Seatzu (2008), " Modeling and Supervisory Control of Railway Networks Using Petri Nets", IEEE Transactions on Automation Science and Engineering, Vol. 5, No. 3, July 2008, pp 431-445

Z. Li, D. Sun, Z. Zhang, J. Song, D. Xiao (2008),"Control Mechanism Analysis of Small-Agent Networks Using a Distinguished Node Model for Urban Traffic Controls", IEEE Transactions on Automation Science and Engineering, Vol. 5 No. 3, July 2008, pp 420-430

Murata T. (1989), "Petri Nets: Properties, Analysis and Applications", Proceedings IEEE, Vol 77, pp 541 – 580

O. Pastravanu (1997), "*Discrete Event Systems. Qualitative techniques in a Petri net framework*", Matrix Rom, ISBN 973-9254-61-6, Bucuresti, Romania 1997

A. Polic, K. Jezernik (2008), "Closed-Loop Matrix Based Model of Discrete Event Systems for Machine Logic Control Design", IEEE Transactions on Industrial Informatics, Vol. 1 No. 1, February 2008, pp 39-46

D. Ungureanu-Anghel (2006a), "Modeling the basic components of automatic transport systems with accumulation areas using sequential automata converted in Petri nets", Sceintific Buletin of "Politehnica" University of Timisoara, Romania, Transactions on Automatic Control and Computer Science, Vol. 51 (65), No. 3, 2006

D. Ungureanu-Anghel (2006b), „Two general untimed Petri net models for the basic components of automatic transport systems with accumulation areas", Sceintific Buletin of "Politehnica" University of Timisoara, Romania, Transactions on Automatic Control and Computer Science, Vol. 51 (65), No. 4, 2006

D. Ungureanu-Anghel, O. Prostean (2007), An Untimed Petri Net Model for an Automatic Transport System with Accumulation Areas, Proceedings of 4th International Symposium on Applied Computational Intelligence and Informatics, SACI 2007, Timisoara, Romania, May 17-18, 2007, pp. 279-284.

D. Ungureanu-Anghel, O. Prostean, D. Ionescu (2008), "General Untimed Sequential Automata Models for the General Components of Automatic Transport Systems with Accumulation Areas", INES 2008 • 12th International Conference on Intelligent Engineering Systems • February 25–29, 2008 Miami, Florida

# An Approach Based in Petri Net for Requirement Analysis

Ermeson Andrade, Paulo Maciel, Gustavo Callou, Bruno Nogueira
and Carlos Araujo
*Federal University of Pernambuco (UFPE)*
*Brazil*

## 1. Introduction

Embedded systems that have timing constraints are classified as real-time systems. In these systems, not only the logical results of computations are important, but also the time instant in which they are obtained. Hard real-time systems are those whose the respective timing constraints must be met at all cost, since violation might be catastrophic. Hence, time predictability is an essential issue (Barreto & Lima (2004)). In addition, the widespread expansion of mobile devices market has forced embedded systems companies to deal with several new challenges in order to provide complex systems in this market niche. In this context, energy consumption deserves special attention, since portable devices generally rely on constrained energy sources (e.g. battery) . As consequence, early estimation of the energy consumption can provide important insights to the designer about the battery lifetime as well as parts of the application that need optimization (Tavares et al. (2007)).

Nowadays, UML (UML (2005)) is the most adopted modeling language for system design in the software engineering organizations and industry. The main reasons are: (i) its friendly and intuitive notations, (ii) availability of commercial and open source tools that support the UML notations and (iii) autonomy of particular programming languages and development processes. However, UML does not provide support for quantitative notations. Quantitative notations are especially important when modeling Embedded Real-Time Systems (ERTS). Hence, we consider UML in combination with MARTE (UML Profile for Modeling and Analysis of Real-Time and Embedded systems) as specification language for the design of ERTS. MARTE foster the construction of models that may be used to make quantitative predictions regarding real-time and embedded features of systems taking into account both hardware and software characteristics (MARTE (2005)).

UML 2.0 is composed of several diagram types (e.g. activity, sequence, use case, class, timing and many others). Interaction Overview Diagram (IO) (UML (2005)) is adopted in this work due to its suitable characteristics for modeling requirements when dealing with ERTS, since UML-IO combine elements of activity diagrams with sequence diagrams to represent the embedded system behavior.

Without loss of generalization, this work aims to depict the mapping process of UML-IO into a Time Petri Net with Energy constraints (ETPN) in order to estimate the energy consumption and execution time of ERTS. These estimates are performed in the early stages of the embedded system life cycle, serving as one instrument for design decision-making process. First, the

execution time and energy consumption constraints are represented as MARTE profile annotations. After that, the ETPN model is generated by a mapping process, and finally the model is evaluated in order to find the Best Case Execution Time (BCET) and Worst Case Execution Time (WCET), the respective energy consumption, and also, adopted for qualitative analysis and verifications. Furthermore, the estimates obtained (time and energy consumption) from the model were compared with the measures obtained from the real hardware platform.
The remainder of the paper is organized as follows: Section 2 presents the adopted methodology. Section 3 introduces basic concepts regarding UML-IO, MARTE, ETPN and measuring activities. Section 4 presents the related work. Section 5 describes the mapping from UML-IO to an ETPN. Section 6 presents a case study and results. Section 7 concludes the paper and briefly discusses further work.

## 2. Methodology

This section introduces the set of activities related to the proposed design methodology. Figure 1 depicts the core activities of MEMBROS methodology, which organizes the activities in three groups: (i) requirement analysis; (ii) energy consumption and performance evaluation; and (iii) software synthesis. As follows, an overview of entire methodology is provided. However, this paper focuses on the activities regarding requirement analysis.

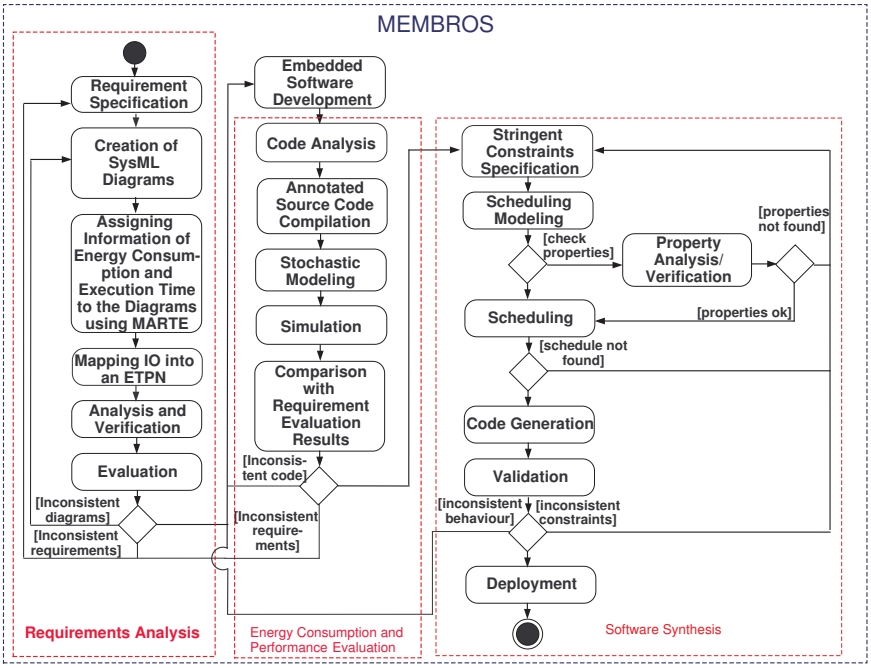



Fig. 1. MEMBROS methodology.

Initially, the activities regarding requirements specifications are performed. After created the requirement specification, then the system requirements are modeled using a set of UML-IOs, which represent the dynamic parts of the embedded software to be developed. Since

timing and energy constraints are of utmost importance in the systems of interest, the UML-IOs are annotated with timing and energy consumption information (e.g. initial estimates) using MARTE. Next, the annotated UML-IOs are automatically mapped into ETPN models in order to lay down a mathematical basis for analysis and verification of properties (e.g. absence of deadlock conditions between requirements). This activity also concerns to obtain best and worst case execution times and the respective energy consumptions, in such a way that the requirements are also evaluated in order to ensure that the timing and energy constraints can be met. As the UML-IOs are constructed by the designer, negative results in the evaluation activity may be not only related to inconsistent requirements, but also to inconsistent UML-IOs.

Afterwards, the embedded software is implemented taking into account the results obtained in previous activities. Once the source code implementation is concluded, the designer analyzes the code in order to assign probability values to conditional and iterative structures. The probability annotations allow the compiled code be evaluated in the context of time and energy consumption, in such a way that these costs may be estimated before running the code on the hardware platform. Next, the compiled code is automatically translated into a coloured Petri net (CPN) model in order to provide a basis for the stochastic simulation of the embedded software. Although not depicted in Figure 1, an architecture characterization activity is also considered to permit the construction of a library of CPN basic building blocks, which provide the foundation for the automatic generation of CPN stochastic models. From the CPN model (generated by the composition of basic blocks), a stochastic simulation of the compiled code is carried out considering the characteristics of the target platform. If the simulation results are in agreement with the requirements, the software synthesis is performed. More information about the activity of Performance Evaluation can be found in Callou et al. (2008).

Software synthesis activities are concerned with the stringent constraints (e.g. time and energy), and, in the general sense, it is composed of two subgroups of activities: (i) tasks' handling; and (ii) code generation. Tasks' handling is responsible for tasks' scheduling, resource management, and inter-task communication, whereas code generation deals with the static generation of the final source code, which includes a customized runtime support, namely, dispatcher. It is important to state that the concept of task is similar to *process*, in the sense that it is a concurrent unit activated during system runtime. For the following activities, it is assumed that the embedded software has been implemented as a set of concurrent hard real-time tasks.

Initially, a measurement activity is performed to obtain the tasks' timing information as well as the information regarding the hardware energy consumption. Next, the designer defines the specification of system stringent constraints, which consists of a set of concurrent tasks with their respective constraints, behavioral descriptions, information related to the hardware platform (e.g. voltage/frequency levels and energy consumption) as well as the energy constraint. Afterward, the specification is translated into an internal model able to represent concurrent activities, timing information, inter-task relations, such as precedence and mutual exclusion, as well as energy constraints. The adopted internal model is a time Petri net extension (TPNE), labeled with energy consumption values and code annotations. After generating the internal model (TPNE), the designer may firstly choose to perform property analysis/verification or carry out the scheduling activity. This work adopts a pre-runtime scheduling approach in order to find out a feasible schedule that satisfies timing, inter-task and energy constraints. Next, the feasible schedule is adopted as an input to the automatic code generation mecha-

nism, such that a tailored code is obtained with the respective runtime control, namely, dispatcher. Finally, the application is validated on a DVS (Dynamic Voltage Scaling) platform in order to check system behaviour as well as the respective constraints. Once the system is validated, it can be deployed to the real environment. More information about the activity of Software Synthesis can be found in Tavares et al. (2007).

## 3. Background

This section presents fundamental concepts for a better understanding of the rest of the paper.

### 3.1 UML

The Unified Modeling Language (UML) is a standard widely adopted graphical graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.
The main goals of UML are:

- provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models;
- provide extensibility and specialization mechanisms to extend the core concepts;
- be independent of particular programming languages and development processes;
- provide a formal basis for understanding the modeling language;
- integrate best practices.

UML is composed by 13 types of diagrams, which can be divided into three parts: behavior, interaction and structural diagrams. The behavioral diagrams specify the dynamic parts used in the system being modeled. The interaction diagrams represents a subset of behavior diagrams, that emphasize the control and data flows among the things (sub-systems, components and agents) in the system being modeled. Lastly, structural diagrams define the static and structural elements used in the system being modeled.
Behavioral diagrams are composed by the following diagrams: use case (provides a high-level description of the system functionality), activity (represents the flow of data and control between activities), sequence (represents the interaction between collaborating parts of a system), state machine (describes the state transitions and actions that a system or its parts performs in response to events), communication (shows an interaction among a set of participants over the course of time), interaction overview (combine elements of activity diagrams with sequence diagrams to show the flow of program execution) and timing (shows the behaviors of elements throughout a given period of time) (UML (2005)). It is important to stress that this paper presents the mapping process of UML-IO into an ETPN.

#### 3.1.1 Interaction Overview Diagram

An interaction overview diagram is an activity diagram in which the nodes represent interaction diagrams. Interaction diagrams can include sequence, communication, interaction overview and timing diagrams. Most of the notation for interaction overview diagrams is the same for activity diagrams. For example, initial, final, decision, merge, fork and join nodes are all the same. However, interaction overview diagrams introduce two new elements: interaction occurrences and interaction elements.

Interaction occurrences are references to existing interaction diagrams. An interaction occurrence is shown as a reference frame (Figure 2 (a)). On the other hand, interaction elements display a representation of existing interaction diagrams within a rectangular frame. They differ in that they display the contents of the references diagram inline (Figure 2 (b)).

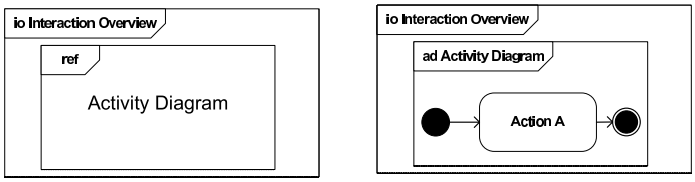



Fig. 2. New Elements: (a) interaction occurrence and (b) interaction element.

### 3.2 Marte

MARTE is a new UML profile standardized by the OMG. MARTE is used to define foundations for model-based description of real-time and embedded systems. These core concepts are then refined concerning: (i) modeling and (ii) analysis. The modeling part provides support required from specification to detailed design of real-time and embedded systems. On the other hand, the analysis part does not intend to define new techniques for analyzing systems, but to support them. Hence, MARTE aims providing facilities to annotate models with information required to perform specific analysis (MARTE (2005)).

Figure 3 illustrates an example of an activity diagram with time and energy constraints specified by MARTE. For this example the stereotype (*ResourceUsage*) and tagged values (*execTime* and *energy*) were used. The stereotype describes an action. Tagged values consist of a property name and an assigned value. In this example, the *ResourceUsage* stereotype describe, respectively, the delay and the energy consumption of the activity $A$, in this case, 10 seconds and 20 joules. The tagged values are $\{execTime = (10,'s')\}$ and $\{energy = (20,'j')\}$. More information about all stereotypes and tagged values supported by MARTE can be found in MARTE (2005).

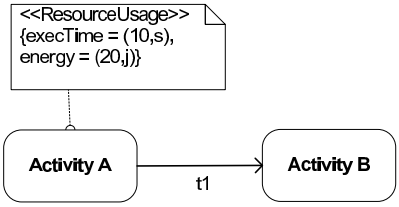



Fig. 3. Activity diagram and MARTE.

### 3.3 Computation Model

**Time Petri Net.** A Time Petri Net (TPN) (Merlin & Faber (1976)) is a bipartite directed graph represented by a tuple $\mathcal{P}= (P, T, F, W, m_0, I)$, where $P$ (set of places) and $T$ (set of transitions) are non-empty disjoint sets of nodes. The edges are represented by $F$, where $F \subseteq A = (P \times$

$T) \cup (T \times P)$. $W : A \to \mathbb{N}$ represents the weight of the edges, such that $W(f) = \{$(i) $x \in \mathbb{N}$, if $(f \in F)$, or (ii) 0, if $(f \notin F)\}$. *A TPN marking $m_i$ is a vector ($m_i \in \mathbb{N}^{|P|}$), and $m_0$ is the initial marking.* $I : T \to \mathbb{N} \times \mathbb{N}$ *represents the timing constraints, where* $I(t) = [EFT(t), LFT(t)]\ \forall t \in T$, $EFT(t) \leq LFT(t)$. $EFT(t)$ *is the Earliest Firing Time, and* $LFT(t)$ *is the Latest Firing Time.*
Considering the previous definition, places ($P$) represent local states and transitions ($T$) denote local actions. The set of arcs $F$ represents the relationships between places and transitions, in such a way that arcs connect places to transitions and vice-versa. Function $W$ assigns to each arc a natural number, which may be interpreted as the amount of parallel arcs. A marking vector $m_i$ associates to each place a natural number, which represents the number of tokens in the respective place. Graphically, places are represented by circles, transitions are depicted as bars or rectangles, arcs are represented by directed arrows labeled with the weight, and tokens (the marking) are generally represented by filled small circles. Fig. 4 depicts a Petri net model.

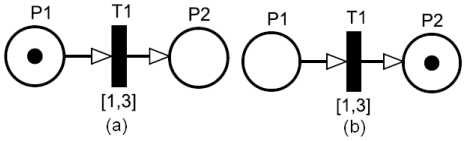



Fig. 4. Petri Net example

**Time Petri Net with Energy Consumption - ETPN $\mathcal{P}_\mathcal{E}$.** An extended TPN with energy consumption values is represented by $\mathcal{P}_\mathcal{E} = (\mathcal{P}, \mathcal{E})$. $\mathcal{P}$ is the underlying TPN, and $\mathcal{E}$:$T \to \mathbb{R}_+ \cup \{0\}$ is a function that assigns transitions to energy consumption values.
**Enabled Transitions.** A set of enabled transitions, at marking $m_i$, is denoted by: $ET(m_i) = \{t \in T \mid m_i(p_j) \geq W(p_j, t), \forall p_j \in P\}$
A transition $t \in T$ is *enabled*, if each input place $p \in P$ contains at least $W(p,t)$ tokens. The time elapsed, since the respective transition enabling, is denoted by a clock vector $c \in (\mathbb{N} \cup \{\#\})^{|T|}$, where # represents the undefined value for not enabled transitions. As an example, the clock vector for the net in Fig. 4(a) contains one element: $c(t_1) = 0$. At this point, the difference between static and dynamic firing intervals associated with transitions is required. The dynamic firing interval of transition $t$, $I_D(t) = [DLB(t), DUB(t)]$, is dynamically modified whenever the respective clock variable $c(t)$ is incremented, and $t$ does not fire. $DLB(t)$ is the Dynamic Lower Bound, and $DUB(t)$ is the Dynamic Upper Bound. The dynamic firing interval is computed in the following way: $I_D(t) = [DLB(t), DUB(t)]$, where $DLB(t) = max(0, EFT(t) - c(t))$, $DUB(t) = LFT(t) - c(t)$. Whenever $DLB(t) = 0$, $t$ can fire, and, when $DUB(t) = 0$, $t$ must fire, since *strong firing mode* is adopted.
**States.** Let $\mathcal{P}_\mathcal{E}$ be a time Petri net extended with energy consumption values, $M \subseteq \mathbb{N}^{|P|}$ be the set of reachable markings (e.g. all possible markings) of $\mathcal{P}_\mathcal{E}$, $C \subseteq (\mathbb{N} \cup \{\#\})^{|T|}$ be the set of clock vectors, and $E \subseteq \mathbb{R}_+ \cup \{0\}$ be the set of accumulated energy consumptions. The set of states $S$ of $\mathcal{P}_\mathcal{E}$ is given by $S \subseteq (M \times C \times E)$, that is, a state is defined by a marking, the respective clock vector, and the accumulated energy consumption from the initial state up to this state.
Considering the Petri net model in Fig. 4(a), the initial state is $s_0 = (m_0 = [1, 0], c_0 = [0], e_0 = 0)$.
**Firing Domain.** The *firing domain* for a transition $t$ at state $s$, is defined by the interval: $FD_s(t) = [DLB(t), \min(DUB(t_k))], \forall t_k \in ET(m)$.

Without loss of generality, enabled transitions are only related to the marking, and firable transitions take into account the marking, and their respective clock values (the time elapsed of each enabled transition). Considering firing domain, a firable transition $t$ at state $s$ can only fire in the interval denoted by $FD_s(t)$. In Fig. 4(a), at the initial state $s_0 = (m_0 = [1,0], c_0 = [0], e_0 = 0)$, $t_1$ is firable when $c_0(t_1) = 1$ and must fire when $c_0(t_1) = 3$ ($FD_{s_0}(t_1) = [1,3]$), if it neither has been fired nor disabled.

### 3.4 Measurement

This section briefly describes the measurement process and also the software that automates the measuring on the target plataform. In order to obtain the energy consumption and execution time values of a microcontroller instruction set, it may be necessary to adopt some measurement techniques in case such values cannot be obtained from manuals and datasheets.
The measurement scheme is presented in Figure 5, in which a hardware platform with the LPC2106 microcontroller, an oscilloscope and a desktop computer (PC) are connected. The AMALGHMA tool (Tavares & Maciel (2006)) - Advanced Measurement Algorithms for Hardware Architectures - has been implemented for automating the measuring activities. AMALGHMA adopts a set of statistical methods, such as bootstrap and parametric methods, which are important in the measurement process due to several factors, for instance: (i) oscilloscope resolution; and (ii) resistor error. Besides, this tool has been validated considering LPC2106 datasheet as well as ARM7TDMI-S reference manual.

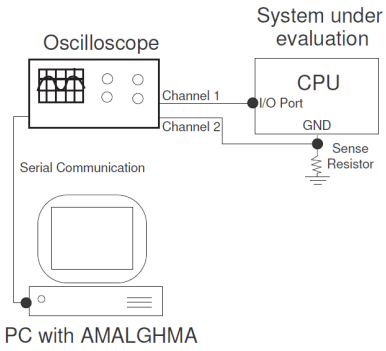



Fig. 5. Measurement Scheme.

## 4. Related Works

Many works are available in the literature reporting mappings of semi-formal representations to formal models. The majority of these works focus on qualitative analysis. Some of them are devoted to performance modeling, but, to the best of our present knowledge, none of them focus on both energy consumption and time evaluation of systems, besides of the papers published by our research team. It is also important to highlight that none work proposes the mapping of UML-IO into a formal model.
Merseguer et al. (2002) presented a systematic derivation of the UML-SM in fragments of a labeled Generalized Stochastic Petri Net (GSPN) that are composed into a single model that represents the behavior of the entire diagram. This work focused on the software performance evaluation in which all delays are represented by exponential distributions.

Trowitzsch & Zimmermann (2005) aimed at the transformation of UML-SM into a Stochastic Petri Nets (SPNs) for the performance evaluation of real-time systems. This approach proposed the decomposition of an UML-SM into basic elements, like states and transitions. These elements are translated into the corresponding SPN representations. The UML Profile for Schedulability Performance and Time (SPT) was used as a specification language for representing the restrictions imposed on the real-time systems. Amorim et al. (2005) proposed an approach to map Live Sequence Chart (LSC) language to an equivalent Coloured Petri Net (CPN) for analysis and verification of embedded systems' properties.
Another approach Lee et al. (2000) aimed to verify scenarios with Time Petri Net. This approach proposed a mechanism to check the acquired scenarios by indication any missing information or wrong information hidden in these scenarios. In Elkoutbi et al. (2002), the authors presented a requirement engineering process for real-time systems that aims a formal specification of the system using Timed Petri Net. In this approach, the first activity is the elaboration of the use case. Next, a corresponding scenario in form of sequence diagram is created for each use case. Afterwards, the scenario specification is derived into a Timed Petri Net. Finally, the partial TPNs are merged to obtain an integrated TPN, and then some verifications are performed.

## 5. Mapping UML-IO into an ETPN

This section describes how to derive the elements of UML-IO including time and energy constraints into an ETPN. The method consists of the deriving UML-IO basic elements and MARTE annotations into an ETPN. After that, all ETPN representation are composed into a single model that represents the behavior of the entire diagram. This section, firstly, presents the mapping of the elements used in this work that compose the nodes, that is, the interaction diagrams (activity and sequence). Lastly, an example illustrates the mapping of an UML-IO. Due lack of space, this paper focuses on the formalism and annotations needed to understand the case study.

### 5.1 Activity Diagrams

The Activity Diagrams (AD) are used to model the dynamic system aspects.

#### 5.1.1 Mapping Activities

The activities are represented by a rectangle with rounded edges (see Figure 6 (a)). They depict the invocation of an operation that may be physical or electronic. Additionally, an action represents a single step within an activity. In the ETPN model generated by the mapping process, the activities are represented by two PN-transitions. These PN-transitions may have assigned a time interval in which the activity must be executed, that is, the maximum and minimum time bounds. One PN-transition is used to represents the execution time for the worst case, and the other PN-transition for the best case of the activity. Energy constraints can be also assigned to these PN-transitions.
Figure 6 (a) depicts the mapping of an activity into an ETPN. The *t_ex_W_A* and *t_ex_B_A* PN-transitions model the worst case and best case, respectively. The time interval for these transitions are [45,45] and [5,5] , that is, the maximum and minimum times spent by activity A. Likewise, the energy consumption for the worst and best cases assigned to these PN-transitions are [70,70] and [18,18], respectively.
Additionally, the *t_in_W_A* and *t_in_B_A* PN-transitions are used due to semantics of TPN (Strong Firing Semantics) (Merlin & Faber (1976)) in order to allow PN-transitions with longer

delay to be able to fire, hence the time interval equal to [0,0] is assigned to these PN-transitions. However, if the constraints (execution time and energy consumption) are omitted from activity *A*, then they will be mapped into a PN-transition whose maximum and minimum times are zero (Figure 6 (b)). Furthermore, the *t_ex_W_A* and *t_ex_B_A* PN-transitions (see Figure 6 (a)) are assigned with a thin interval (interval in which the upper and lower bound values are the same (David (2005))), because the state space is considerably smaller than if only one PN-transition with the interval equal to [5,45] was adopted. Hence, this model allows a faster reachability graph path search. The *in_A* place represents the activity *A* entry as well as the choice point between the worst and best related to the execution time and energy consumption of the activity *A*. The others places (*W_A*, *B_A* and *out_A*) represent, respectively, the worst case state, best case state and activity *A* exit.

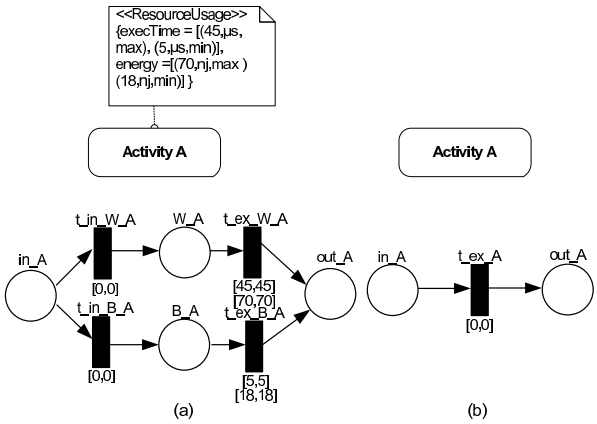



Fig. 6. Mapping activities.

### 5.1.2 Mapping Transitions

In UML-AD, transitions represent cause/effect activity relations. Figure 7 (a) illustrates the mapping of an AD-transition with time and energy constraints. The AD-transition (*t1*) from A to B is mapped in two PN-transitions (*t_ex_W_t1* and *t_ex_B_t1*), in which a time period of [25, 25] and [15, 15] seconds are assigned. These PN-transitions represent the execution time for the worst and best case, respectively. Figure 7 (a) also has energy constraints assigned to these PN-transitions.

AD-transitions that do not spend time and energy are mapped into PN-transitions whose maximum and minimum times are zero (Figure 7 (b)). The PN-transition generated is connected to the *out_A* and *in_B* places. These places represent, respectively, the activity *A* exit as well as the choice point between the worst and best related to the execution time and energy consumption of the *t1* AD-transition and the activity *B* entry.

### 5.1.3 Mapping Initial and Final States

An initial state represents the start point of a UML-AD. This initial state is mapped into a place in ETPN model, where the *staIni_A* place gets the initial marking of one token. Tokens are used in these models to simulate the dynamic behavior of systems. Furthermore, the

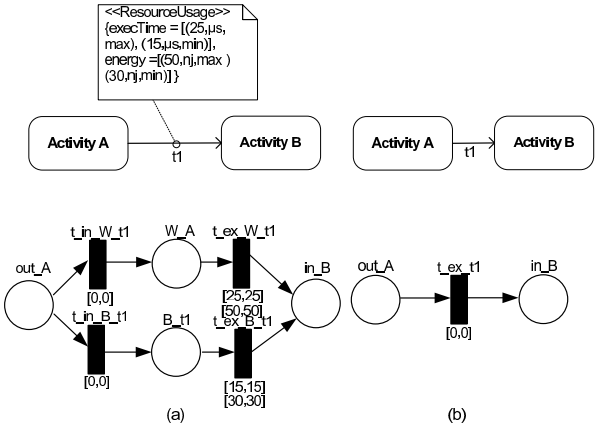



Fig. 7. Mapping transitions.

*t_in_A* PN-transition is used to represent the AD-transition between the initial state and the activity *A*. Figure 8 (a) illustrates a mapping of an initial state.
A final state represents the respective final state in UML-AD. This final state is mapped into a place in ETPN model (Figure 8 (b)), such that the presence of a token in the *endSta_A* place represents the end of the UML-AD. Moreover, the *t_end_A* PN-transition is used to represent the AD-transition between the activity *A* and the final state.

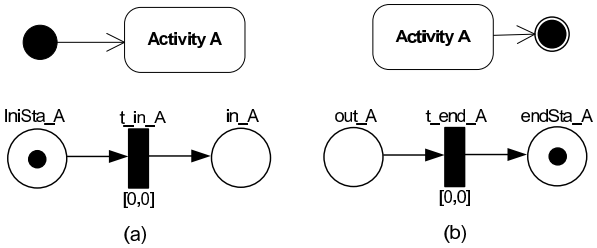



Fig. 8. Mapping initial and final states.

#### 5.1.4 Synchronization Bar

Synchronization bars are used to split processing (fork), or to resume processing when multiple activities have been completed (join). The synchronization bars are modeled as solid rectangles, with multiple transitions going in and/or out.
Figure 9 presents an example, in which the activity A is split into 2 activities (B and C). In the mapping process, the *t_fork_A_B* PN-transition represents the split processing and the *brc_B* and *brc_C* places represent the starting points for the branches (see Figure 9). On the other hand, Figure 11 depicts the synchronization process (join), that is, the synchronization between the activities A and B. In the mapping process, the *t_join_A_B* PN-transition is used to represent the synchronization process and the *join_A_B* place represents the synchronization point. In this case, the PN-transition only can be fired if both *out_A* and *out_B* places contain a token.

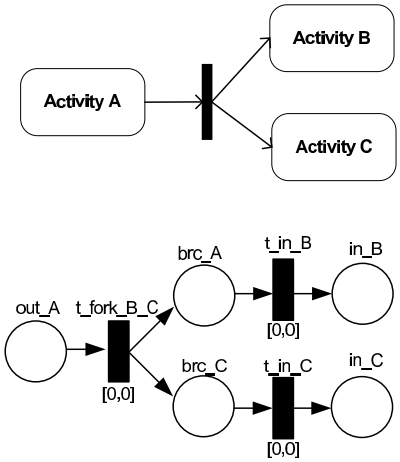



Fig. 9. Mapping fork.

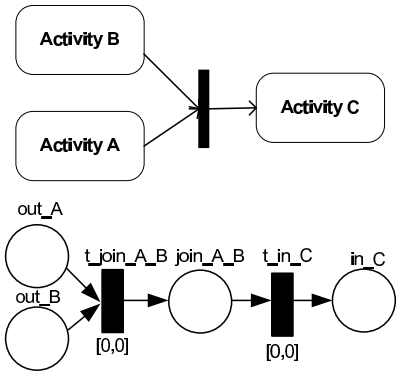



Fig. 10. Mapping join.

#### 5.1.5 Decision

In UML-AD, decision is represented by a diamond with one flow entering and several leaving. The flows leaving include conditions although some modelers may not indicate the conditions if it is obvious. In the mapping process, the *t_in_deci* PN-transition represents the entry on the decision and the places *brc_B* represents the decision point.

### 5.2 Sequence Diagram

Sequence Diagrams (SD) are commonly used UML diagram for representing element's collaboration over time.

#### 5.2.1 Mapping Lifeline and Message

A lifeline represents the involvement of a given participant in a particular interaction. The white rectangles on a lifeline (vertical dashed lines) are called activations (see Figure 12) and

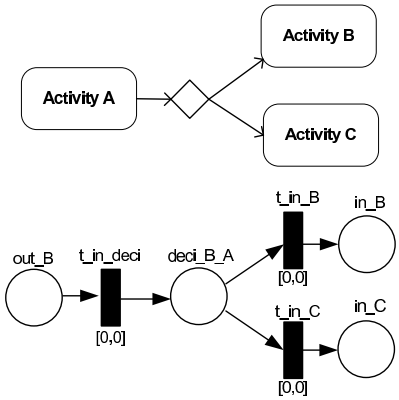



Fig. 11. Mapping decision.

indicate a participant response to a message. The communication between lifelines is performed by messages or calls, in the same order in which the events occur. Furthermore, the message specifies not only the communication type, but also the sender and the receiver. The return messages are an optional part of an UML-SD. In this paper, only asynchronous messages are considered. Figure 12 depicts an example, in which two participants (A and B) are communicating by two messages. The first message has time and energy constraints assigned. These constraints are specified by MARTE profile. The other message has neither time nor energy constraint assigned.

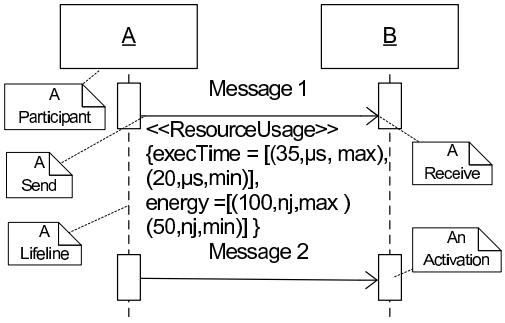



Fig. 12. UML-SD example.

Figure 13 presents the mapping of the UML-SD depicted in Figure 12 into an ETPN. The places (*start_A*, *D_1* and *end_A*) and the PN-transitions (*t_s_M1* and *t_s_M2*) represent the participant *A* lifetime. In this model, each place represents one state of that participant at a particular time instant, and the PN-transitions represent a state transition. The *start_A* place represents the start point on the lifeline *A*, and this place gets the initial marking of one token. The place *D_1* is a dummy place that interfaces the PN-transitions and the place *end_A* represents the lifeline end. Finally, *t_s_M1* and *t_s_M2* represent the message transmissions. In a similar manner, the participant *B* was mapped. The difference, in this case, is that the PN-transitions (*t_r_M1* and *t_r_M2*) represent the message receptions.

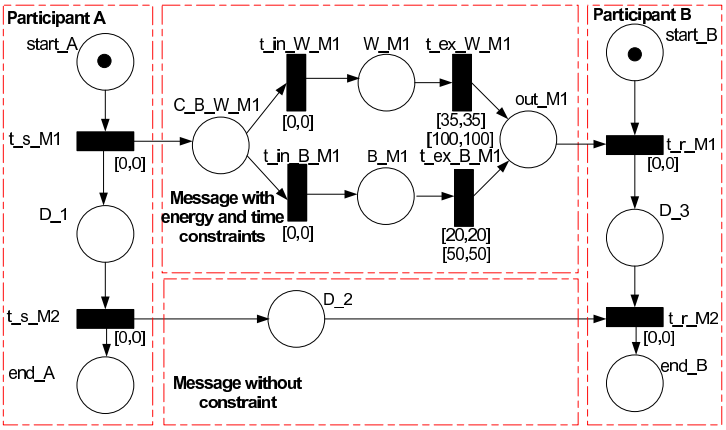



Fig. 13. Mapping UML-SD example.

Figure 13 also shows the mapping of messages. For example, *Message 1* represents the thin intervals of the execution time for the worst and best case, respectively, [35,35] and [20,20] assigned to the *t_ex_W_M1* and *t_ex_B_M1* PN-transitions. In a similar manner, the energy consumption constraints are considered. The energy consumption for the worst and best cases assigned are [100,100] and [50,50], respectively. The *C_B_W_M1* place represents the choice point between the worst and best case related to the execution time and energy consumption of the *Message 1* as well as the *Message 1* entry. The *W_M1*, *B_M1* and *out_M1* places represent, respectively, the worst case state, the best case state and the *Message 1* out.
However, if time and energy constraints are omitted from the messages as the *Message 2* shows (see Figure 12), then the mappings is performed as depicted in Figure 13, in which the place *D_2* is used as a dummy place for bridging the *t_s_M2* and *t_r_M2* PN-transitions.

### 5.2.2 Mapping self message

Figure 14 presents an example of self message, that is, the participant sends a message to himself.

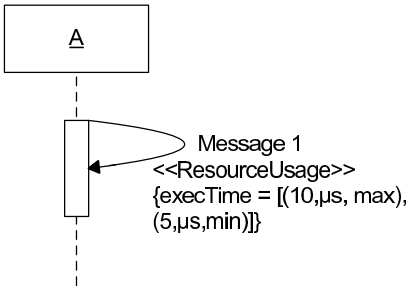



Fig. 14. Self a message example.

Figure 15 illustrates the mapping of a self message. The rules for mapping the self message are similar to the others mentioned above, the difference, in this case, is that the ETPN model backs to the same participant.

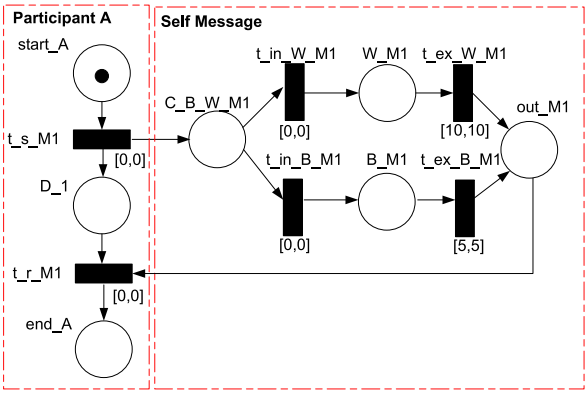



Fig. 15. Mapping self message.

### 5.2.3 Combined Fragments

A combined fragment is used to group sets of messages together in order to represent the SD conditional flows.

### 5.2.4 Mapping Alternatives

Alternatives are used to designate a mutually exclusive choice between two or more message sequences. The interaction operator *alt* (see Figure 16) shows that at least one of the operands will be executed.

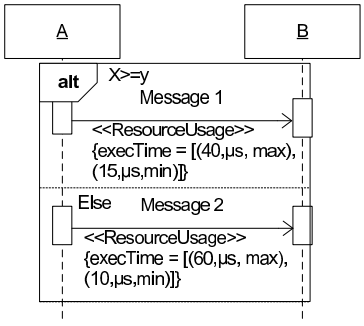



Fig. 16. Alternative combined fragment example.

Figure 17 depicts the mapping of alternatives presented in the Figure 16 into an ETPN. The rules for mapping this example are similar to the others mentioned before. However, this mapping adopts one place (alt_M1_M2) and two transitions (t_in_M1 and t_in_M2). The place represents the choice point between the operands *IF* and *ELSE* as well as the *alt* entry. One

transition is used to represent the *Message 1* entry (*IF*). The other is used to represents the *Message 2* entry (*ELSE*).

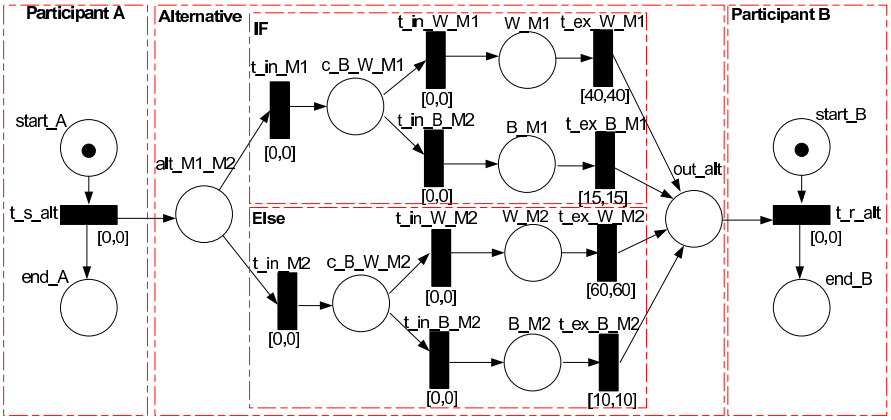



Fig. 17. Mapping alternative combined fragment example.

### 5.2.5 Mapping Parallel

The interaction operator *par* is used to define two or more processes that are concurrently executed. Figure 18 presents an example, in which the *Message 1* and *Message 2* are performed simultaneously.

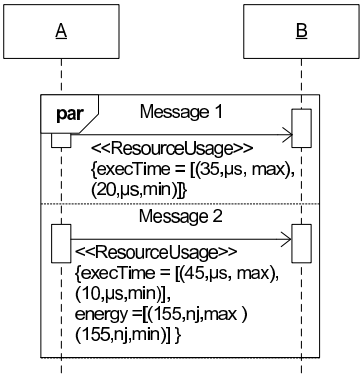



Fig. 18. Parallel combined fragment example.

Figure 19 depicts the mapping of parallel activities presented in the Figure 18 into an ETPN. The *t_in_par* and *t_syn_par* transitions represent the beginning and synchronization of concurrent activity execution, respectively. Each of these regions contains a distinct message and is mapped according to the rules mentioned before.

## 5.3 Mapping Interaction Overview Diagram

This section presents the mapping of an UML-IO example. Figure 20 (a) depicts this example in which two different interaction diagrams (activity and sequence) are connected by the *t1*

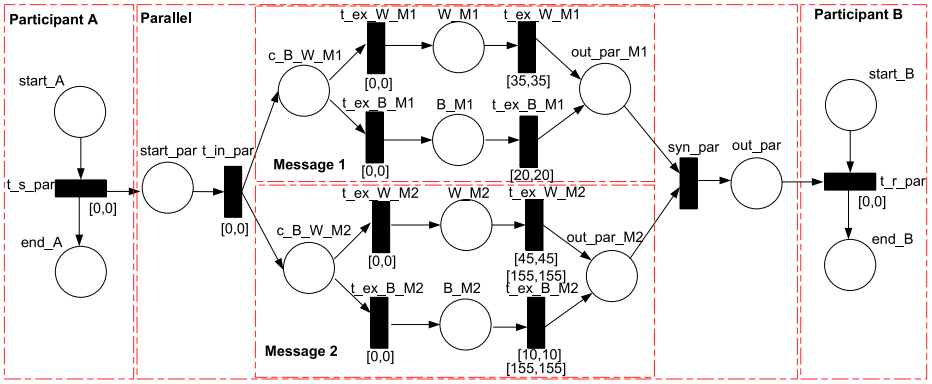



Fig. 19. Mapping parallel combined fragment.

IO-transition. The rules for mapping this example are similar to the others mentioned before. However, in the ETPN model (see Figure 20 (b)) the output arc of the *in_SD* place connects the *t_s_M1* PN-transition, and the output arc of the *t_r_M1* PN-transition connects the *out_sd* place. In this example, for sake of simplicity, the mapping of the initial and final states of UML-AD were omitted.

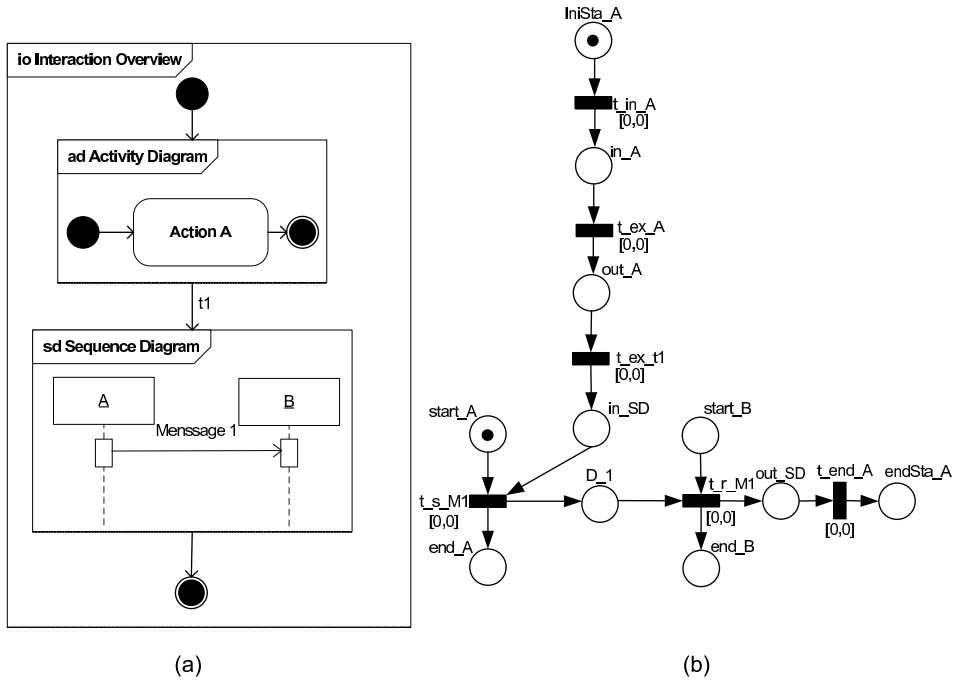



Fig. 20. Mapping UML-IO diagram.

## 6. A Case Study

In order to show the practical usability of the proposed mapping process, a pulse-oximeter specification (Júnior (1998)) was considered as a case study. This electronic equipment is responsible for measuring the blood oxygen saturation through a non-invasive method. A pulse-oximeter may be employed when a patient is sedated during surgical procedures. It checks the oxygen saturation to guarantee an acceptable level of blood oxygenation. This device is widely used in health care centers.

The pulse-oximeter is composed by three processes: (i) excitation, (ii) acquisition-control and (iii) management. The excitation process (see Figure 21) is responsible to dispatch pulse streams to the LEDs in order to generate radiation pulses. In this paper, only the excitation process was considered. Figure 21 depicts the excitation process according to Júnior (1998). Due to the restrictions of space, the ETPN model of the excitation process was omitted.

Figure 21 shows the UML-IO describing the excitation process. This process has requirement constraints (execution time and energy consumption). These constraints are deeply related to the hardware platform (Philips LPC2106 processor, an 32-bit microcontroler with ARM7 core). In addition, we consider the same values obtained from the pulse-oximeter specifications present in Júnior (1998) for modeling. As can be observed, the nodes represent interaction diagrams, in this case, we use the activity and sequence diagrams.

Initially the excitation process is started with the reception of an interruption in the the activity *Timer T1*. The timers are used to determine which process will be executed first. In the subsequent steps, the activities *Led Infrastructure*, *Set Led Guada* and *Set Led Red* are carry out. These activities are responsible for dispatching non-simultaneous pulse currents to the LEDs that cross the finger of a patient. In such cases, all activities have constraints of time and energy. For instance, the activity *Set Led Infra* has constraints of time and energy, and it is modeled with an <<ResourceUsage>> of: execTime = [(14,69,$\mu s$, max), (14,11,$\mu s$,min)] and energy = [(835,99, nJ, max ), (795,21, nJ, min)].

After all activities responsible for dispatching pulse currents to the LEDs were performed, the activity of the next node takes place. This node is responsible for controlling the dispatch of pulse currents to the LEDs. As can be observed, an alternative combination fragment element is used to designate a mutually exclusive choice between two calls (*CalibrationNotRun()* and *CalibrationRun()*). These calls are responsible, respectively, to modify the intensity of the pulse currents or not. The calls (*CalibrationNotRun()* and *CalibrationRun()*) have constraints of time and energy. For instance, the call *CalibrationRun()* is modeled with an <<ResourceUsage>> of: execTime = [(1,87,$\mu s$, max), (1,79,$\mu s$,min)] and energy =[(180,01, nJ, max ), (175,45, nJ, min)]. After the calibration of the attributes related to the LEDs, the excitation process returns to the node of the activity *Timer 1*, and waits for a new interruption to restart the process.

The following procedure is adopted to obtain the correspondent ETPN model that represents the activity diagrams present in the nodes of Figure 21. Each activity in the UML-AD is represented, basically, by two places and two PN-transitions. One place is used to represents the activity entry. The other place represents the activity exit, and the PN-transitions are used to represent the constraints assigned to activity. Additionally, two PN-transitions are adopted in the ETPN model due to semantics of TPN (Strong Firing Semantics). Besides, dummy places are used for bridging the PN-transitions. The beginning of UML-AD is depicted by initial state. The correspondent ETPN for initial state is a place with one token. Tokens are used in these ETPNs to simulate the dynamic behavior of systems. After all individual models have been built, the AD-transitions are mapped into PN-transitions in the ETPN model.

Furthermore, time and energy constraints are taken into account and included into ETPN PN-transitions.
Likewise, the following procedure is adopted to obtain the ETPN model that represents the sequence diagram presents in the node of Figure 21. Each participant is represented by at least two places and one transition in the ETPN model. One place with one token is used to represents the beginning of the involvement of a given participant. The other place represents the end of the involvement of a given participant, and the transition represents the send/receive of a message/call. After all individual ETPN models, that represent the participants, have been built, the messages/calls are mapped into transitions in the ETPN model. Messages/calls that do not have either time or energy constraints, will be mapped in one place, that is used as a dummy place for bridging the transitions. Furthermore, time and energy constraints are taken into account and included into ETPN transition.

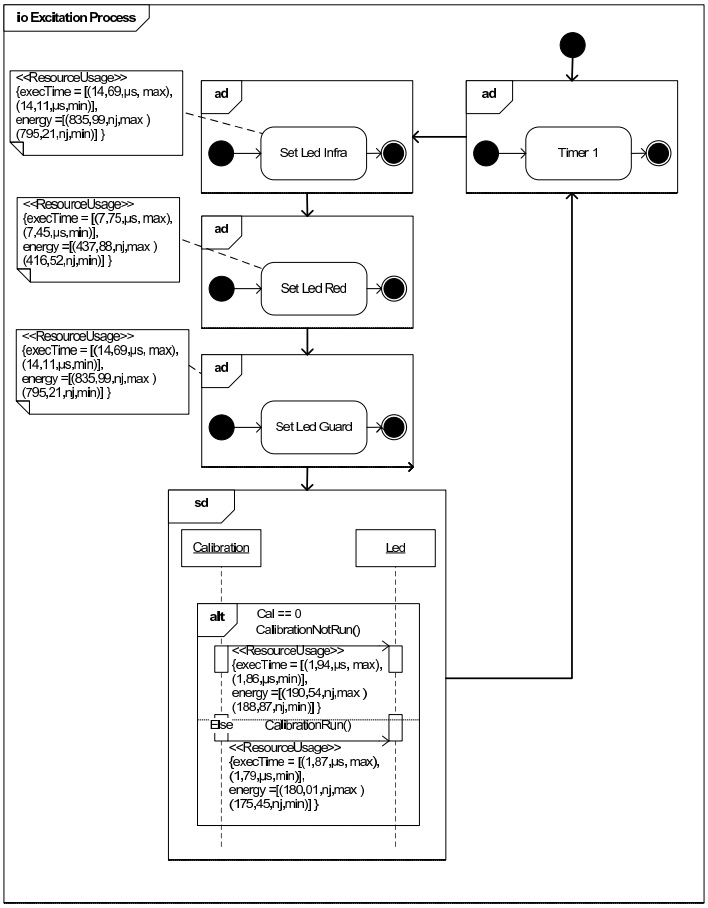



Fig. 21. UML-IO diagram of excitation process with time and energy constraints.

The analysis of the obtained ETPN model generated by the mapping process of the excitation process is safe, pure, homogeneous, conservative and bounded (David (2005)). The BCET

calculated of the excitation process was 37,46 $ms$ and the WCET was 39,07 $ms$. Moreover, the respective energy consumption of the excitation process was also computed. The results obtained were 2182,39 $\mu J$ and 2300,4 $\mu J$, respectively. The INA Tool (Starke & Roch (1999)) is adopted to compute the best and the worst path of the execution time. Once the worst and best path were found, then the energy consumption values are computed. In this paper, for the shortest and worst execution path time, respectively, the best and worst energy consumptions are computed.
The values measured on the hardware platform (execution time and energy consumption) were 38,88 $\mu s$ and 2251,84 nJ, respectively. The analysis results show that the execution time and energy consumption, when considered the best and the worst paths, computed errors were smaller than 5% in related to the measurements conducted on hardware platform. The experimental results show that the values computed from the models are quite similar to the real measurement on the pulse-oximeter system.
UML is a user-friendly specification language that supports the specification, analysis, design, verification and validation of a broad range of complex systems. So, if the advantages of UML are allied to the power of formal models, some misinterpretations can be avoided, allowing both: reducing the risks of faults propagations from early specification to final code, and system properties analysis and verification. Hence, it can be used to reduce the risks as well as the amount of money or effort that can be spend building embedded projects.

## 7. Conclusions

Requirement analysis is a critical task in any embedded real-time system project. Normally, these systems have stringent timing constraints that must be satisfied for the correct functioning, since violation might be catastrophic, such as loss of human lives. In addition, there are systems where energy is another constraint that must also be satisfied. Hence, early detection of potential problems may reduce risks of faults propagations from early specification to the final code.
This work brings an approach based on ETPN for estimating embedded software execution time, energy consumption and verification of properties in early phases of the development life-cycle. The proposed method consists on a decomposition of a UML-IO into basic elements like activities, transitions and lifeline. These elements are translated into the corresponding ETPN representations. Quantitative annotations from MARTE profile, such as time and energy data, are taken into account and included into the ETPN. The obtained model is evaluated in order to check a set of qualitative properties as well as time and energy consumption requirements. That method has been applied into a practical system, namely, a pulse-oximeter, showing that this is a promising approach for modeling, analysis, and verification of real-world case scenarios.
Future research will explore the automatic generation of ETPNs from UML-IO with MARTE annotations. This work is also being extended to cover other significant UML diagrams like sequence and state machine diagrams. Another future work is related to stressing simulation and analysis capabilities to get as much significant information as possible from the Time Petri Net models.

## 8. References

Amorim, L., Maciel, P., Nogueira, M., Barreto, R. & Tavares, E. (2005). A Methodology for Mapping Live Sequence Chart to Coloured Petri Net, *Systems, Man and Cybernetics, 2005 IEEE International Conference on* .

Barreto, R. & Lima, R. (2004). A novel approach for off-line multiprocessor scheduling in embedded hard real-time systems, *Design Methods And Applications For Distributed Embedded Systems* .

Callou, G., Maciel, P., Andrade, E., Nogueira, B. & Tavares, E. (2008). A coloured petri net based approach for estimating execution time and energy consumption in embedded systems, ACM, New York, NY, USA, pp. 134–139.

David, R. (2005). *Discrete, Continuous, And Hybrid Petri Nets*, Springer.

Elkoutbi, M., Bennani, M., Keller, R. K. & Boulmalf, M. (2002). Real-time system specifications based on uml scenarios and timed petri nets, *2nd IEEE Internationnal Symposium on signal processing and information technology, ISSPIT'02* pp. 362–366.

Júnior, M. N. O. (1998). *Desenvolvimento de Um Protótipo para a Medida Não Invasiva da Saturação Arterial de Oxigênio em Humanos - Oxímetro de Pulso (in portuguese)*, MSc Thesis, Departamento de Biofísica e Radiobiologia, Universidade Federal de Pernambuco.

Lee, J., Pan, J., Kuo, J., Fanjiang, Y. & Yang, S. (2000). Towards the verification of scenarios with time Petri-nets, *Computer Software and Applications Conference, 2000. COMPSAC 2000. The 24th Annual International* pp. 503–508.

MARTE, O. (2005). Profile for Modeling and Analysis of Real-Time and Embedded Systems, http://www.omgmarte.org/.

Merlin, P. & Faber, D. J. (1976). Recoverability of communication protocols: Implicatons of a theoretical study, *IEEE Transactions on Communications* **24**(9): 1036–1043.

Merseguer, J., Campos, J. & Mena, E. (2002). Performance evaluation for the design of agent-based systems: A Petri net approach, *Proceedings of the Workshop on Software Engineering and Petri Nets, within the 21st International Conference on Application and Theory of Petri Nets* pp. 1–20.

Starke, P. & Roch, S. (1999). *INA - Integrated Net Analyzer - Version 2.2*, Humbolt Universität zu Berlin - Institut für Informatik.

Tavares, E., Barreto, R., Maciel, P., Meuse Oliveira, J., Amorim, L., Rocha, F. & Lima, R. (2007). Software synthesis for hard real-time embedded systems with multiple processors, *SIGSOFT Softw. Eng. Notes* **32**(2): 1–10.

Tavares, E. & Maciel, P. (2006). Amalghma tool, http://www.cin.ufpe.br/∼eagt/tools/.

Trowitzsch, J. & Zimmermann, A. (2005). Real-Time UML State Machines: An Analysis Approach, *Object Oriented Software Design for Real Time and Embedded Computer Systems* .

UML, O. (2005). 2.0 Superstructure Specification, http://www.uml.org/.

# Intuitive Transformation of UML2 Activities into Fundamental Modeling Concept Petri Nets and Colored Petri Nets

Anthony Spiteri Staines
*University of Malta*
*Malta*

## 1. Introduction

### 1.1 UML 2 Activity Modeling

UML 2 activities constitute important modeling notations for specifying and modeling different types of behavior found in systems and object oriented behavior. Activity modeling is useful for understanding and defining web processing, web service composition, business process modeling, workflow modeling, systems integration, task management and low level tasks like software operations. Activity modeling is not just found in UML but also in languages like the systems modeling language (SysML) where activities are modeled using enhanced functional flow block diagrams (EFFBD), business process execution language (BPEL) and Agile. Activity modeling is not restricted to a particular use but can be successfully integrated into different approaches as required.
UML 2 activities are classified into i) fundamental, ii) basic , iii) intermediate, iv) complete, v) structured , vi) complete structured and vii) extra structured. Each class is useful for a particular problem area. E.g. structured activities address traditional programming language modeling, whilst fundamental and basic activities are ideal for high level business process modelling (Spiteri Staines, 2008).

### 1.2 Petri Nets and Activity Semantics

Activities are based on Petri Net like semantics, existing at a higher level of abstraction. They share common properties with Petri nets, although these are loosely defined in the UML 2 Superstructure specification (OMG, 2009). Rules for node execution and tokens are presented. There are also complex rules for token flows. New extensions, new rules and complex token types are mentioned. Unfortunately, the rules for activity execution are not clearly explained and defined in the UML specification. Problems can arise for concurrency, synchronization and other types of behavior. UML activities lack proper formal concepts. Activities are unsuitable for simulation, and analysis, lacking proper validation and verification techniques. The activity diagram constructed is a non executable model.
Petri nets are formalisms for describing or explaining the partial ordering of system events or actions, following a particular temporal order. In a Petri net event sequencing depends on

the system state following specific rules. Knowing the causal event ordering and current state gives predictable behavior. Petri nets seem to be the best natural choice to support activities for various reasons. Evidence in the UML Superstructure specification suggests activities to be derived in part from Petri nets. Petri net formalisms can support all activity constructs and more complex modeling. They have over three decades of coverage, and are based on sound mathematical properties. Petri nets are useful for visualization, formal verification, simulation and detailed analysis. Transforming UML 2 into Petri nets creates some new issues: i) which classes of Petri Nets are best suited for conversion, ii) how to convert, iii) visualization issues. The best classes to support activity modeling are higher order net and Colored Petri net (CPN) classes. However if activities are directly translated into CPNs, these will normally contain more nodes and edges than the actual activity diagram. One solution is to initially transform the activity into a Fundamental Modeling Concept Petri Net diagram concise notation and later construct the detailed CPN.

UML 2.2 Activity diagrams (OMG, 2009) introduce many new concepts and notations not present in UML 1.x. Some of these are collections, streams, loops and exception handling mechanisms. UML 2.2 replaces UML 1.5 'Activity graph' concepts based on state machines with activity modeling that is supposedly based on Petri net semantics (Borger et al., 2000). This is because the 'Petri net' like semantics offer advantages such as the possibility of multiple and parallel flow modelling, better control and sequencing, etc. The UML super structure specification (OMG, 2009) gives basic rules for node execution and tokens. Tokens are removed from nodes and offered to output edges. UML rules for executing an action are surprisingly similar to Petri nets, although the definition is not clear. New extensions may add new token types having their own flow rules making things more complex.

The following explain some main properties of activity diagrams:-

i) Activity diagram nodes have flow-of-control constructs like synchronization, decision, concurrency and sequence. These are fundamentally similar to those of Petri Nets.

ii) Activity diagram semantics are based on token flows (OMG, 2009). Tokens can contain objects, data, control information. Tokens are normally distinguishable through an individual time-stamp.

iii) These classes include task sequencing, data flows and control flows based on normal resources. The activity types suited to convert into Petri net models are fundamental, basic and intermediate, although conversion work can be extended to other classes.

This work will explain how UML activities can be easily transformed or converted into Petri nets, FMC-PND and CPNs. A case study of a web ordering system is given to illustrate this approach.

## 2. Transformational Mapping of Activities into FMC-PNDs and CPNs

### 2.1 Related Work

This section presents some evidence of Petri Net use for supporting UML activities which is based on available literature. Basically there are two mainstream approaches to transform UML constructs into Petri nets. The approaches are i) informal and ii) formal. Not all approaches use higher order nets, so some information loss might occur.

In (Canevet et al., 2004) a UML 2 activity model for an online multi role playing game is transformed into a PEPA net and analyzed. There is no explanation of the translation process, which seems to be informal. This indicates the usefulness of CPNs for requirements

engineering. Another method suggests mapping of UML use case constructs to CPNs based on multi layers (Shin et al., 2003). UML use cases are the starting point for activity modeling. No proper formalisms have been used. Statecharts are easily converted into OPNs (Saldhana & Shatz, 2000). Basic constructs have been mapped informally to CPNs for consistent models (King & Pooley, 2002; Shinkawa, 2006). CPNs have been successfully applied in the modeling of a real bank's advisor portal system's use cases. A complete system was constructed and implemented based on CPNs (Jørgensen & Lassen, 2006).
In (Lopez-Grao & Campos, 2004; Merseguer et al., 2006) a well structured, semi formal method is presented to translate activities into LGSPNs (labeled generalized stochastic Petri nets). These are very useful for performance analysis but the final net is a simplification of the activity diagram and includes the time dimension. LGSPNs cannot contain all the details CPNs can have, although they are very useful for performance estimation. The LaQuSo project (LaQuSo, 2007) has case tools to transform activity diagrams into simple Petri nets for soundness analysis.
UML can be formalized using Petri net like semantics. HLTPNs (higher level timed Petri nets) which are similar to CPNs (Baresi & Pezze, 2001) suggest this. In (Garrido & Gea, 2002) a CPN based formalization of the UML is presented. In (Störrle, 2004a) a simple formal transformation is presented, this idea is explained in section 4.2. Looking up literature other various examples and approaches are possible.
FMC-PND are based on the concepts found in traditional Petri nets and higher order Petri nets. Fundamental modeling concepts, as the name implies focus on practical and effectively understanding structures in the real world. FMC diagrams have been successfully applied to model various ERP projects at SAP, client server activity and even basic algorithms (Knöpfel et al., 2006). These diagrams are based on visualization concepts and pattern identification. This modelling approach is easy to apply for systems and software design. Special tools are not required.

### 2.2 Reasons for Mapping and Transforming Activities into Colored Petri Nets

Several motivating factors exists for mapping UML activity diagrams into Petri nets (Baresi & Pezze, 2001; Bock, 2005; Canevet et al., 2004; Dumas & Ter Hofstead, 2001; Ghodsi & Kent, 1991; Hu & Shatz, 2004; Jancar et al., 1999; Lopez-Grao & Campos, 2004; Petriu & Shen, 2002; Sivaraman & Kamath, 2003; Störrle, 2004a,b; Van der Aalst, 1998; Ziaei & Agha, 2003). i) UML lacks support for strong simulation and analysis techniques (Baresi & Pezze, 2001), ii) According to the OMG, UML 2 activity diagrams are based on Petri net semantics making them the best choice, iii) Petri Nets are graphical formalisms, iv) Petri nets model all UML activity constructs, v) Petri nets can be validated ,verified and simulated, vi) Petri nets have sound mathematical properties, vii) Petri nets are suitable for visualization, aesthetic representation of systems, etc. (Knöpfel et al., 2006, Kristensen et al., 1998, 2004; Van Hee, 2005).
UML activities can also be transformed into other formalisms such as process algebras, formal languages, logics, CCS, etc. In (Störrle & Hausmann, 2005) it is explained that intuitive mapping from UML 2 activities into other formal notations such as algebras, CCS or other transition systems does not exist. There is considerable separation between the actual syntax and semantics. Most formalisms are non graphical and do not support visualization.

Higher order Petri nets (Jancar et al., 1999; Störrle, 2004; Störrle & Hausmann, 2005) are best suited to activity modeling. Activities or actions are defined using token types, objects, sets, colored sets and advanced data types. Combining arc inscriptions with transitions and special languages expand the modeling capabilities. Colored Petri Nets (CPNs) (CPNTools ,2009; Jørgensen ,2002;Kristensen et al. ,1998; Kristensen, 2004) based on languages like ML are good for activity modeling. The ML and CPN ML languages can completely imitate the behavior of activity diagrams ranging from streaming /non streaming activities, multiple control flows, control buffers and pins, timing issues, etc. Some of these are part of the SysML language (Bock,2005). CPNs show a close correspondence to activities and offer enhanced modeling even though other Petri net classes are still useful.
However problems are created when transforming complex activity diagrams directly into CPNs and other high level Petri net classes as is evidenced in (Dumas & Ter Hofstead, 2001; Hu & Shatz, 2004; Jancar et al, 1999; Petriu & Shen, 2002; Sivaraman & Kamath,2003; Störrle, 2004 a,b; Störrle & Hausmann, 2005; Van der Aalst,1998). Some are listed below:

- Constructing the CPN is a long and tedious task requiring expert knowledge.
- The CPN might end up have substantially more nodes and edges than the UML diagram. This is the case if a one-to-one mapping between the activity diagram and the CPN is maintained. Additional places need to be added for concurrency and other activities.
- Visually the constructed CPN might be difficult to read and understand, i.e. it is unsuitable for visualization.
- Other constructs like sets, data types, functions, etc. must be defined and programmed to make the CPN operational. These are not available directly from the activity diagram.

It is important to find solutions for these issues. CPNs are the best suited Petri net class for modeling activity diagrams. Unfortunately they can become complicated for stakeholders to comprehend (Dori,2003; Kaindl & Carroll,1999; Soderborg et al., 2003). A solution should simplify the comprehension whilst retaining the fundamental features. Models must be clear, simple and precise for proper comprehension by stakeholders during the analysis phase.

### 2.3 A Fundamental Modeling Concept Petri Net and Colored Petri Net Solution

An intuitive mapping or transformation of UML activities into Petri nets and vice-versa is possible. This is because UML 2 activities are based on Petri net concepts and add other details.
The fundamental modeling concept (FMC) idea (Knöpfel et al.,2006) originated at the HPI Institute, Potsdam, Germany. FMC present visualization guidelines to creating models that are more comprehensible, compact and easier to construct. The idea in this work is to apply the principles of FMC to construct a comprehensible Petri net or CPN model from the UML activity. To this end reduced FMC-PND are suggested. FMC-PND can serve to represent repeated patterns with emphasis on the layout of the notational elements clearly explaining what is happening.
The proposed solution is to i) identify the activity diagram, ii) transform it into Fundamental Modeling Concept Petri net diagram (FMC-PND) and iii) construct a CPN. Steps ii) and iii) can be carried out concurrently.

FMC focus on system related structures, system behavior at a high level and performance. UML 2 activity diagrams focus on a wide spectrum of issues complicating things. UML activities are more suited to model software behavior. FMC-PND (Knöpfel et al.,2006) offer several advantages over creating an immediate CPN simplifying the UML activity diagram. A FMC-PND (Knöpfel et al.,2006) is readable and simpler to create than a CPN. At this point, understanding the system and identifying patterns are more important than execution.

- FMC are based on i) abstraction, ii) simplicity, iii) universality, iv) separation of concerns, v) aesthetics and secondary notation.
- FMC are visually attractive.
- FMC are based on specific principles like diagram size, node arrangement, node shape, enumeration, harmonization, and proper labeling making the diagrams easier to comprehend.
- FMC are based on pattern identification. Repeated patterns can be simplified or reduced.
- FMC present a way to abstract process representation and to study the activities involved and reengineer the system.
- The complete work behind FMC is documented in (Knöpfel et al.,2006). FMC model systems on the static and dynamic aspect, they are based on proven techniques and have been used to design interactive software systems (Gröne et al., 2003, 2004, 2006; Tabeling, 2002, 2004; Tabeling & Gröne, 2005).

The FMC-PND is just a block diagram or visual representation. The aesthetical layout of the model places strict emphasis on the layout of notational elements, nodes and edges to present better graphical structures and patterns. These are suitable for comprehension by different system stakeholders. On the other hand for proper execution, analysis and performance estimation, the activity should be converted to a CPN.

## 3. Transformation Approach

### 3.1 Activities and Corresponding Petri Net Notations

Referring to previous work presented in (Störrle, 2004 a,b; Störrle & Hausmann, 2005), the activity diagram specification in (OMG, 2009) and working experience with Petri nets, it is possible to find and define combinations for converting or transforming activity semantics into their Petri net counterpart.

Normal activity edges are converted into Petri net places with input and output arcs. There are exceptions when the activity edge links: i) merge node to merge node, ii) start node to merge node, iii) merge node to activity final node, iv) merge node to flow final node. These convert to a Petri net transition with input and output arcs. Other activity edge exceptions exist, i) edges outgoing from a start node, ii) activity final incoming edge and iii) flow final incoming edge, here these convert into a Petri net arc or flow. Conversion for object nodes and signal nodes are included in a separate table.

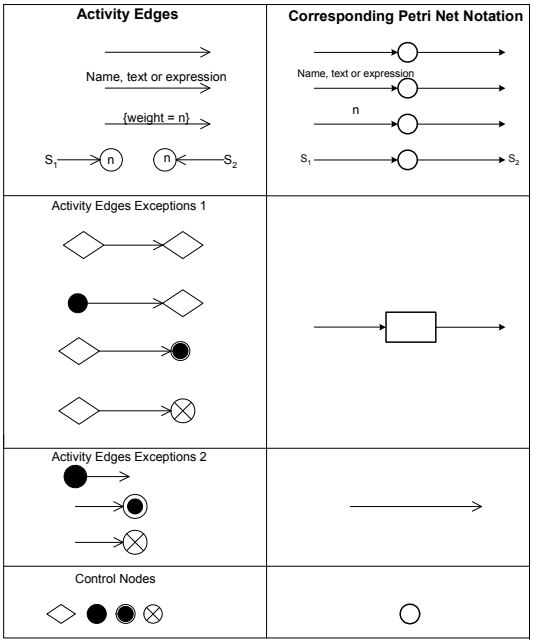



Table 1. Activity Edges and Control Nodes corresponding Petri Net notations

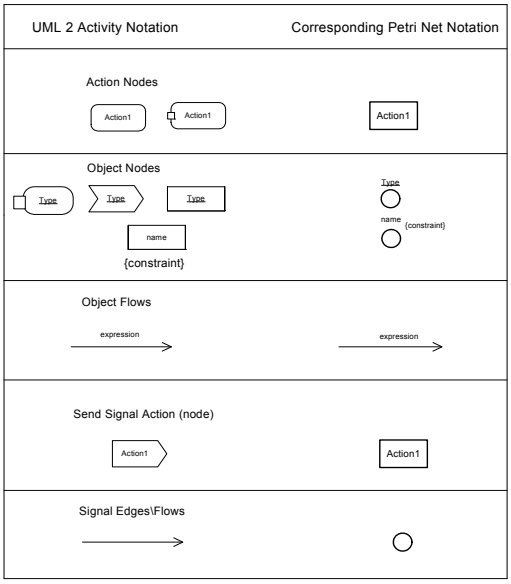



Table 2. Other Activity Notations with corresponding Petri Net notations

The notations shown in Table 1 and 2 summarize the main notations that are used in UML 2 activity models. Table 1 and 2 explain this conversion process taking into consideration many of the constructs found in UML 2 activities, although there are exceptions. From table 2 it is evident that the conversion is similar to that of normal activities and edges except for object flows. Normally activity diagrams would be made up using the main notations.
The UML superstructure specification defines a variety of presentation options that can be used in conjunction with these notations. The notations and their depiction vary, depending on the type of activity that is being modeled. It is possible to annotate these notations to show more detail. Changing the shape or way a notation is depicted will have no effect on the corresponding Petri net.
Table 3 shows signal node exceptions. There is an accept signal, with no incoming edges. This type of signal is always enabled, hence it initiates an activity or series of actions. The other type of signal node is a repetitive time event. It means that this event occurs at regular time intervals. In these cases a Petri net token generator fits closely with this behavior. Alternatively, a single place, with tokens always available can be used.

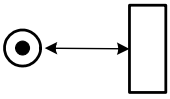
| Signal Node Exceptions | Corresponding Petri Net |
|---|---|
| Accept Signal (no incoming edges) always enabled | Token Generator |
| Repetitive Time Event |  Transition is always enabled |

Table 3. Signal Node Exceptions and Corresponding Petri Net

### 3.2 Conversion of Enhanced Activity Structures

An expansion region is convertible into a transition. The transition can be decomposed further to sub levels.

i. Object node pins are just an alternative style for display object nodes. Hence they do not affect the conversion process.
ii. Parameter sets are used to direct token flows. Petri net places can be used for stand alone pins.
iii. Parameter nodes are similar to object nodes. They have no significant effect on the resultant Petri net. They are useful for partitioning, thus places could be used for parameter nodes, just as is done for object nodes.

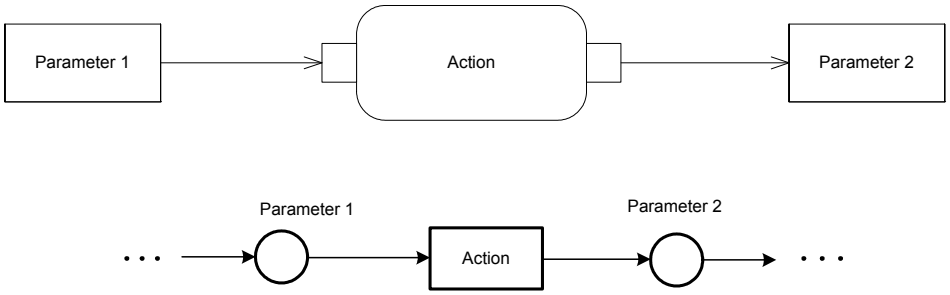



Fig. 1. Activity with Parameter Nodes and Corresponding Petri Net

iv. Partitioning and swim lanes of the activity model have no effect whatsoever on the Petri net model.

v. For intermediate activities, special types of nodes are possible. E.g. <<central buffer>>, <<data store>>. Some of these can be represented using a single place. They can be decomposed to a full Petri net model with detailed behavior .

### 3.3 FMC-PND Basic Patterns

When examining activity models, four or five basic patterns well known to workflow nets and BPEL (Van der Aalst, 1998), generalized task graphs (GTG) (Ghodsi & Kent, 1991), task graph mappings (Chen et al., 1995) and FMC can easily be identified.

The patterns are classified as: i) sequence, ii) and/split join, iii) xor split, ,iv) xor join and v) iteration. Having FMC principles of abstraction, simplicity, universality, separation of concerns, aesthetics and secondary notations (Knöpfel et al., 2006) a simplified, neater and comprehensible Petri net can be drawn.

Fig. 2-6 illustrate from left to right the: i) activity diagram pattern, ii) Petri net and iii) reduced FMC-PND. From previous work presented in (Störrle, 2004 a,b; Störrle & Hausmann,2005) the activity diagram specification and practical experience with Petri nets, combinations to convert or transform activity semantics into their Petri net counterpart are easily discovered.

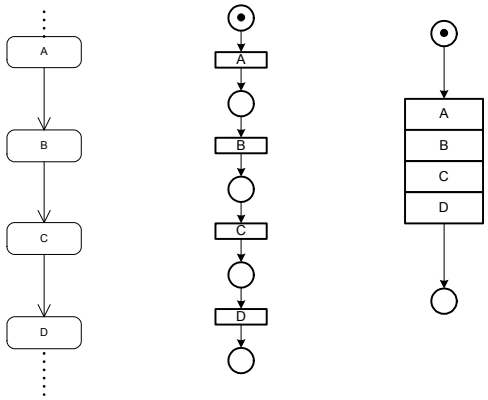



Fig. 2. Activity Sequence

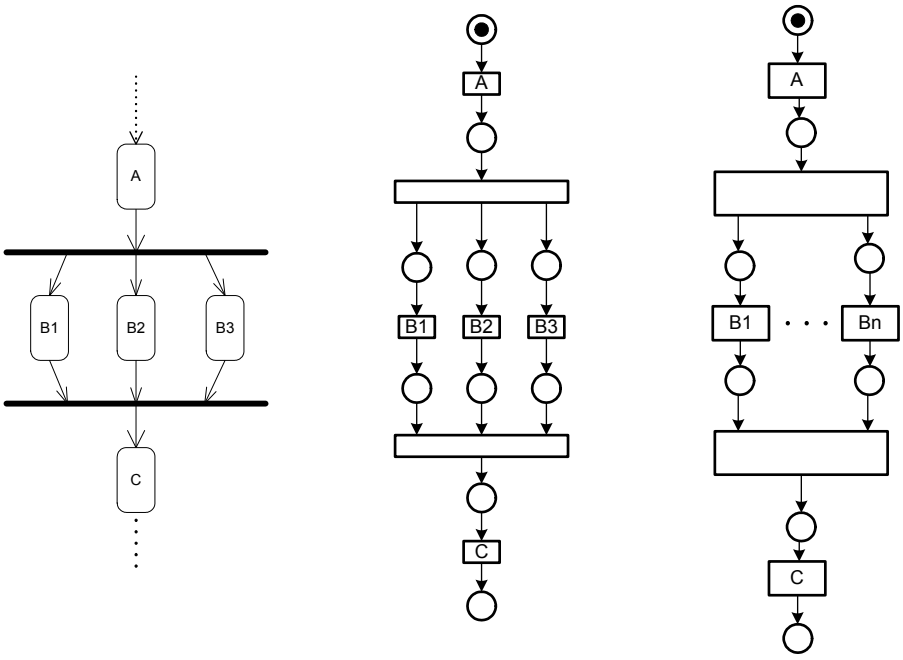



Fig. 3. Activity Fork and Join

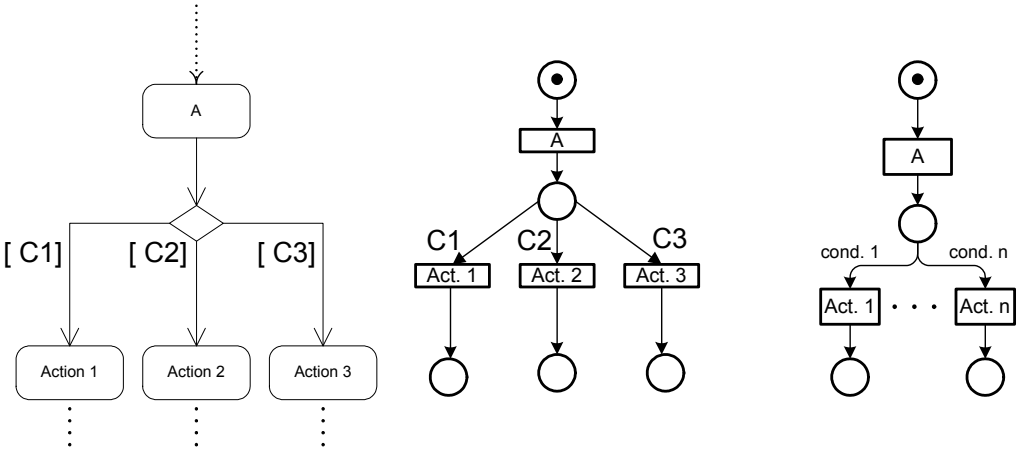



Fig. 4. Activity Decision/Choice Node

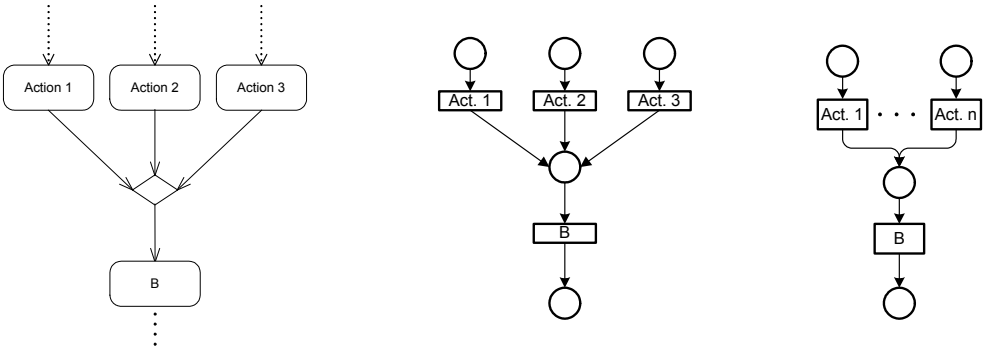



Fig. 5. Activity Merge Node

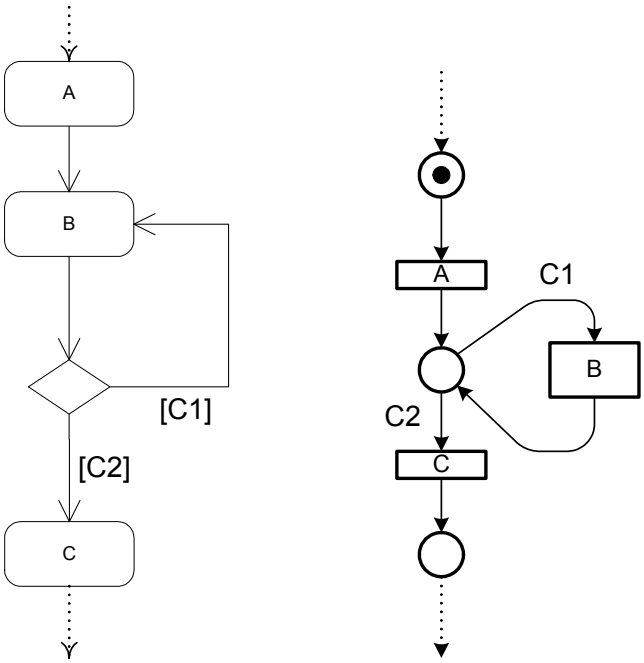



Fig. 6. Activity Iteration

The UML 2 activity semantics cater for special cases. Advanced constructs like buffers, client server communication, etc. can be used. These can also be modelled using FMC patterns. Fig. 7 shows an exception handler not found in UML 1.x. The complete Petri net for the exception handler is quite long. The reduced FMC Petri net simplifies this, showing some similarity to the activity diagram. Swimlanes, expansion regions, pin notations, parameter sets, interruptible regions similar to exception handling in fig. 7 can be modeled with FMC and CPNs.

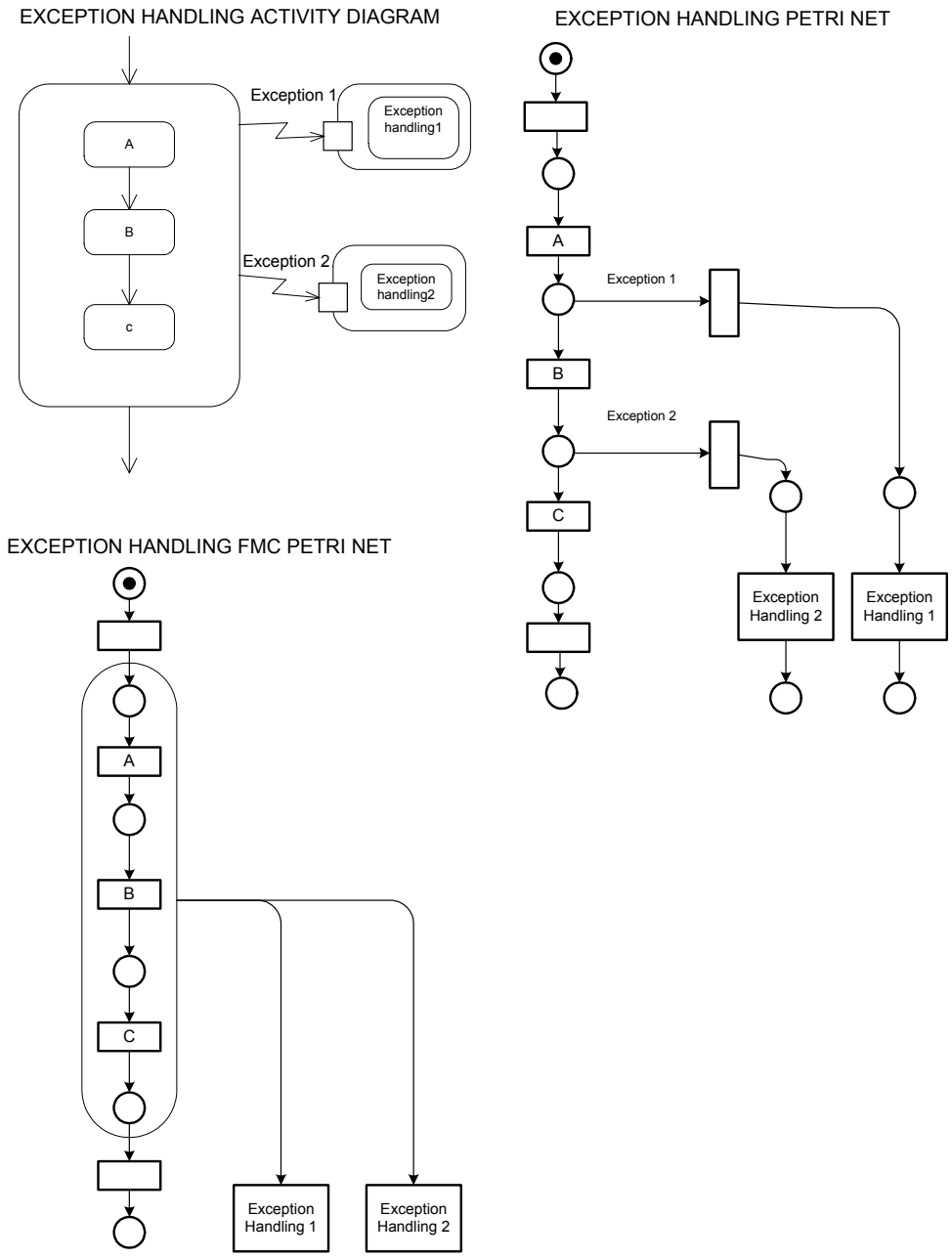



Fig. 7. Exception Handling

## 4. Transformational Approach for Activity to Colored Petri Net

### 4.1 Simple Conversion

Conversion from the activity model to the CPN can be done simply by looking up tables 1-3. The algorithm outline proposed for this is:

*If normal (edge or node) lookup corresponding petri net notation and replace*
*Else lookup exception and replace*

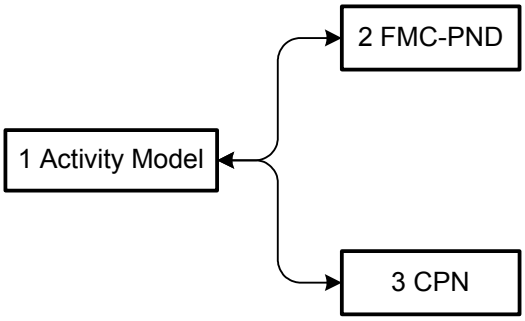



Fig. 8. Model Mapping Informal or Formal

Fig. 8 shows how the activity model could be used for the FMC-PND and the CPN. The arrows connecting the models are bi-directional indicating that a change to one of the models must be reflected in the others. The FMC-PND is of greater interest to the shareholders, it is concise and more readable. On the other hand the CPN has more detail. It is suitable for execution and validation. The FMC-PND focuses on activities, ignoring boxes, input pins and other UML constructs, etc. The main patterns listed in 3.3 should be identified and the resultant Petri net is simplified. Using FMC principles the Petri net is more readable and easy to visualize.

### 4.2 Formal Mapping and Transformation

To obtain a fully functional CPN model, the input and output arcs have to be defined from an existing color set. Places must have corresponding color. Normally this information is not available from the UML activity. E.g. Places have type definitions. Arcs might be required to have functions. This information is not available from the activity which is normally a non executable model. The arc expression function, the initialization function, etc. are not directly obtainable. The transformation of UML 2 activities into a CPN has been formalized (Störrle, 2004b, Garrido & Gea, 2002). In (Störrle, 2004b) a semantic function is to used map activity nodes or edges into a CPN. The function converts an activity diagram <activity node, activity edge> into a CPN defined as <N, SigAlg, color, guard, effect>, where N=<P,T,A>. A set of transformation mappings for UML into CPN can be defined. The FMC-PND are based on templates. The CPNtool (CPNTools, 2009) uses XML format.

The CPN can be defined as nine tuple set =(∑,P,T,A,N,C,G,E,I), where ∑ is a finite set of non-empty types or color sets. P is a finite set of places. T is a finite set of transitions. A is a finite set of arcs. N is the node function. C is a color function. G is a guard function. E is an arc expression function. I is an initialization function. The information below explains in

simple terms the transformation of activity constructs into CPN constructs. The activity constructs can be stored in a list having <activity nodes, activity edges>. The information below can be used to construct a semantic function for the transformation.

i. Initial node, final node, object nodes, control nodes, signal edges and activity edges excluding exceptions all transform or correspond to places
ii. Action nodes, control nodes , signal nodes, activity edge exceptions1 transform or correspond to transitions
iii. Object flows, activity edge exceptions 2, activity edges transform or correspond to arcs connecting to input flows and output flows
iv. Object node types, object types, control node types, parameters, etc. correspond with the color type of the CPN places
v. Expressions on activity edges, object flows correspond to expressions or arc inscriptions on the corresponding CPN arcs.
vi. Guards at merge, fork or joining nodes convert to expressions on the corresponding arcs based on the arc inscription language.

### 4.3 Triple Graph Grammer Correspondence

The activity diagram data could be mapped directly into a CPN using a model-to-model mapping approach based on the OMG QVT (Query View Transform) or ATL (Atlas Transformation Language). According to (Lohmann et al., 2007) Triple Graph Grammars (TGGs) are better to express workflow patterns of UML activities as we have graph-to-graph mappings. TGGs seem to offer the best choice for mapping activity models into CPNs and FMC-PNDs. The concepts behind the QVT of the OMG standard are surprisingly similar to TGG. The idea behind TGG is to keep a transformational correspondence between two different models (Kindler & Wagner, 2007).

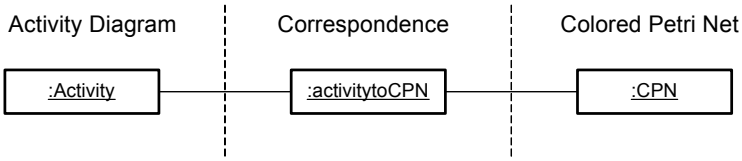



Fig. 9. Triple Graph Grammar High level Correspondence between Activity and CPN

Figure 9 shows high level correspondence between the activity model and a CPN using TGGs. The correspondence will define all the transformations or mapping rules from the activity into the CPN and vice-versa. These rules can be built using the information given previously. New rules can be added to ensure that the Petri net is complete, etc. Rules for reduction for identifiable patterns can be included.

The transformations can be forward or backward. E.g. if changes are done to the activity these are automatically reflected in the CPN. The opposite is also true, i.e. certain changes to the CPN result in automatic updates in the activity model. This approach can even include correspondence with the FMC-PND. Literature is available about TGGs in some publications.

## 5. Case Study: An Activity Model for Online Ordering

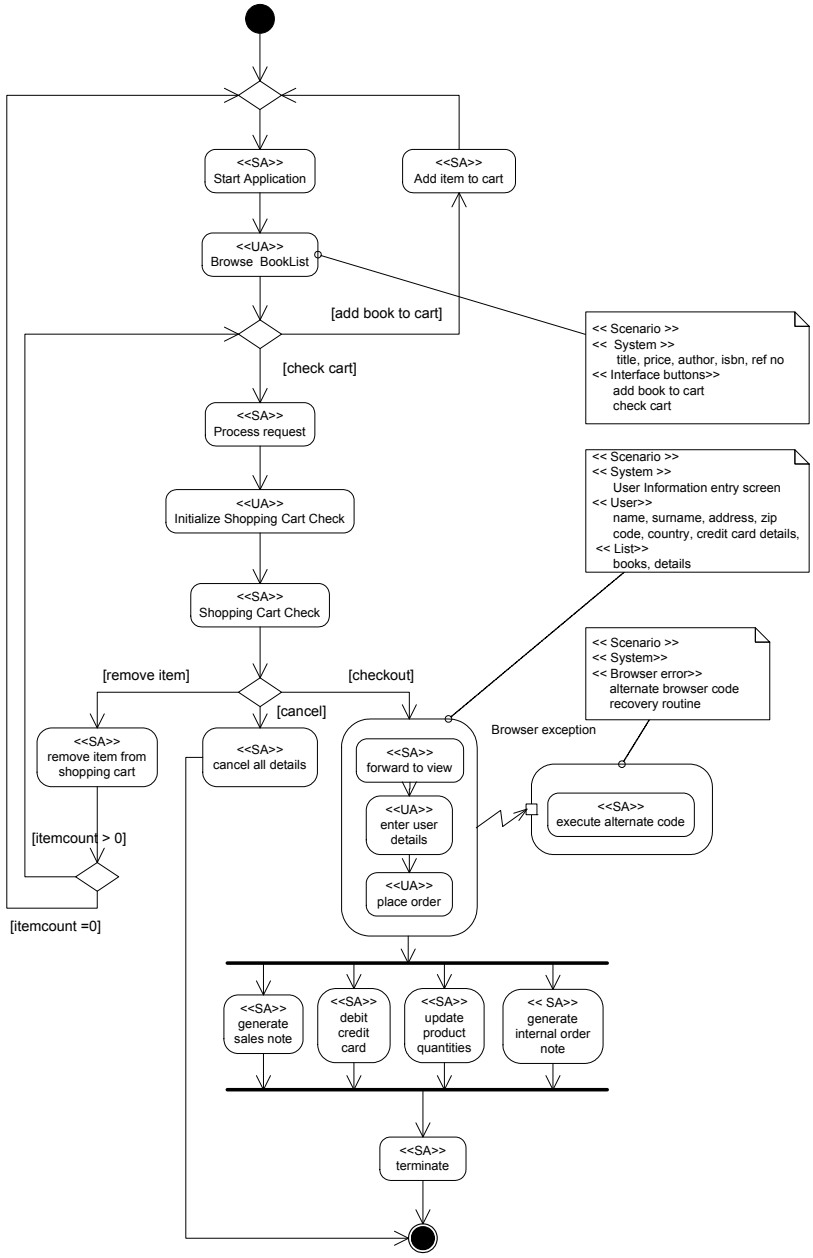



Fig. 10. Activity for An Online Book Ordering System

An activity model of an online book ordering system is shown in fig. 10. The diagram is based on an approach in (Lorenz & Six, 2006) where user oriented and interaction oriented activities are combined into a specification oriented diagram. This activity can be mapped to a target platform like J2EE with web tier, struts, BPEL, etc., implementing the programming code directly.
The activity model combines three different views. These are i) user oriented, ii) interaction oriented and iii) specification oriented. The specification oriented view is the final result. Most of the activities are "system activities" and hence stereotyped <<SA>> along with "user activities" stereotyped << UA>>. Ordering involves adding books to a cart. A user can include new books. At the end the user can i) cancel, ii) remove books and continue adding or iii) checkout. Checkout then proceeds to placing an order where the name, address, credit card details, etc. are required. Then a sales note, internal note and debit note are generated. The product quantities are updated. An exception handler checks for browser errors. When an error occurs alternative code is executed. This activity includes three possible iterations. There are two termination options either i) cancel or ii) checkout. Notes are added for better comprehension in this example. The diagram is becoming more complex to interpret. For non-technical stakeholders there is a comprehension problem.

### 5.1 Fundamental Modeling Concept- Petri Net for Online Ordering

Using the activity in fig.10 and the information given in section 3.3 it is possible to create the FMC-PND. The model follows the FMC principles listed in 2.3. These are explained in (Knöpfel et al., 2006).
Compact representation can be used for sequences, error alternatives, choice and merge. A simpler model which is easier to read is produced. This can be seen from the diagram shown in fig. 11. Interpretation is simpler. The important processes for ordering are clearly highlighted. The activity model combines three different views. These are i) user oriented, ii) interaction oriented and iii) specification oriented. The specification oriented view is the final result. Most of the activities are "system activities" and hence stereotyped <<SA>> along with "user activities".

### 5.2 Colored Petri Net for Online Ordering

If the activity model is structurally correct it is possible to construct the CPN. The CPN is similar to the FMC-PND but contains more places and transitions. These have to be added to model precisely the UML 2 activity in fig. 10. Additionally the CPN construction is necessary to build an executable model for testing and verification. The most adequate tools to construct the CPN are CPNTools (Kristensen et al. 1998, CPNTools, 2009).
The final CPN is shown in fig. 12. This shows the online book ordering activity. It is a fully working model containing detailed processing logic, bringing the model as close as possible to the real scenario. The complete executable specification was created for the book ordering system. The payment and credit card details can be entered as a string. Random generators are used to generate random book order numbers; price and random errors are generated as required. The programming needed for the model was done using CPN ML. The functions used are listed in fig. 12. Extensive code that has not been presented here was needed for perfect execution. If problems turn up during execution these are seen visually and they can be immediately amended in all the models.

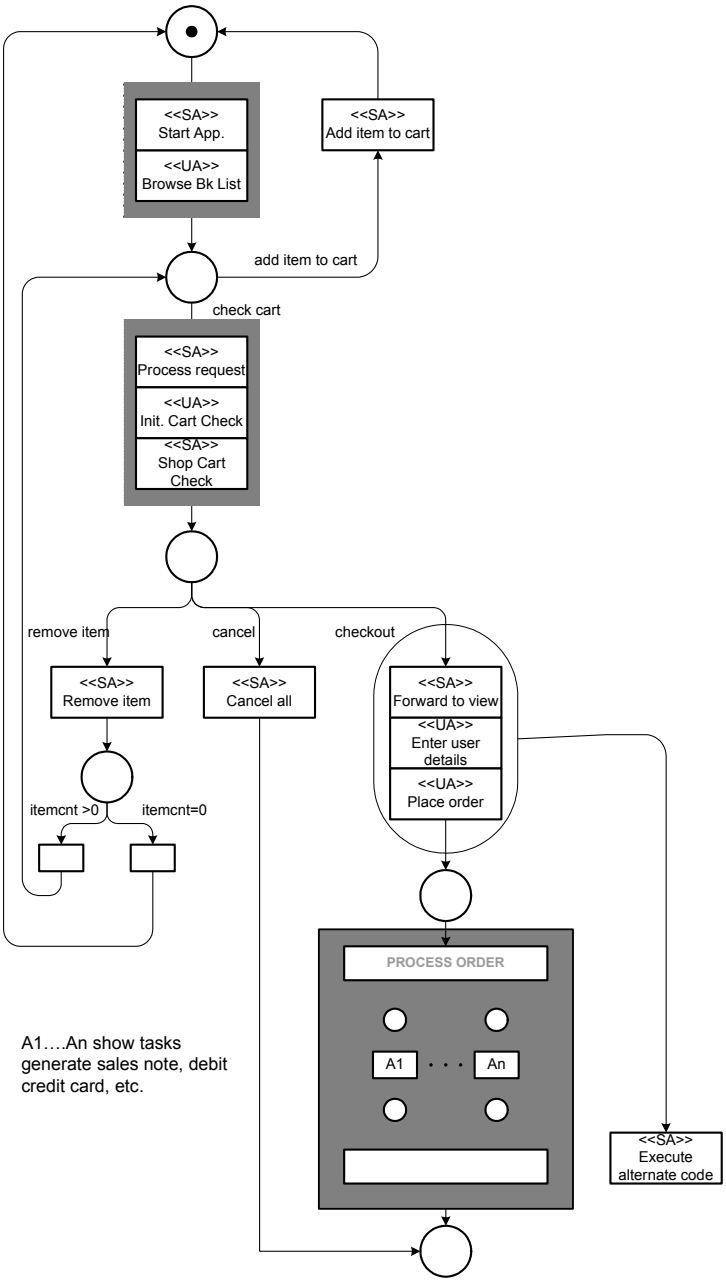



Fig. 11. Fundamental Modeling Concept Petri Net For Online Book Ordering

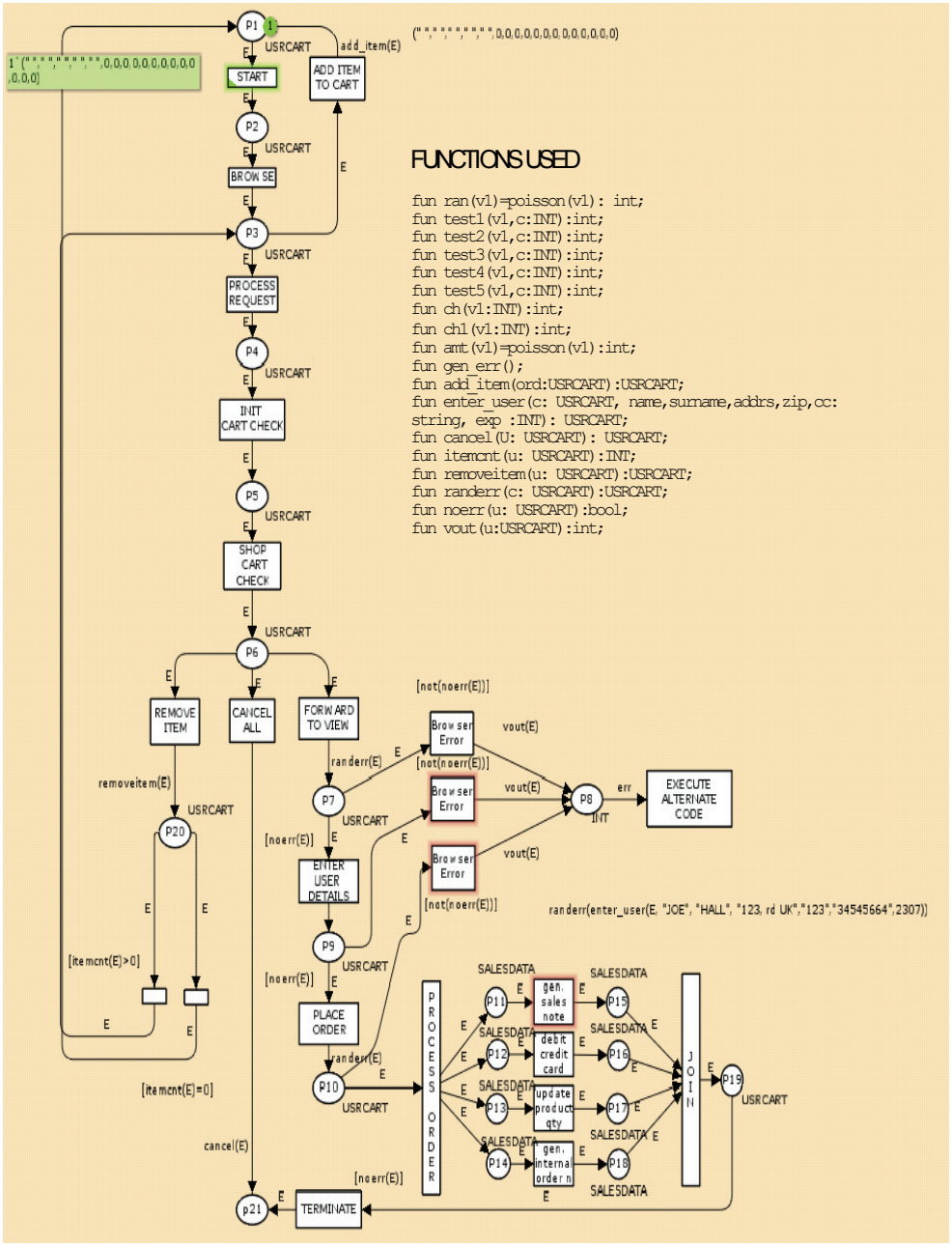



Fig. 12. Colored Petri Net For Online Book Ordering

## 6. Brief Analysis of the Modeling Approaches

In activity modeling, processes should have a clear start and controlled termination (Van Hee, 1995,2005). UML activities, model processes as abstract entities. These are converted into well defined executable processes in the CPN. The FMC-PND helps to properly comprehend the CPN model. The CPN is validated and tested using the CPNTools. Once the models are finalized they can be used to define an executable specification.
Activity modeling covers a variety of system aspects. Three possible checks that can be carried out to a system (Van Hee, 2005) are identifiable. These are i) reviews, ii) proofs and iii) tests. The activity diagram and the FMC-PND are suitable for i). The CPN is suitable for ii) and iii). The FMC-PND can be used for conformance analysis based on balanced and /or splits or joins, conflict issues, well structured and well handled nets principles (Van der Aalst, 1998). The CPN is suited for performance analysis and can be analyzed using classic Petri net theory like linear algebra, place invariants, transition invariants, non reachable conditions, soundness, reachability tree, model checking, reduction techniques, markov chain analysis, etc. It can also be checked for deadlock and livelock thus eliminating logically incorrect specifications.
The FMC-PND is definitely more simple, understandable and readable than the CPN or other higher level nets, having just 9 places (see fig. 11) compared to the 21 places in the CPN (see fig. 12). This implies that there are substantially less nodes and edges in the FMC-PND. The resultant diagrams in fig. 11 and 12 are cyclical and not choice free, i.e. there are some conflicting transitions. Some parts of the net are strongly connected and other parts loosely connected. The patterns presented in the FMC-PND can be used to reduce the complexity in the activity diagram.

| Model Type | Pattern Based | Suitability for Visualization | Suitability for Execution | Testing and Verification |
|---|---|---|---|---|
| UML ACTIVITY | No | Moderate | n/a | possible |
| FMC-PND | Yes | Very good | n/a | possible |
| CPN | No | Moderate | YES | possible |

Table 4. Common Properties for: UML Activity, FMC-PND and CPN

Table 4 indicates the main features of both Petri net types. FMC-PNDs are better suited for stakeholders. The idea of translating UML 2 activity diagrams into FMC-PND simplifies many of the difficulties encountered in transforming UML activities into other classes of Petri nets. From experience, basic patterns are useful in over 90 % of all scenarios. FMC-PND cover other types of patterns that can be extended to this work e.g. producer/consumer, recursion, inter task, etc.

## 7. Conclusion

This topic has dealt with UML activities, FMC-PND and CPNs. There is evidence suggesting intuitive, informal or formal mapping of UML activities into their Petri net counterparts. The models or examples given in the UML super structure specification are all transformable into CPNs. Certain classes of UML activities follow closely the behavior of Petri nets. This topic is supported with a vast amount of literature. UML 2 activities are

based on Petri net semantics. They can be suitably represented using reduced FMC-PND which are more comprehensible than a complete CPN. The FMC-PND has to be converted into a CPN for more detailed verification and simulation. CPNs are best suited for detailed modelling and execution of activity diagrams. Other constructs like pins, complex nodes, hierarchy can also be included. CPN modeling is suitable for i) fundamental, ii) basic and iii) intermediate activities of UML, although this can be extended to other classes. Activity diagrams and their semantics are important for model driven approaches. Model driven approaches used for requirements engineering, offer better choices than code driven approaches. When modelling business processes, workflows, complex software and information systems, model driven approaches are preferable. The FMC-PNDs are suitable for representing repeated patterns in activity models using a concise notation that is simpler to comprehend and visualize.

CPN and higher order net semantics are richer and more expressive than activity semantics. This implies that even if direct one-to-one mapping is done using TGG (triple graph grammars) or the OMG QVT, many additional details like type definitions, token types, place types, firing rules, functions for transitions and arcs, etc. must be added to make the CPN a proper detailed executable model. Functions, in particular, require programming using the ML language. CPN's seem to be the best natural choice for modeling FMC-PNDs and UML activities. The CPN is suitable for detailed stepwise execution and optimization. If the CPN is mapped to the activity using TGG correspondence any change to the CPN's structure will automatically update the activity model accordingly. Petri nets and Colored Petri nets are translatable into formal languages and can be expressed using other notations. Activity diagrams can be formalized to a certain extent using the UML's OCL (object constraint language).

As previously explained, the approaches to transform activities into Petri net models can be i) formal or ii) informal. Different classes of Petri nets can be used for different needs. To capture all the detail from the activity model, CPNs are definitely the best choice. However on some occasions a simple place transition Petri net could have its valid use. Petri nets offer us visual formalisms useful for examining system properties. A reduced Petri net like the FMC-PND is suitable for better comprehension.

## 8. References

Baresi, L. & Pezze, M. (2001). Improving UML with Petri Nets, *Electronic notes in Theoretical Computer Science*, pp. 107-119, UNIGRA, doi: 10.1016/ S1571-(04)80947-2 Vol 44., No. 2, Jul 2007, Elsevier

Bock, C. (2005). SysML and UML 2 support for Activity Modeling, *International Journal Council of Systems Engineering*, Vol. 9 No. 2, Nov 2005, pp. 160-186, *Wiley-InterScience DOI 10.1002/sys.20046* www.interscience.wiley.com

Borger, E.; Cavara, A. & Riccobene, E. (2000). An ASM Semantics for UML Activity Diagrams, *Proc. of 8th International Conference on Algebraic Methodology and Software Technology*, pp. 293 – 308, ISBN: 3-540-67530-2, Iowa City, May 2000, Springer-Verlag, UK

Canevet, C.; Gilmore, S. , Hilliston, J., Kloul, L. & Stevens, P. (2004). Analysing UML 2.0 Activity Diagrams in the Software Engineering Performance Process, Proc. of

WOSP'04,pp. 77-78, ISBN~ ISSN: 0163-5948,1-58113-673-0, Redwood City CA, Jan 2004, ACM

Chen, S.; Eshaghian, M. & Wu, Y.C. (1995). Mapping Arbitrary Non-Uniform Task Graphs onto Arbitrary Non-Uniform System Graphs, *Proceedings of the 1995 International Conference on Parallel Processing*, pp. 191-195, ISBN: 0-8493-2616-8, Urbana-Champain, Illinois, USA, Aug 1995, CRC Press, USA

CPNTools (2009). CPN Group, Department of Computer Science, University of Aarhus, Denmark http://www.daimi.au.dk/CPnets/

Dori, D. (2003). Conceptual Modeling and System Architecture, Communications of the ACM, Vol 46 No 10, Oct 2003, pp. 62-65, ISSN: 0001-0782

Dumas, M. & Ter Hofstead, A.H.M. (2001). UML Activity Diagrams as a Workflow Specification Language, *Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, pp. 76-90, ISBN: 978-3-540-42667-7, Toronto, Oct 2001, Springer-Verlag, Berlin

Garrido, J.L. & Gea, M. (2002). A Colored Petri Net Formalization for a UML-based Notation Applied to Cooperative System Modeling, Interactive Systems: Design, Specification and Verification, *LNCS 2545*, pp.16-28, Springer-Verlag, ISBN: 978-3-540-00266-6, Berlin/Hiedelberg

Ghodsi, M. & Kent, K. (1991). Well-formed Generalized Task Graphs, *Proceedings of the 3rd IEEE Symposium on Parallel and Distributed Processing*, pp.344-351, ISBN: 0-8186-2310-1, Dallas TX, Feb 1991, IEEE, USA

Gröne, B. & Tabeling, P. (2003). A System of Conceptual Architecture Patterns for Concurrent Request Processing Servers, *Proc. of 2nd Nordic Conference on Pattern Languages of Programs VikingPLOP03*, Bergen, Norway, Oct 2003

Gröne, B.; Knöpfel, A., Kugel, R. & Schmidt, O. (2004). The Apache Modeling Project, *Technical Report 5*, *Hasso-Plattner-Institute*, Potsdam, 2004. http://www.f-m-c.org

Gröne, B. (2006). Conceptual Patterns, *Proc. of 13th IEEE International Symposium and Workshop on Engineering of Computer Based Systems*, pp. 241-246, ISBN: 0-7695-2546-6, ECBS, Potsdam, Germany, Mar 2006, IEEE,USA

Hu, Z. & Shatz, S.M. (2004). Mapping UML Diagrams into a Petri Net Notation for System Simulation, *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2004)*,pp, 213-219, ISBN: 1-891706-14-4, Illinios Chicago, Jun 2004, Knowledge Systems Institute Grad. School, Banff, Alberta, Canada

Jancăr, P.; Esparza, J. & Moller F. (1999). Petri Nets & Regular Processes, *Journal of Computer and System Sciences* , Vol. 59, No. 3, Dec 1999, pp. 476-503, ISSN: 0022-0000

Jørgensen, J.B. (2002). Coloured Petri Nets in UML-Based Software Development – Designing Middleware for Pervasive Healthcare, *Proc. 4thInt. Workshop on the Practical use of CPN & CPN Tools* Aug 2002, pp. 61-80, Diami Technical Report PB-560, Aarhus Denmark

Jørgensen, J.B. & Lassen, K.B. (2006). Requirements Engineering for the Adviser Portal Bank System, *Proc. of 13th IEEE International Symposium and Workshop on Engineering of Computer Based Systems ECBS*, pp. 259-268, ISBN: 0-7695-2125-8, Potsdam, Germany, Mar 2006,IEEE, N.Y.

Kaindl, J.H. & Carroll, J.M. (1999). Symbolic Modeling in Practice, *Communications of the ACM*, Vol. 42 No. 1, Jan 1999, pp. 28-30, ISSN: 0001-0782

Kindler, E. & Wagner, R. (2007). Triple Graph Grammers: Concepts, Extensions, Implementations and Application Scenarios, *Technical Report Tr-ri-284*, University of Paderborn, Paderborn, Germany, 2007

King, P. & Pooley, R. (2002). Derivation of Petri Net Models from UML Specifications of Communication Software, *Proc. of 11th Int. Conf. On Tools and Techniques for Computer Performance Eval*, pp. 262-276, ISBN: 3-540-67260-5, Illinois USA, Mar 2002, Springer-Verlag, London, UK

Knöpfel, A.; Gröne, B. & Tabeling, P. (2006). *Fundamental Modeling Concepts*, Wiley, ISBN-13:978-0-470-02710-3, West Sussex, UK

Kristensen, L.M.; Christensen, S. & Jensen, K. (1998). The Practioner's Guide to Coloured Petri Nets, *International Journal On Software Tools for Tech. Transfer (STTT)*, Vol. 2, No.2, Dec 1998, pp. 98-132, ISSN: 1433-2779, 1433-2787

Kristensen, L.M.; Jørgensen, J.B. & Jensen K. (2004). Application of Coloured Petri Nets in System Development, *Lecture Notes in Computer Science*, 2004, Vol. 3098/2004, pp. 99-27, ISSN: 0302-9743, 1611-3349

LaQuSo (2007). LaQuSo Work Group / Project, LaQuSo Repository, Eindhoven, www.Laquso.com

Lohmann, C., Greenyer, J. , Jiang, J. & Systä, T. (2007) Applying Triple Graph Grammars For Pattern-Based Workflow Model Transformations, *Journal of Object Technology, Special Issue:Tools*, pp. 253-273, Oct 2007, Europe,
http://www.jot.fm/issues/issue_2007_10/paper13/

Lopez-Grao, J.P. & Campos J. (2004). From UML Activity Diagrams to Stochastic Petri Nets: Application to Software Performance Engineering, *ACM Sigsoft Software Engineering Notes*, Vol. 29, No. 3, -36, 2004, pp. 25-36, ISSN: 0163-5948

Lorenz, A. & Six, H.W. (2006). Tailoring UML Activities to Use Case Modeling for Web Application Development , *Proceedings of the conference of the Centre for Advanced Studies on Collaborative Research*, pp. 333-338 , Toronto, Oct 2006, IBM, Toronto

Merseguer, J.; Campos, J & Mena, E. (2006). On the Integration of UML and Petri Nets in Software Development, *Proc. of ICATPN'06 LNCS 4024*, pp.19-36, ISSN: 0302-9743, Turku Finland, Jun 2006, Springer, Berlin

OMG (2009).UML2 Superstructure Specification. V2.2, OMG, http://www.omg.org/technology/documents/formal/uml.htm

Petriu, D.C. & Shen, H. (2002). Applying the UML Performance Profile: Graph Grammer based derivation of LQN models from UML Specifications, *Proceedings of the 12th International Conference on Computer Performance Evaluation Modelling Techniques and Tools*, pp. 159-177, ISBN: 3-540-43539-5, London, 2002, Springer-Verlag, UK

Saldhana, J.A. & Shatz, S.M. (2000). UML Diagrams to Object Petri Net Models: An Approach for Modeling & Analysis, *SEKE 2000*, pp. 103-110, ISBN: 1891706955/9781891706059, Chicago Illinois, Jul 2000, Knowledge Systems Institute, USA

Shin, M.E.; Levis, A.H. & Wangenhals, L.W. (2003). Transformation of UML-Based System Model to Design/CPN model for Validating System Behavior, *Proc. of the 6th Int. Conf. on the UML/Workshop on Compositional Verification of the UML Models*, pp. 3-22, doi: 10.1016/j.entcs.2005.12.059, Electronic Notes in Theoretical Computer Science, Vol. 159, May 2006, Elsevier Science.

Shinkawa, Y. (2006). Inter-Model Consistency in UML based on CPN Formalism, *Proc. of APSEC'06*, pp. 411-418, ISSN: 1530- 1362, 0- 7695-2685-3, Bangalore India, Dec 2006, IEEE, USA

Sivaraman, E. & Kamath M. (2003). On the use of Petri Nets for Business Process Modeling, *On The Move to Meaningful Internet Systems 2003: OTM 2003Workshops, Business Contract Obligation Monitoring through Use of Multi Tier Contract Ontology LNCS 2889/2003*, Oct 2003, pp. 690-702, ISSN: 0302-9743

Soderborg, N.R.; Crawley, E.F. & Dori, D. (2003). OPM- Based Definitions & Operational Templates, *Communications of the ACM*, Vol. 46 No. 10, Oct 2003, pp. 67-72, ISSN:0001-782

Spiteri Staines, T. (2008). Intuitive Mapping of UML 2 Activity Diagrams into Fundamental Modeling Concept Petri Net Diagrams and Colored Petri Nets, *Proc. of the 15th IEEE International Symposium and Workshop on Engineering of Computer Based Systems*, pp. 191-200, ISBN: 0-7695-3141-5, Belfast, Ireland, ECBS, Apr 2008, IEEE, USA

Störrle, H. (2004a). Structured Nodes in UML 2.0 Activities, *Nordic Journal of Computing*, Vol. 11, No. 3, Sep 2004, pp. 279-302, ISSN: 1236-6064

Störrle, H. (2004b). Semantics of Control Flow in UML 2.0 Activities, Proc. of 2004 IEEE Symposium on Visual Languages and Human Centric Computing, pp. 235-242, ISBN: 0-7803-8696-5, 2004,IEEE, USA

Störrle, H. & Hausmann, J.H. (2005). Reasoning about UML Activity Diagrams, *Publ. Assoc. Nordic Journal of Computing*, Vol. 14 No. 1, 2005, pp.43-64, ISSN: 1236-6064

Tabeling, P. & Gröne, B. (2005). Integrative Architectural Elicitation for Large Scale Computer Based Systems, *Proc. of the 12th IEEE International Symposium and Workshop on Engineering of Computer Based Systems*, pp. 51-61, ISBN: 0-7695-2308-0, Green Belt, MD, ECBS, Apr 2005, IEEE, USA

Tabeling, P. (2002). Multi Level Modeling of Concurrent and Distributed Systems, *Proc. of the International Conference on Software Engineering Research and Practice*. CSREA Press, Jun 2002, http://www.fmc-modeling.org/download/publications/tabeling_2002-multi-level_modeling_of_concurrent_and_distributed_systems.pdf

Tabeling, P. (2004). Architectural Description with Integrated Data Consistency Models, *Proc. of the 11th IEEE International Symposium and Workshop on Engineering of Computer Based Systems*, pp. 178- 185, ISBN: 0-7695-2125-8, Brno Czeck Republic, ECBS, May 2004, IEEE, USA

Van der Aalst, W.M.P. (1998). The Application of Petri Nets to Workflow Management. *Lecture Notes*, Eindhoven University of Technology, Eindhoven, http://tmitwww.tm.tue.nl/staff/wvdaalst/publications/p53.pdf

Van Hee, K.M. (1995). *Information Systems Engineering A Formal Approach*, Cambridge University Press, ISBN: 0-521-45514-6, N.Y. USA

Van Hee, K.M. (2005). Information Systems Architecture A Practical and Mathematical Approach, Lecture Notes, Technische Universiteit Eindhoven, 2005, http://wwwis.win.tue.nl/~wsinhee/sm1/

Ziaei, R. & Agha, G.(2003). SynchNet : A Petri Net Based Coordination Language for Distributed Objects, *Generative Programming and Component Engineering LNCS 2830/2003*, Springer Verlag, Vol 48, 2003, pp. 324-343, ISBN: 3-540-20102-5, N.Y.

# Multilevel Petri Nets for the Specification and Development of Workflow Automation Systems

Marina Flores-Badillo and Ernesto López-Mellado
*CINVESTAV Unidad Guadalajara*
*México*

## 1. Introduction

Since its first use in 1980, the term Workflow has been gaining a lot of importance, especially in the improvement of organizational performance in a broad variety of industries (Shi et al., 1998).

Nowadays, Workflow Management represents a critical issue for achieving enterprise competitiveness among organizations, where workflow (WF) is concerned with the automation of procedures where documents, information, or tasks are passed to the participants according to a defined set of rules to achieve, or contribute to, an overall business goal (WfMC, 1999; van der Aalst & Hee, 2002; van der Aalst, 1998). Workflow management aims to improve process performance and helps to achieve business goals with high efficiency (Reijers & van der Aalst, 2005).

According to the Workflow Management Coalition, a WF Management System (WFMS), is "a system that defines, creates, and manages the execution of workflows through the use of software, running on one or more WF engines, which is able to interpret the process definition, interact with WF participants and, where required, invoke the use of IT tools and applications" (WfMC, 1999).

In WFMS development lifecycle, modelling is the first and most important stage and it lacks a standardized theory that provides a theoretical background (van der Aalst, 1998); that is why there are several modelling approaches which have their strengths and weaknesses in different aspects (Lu & Sadiq, 2007). However, Petri Nets (PN) are usually used for formal modelling of WFs due to its formal semantics, graphical description, and abundance of analysis techniques (van der Aalst, 1996).

Since Zisman (Zisman, 1977) used PN to model WF processes, PN has been widely used for specifying, analysing and verifying the properties of static workflow models (van der Aalst, 1998; van der Aalst & Anyanwu, 1999; Adam et al., 1998; Li et al., 2004), and extended for improve the expressiveness (Eshuis & Dehnert, 2003; van der Aalst & Hofstede, 2005).

Although PN provides clear and unambiguous models, when the systems are large and complex the handling and building of ordinary PN models becomes a difficult task.

Also, complex WFMS need to integrate other technologies, such as agent technology, which provides flexible, distributed, and intelligent solutions for business process management.

The notion of *agent* in (Yan et al., 2001) is used as "a computer system situated in some environment, which is capable of autonomous action in this environment in order to meet its design objectives" (different notions can be found in (Wooldridge, 2002; Nwana, 1996)). These works also highlight the benefits of applying agent technology to *business process* (BP) management; some of these benefits are: distributed system architecture, the inherent autonomy of software agents because agents can start a WF based on event trigger, the agent reactivity because it have the ability to generate alternative execution paths, etc. An intelligent agent is capable of autonomous operation and flexible behaviour in order to meet its design goals and also has the properties of reactivity, pro-activity, and social ability (Wooldridge, 2001).

In other works both concepts are integrated. In (Repetto et al., 2003), a methodology for the design of agent based WF was presented; it consisted in three steps. In the first step the authors model the *BP* with UML Activity diagrams by identifying all the necessary resources and activities. In the second step, all the activities identifying roles in parallel paths are grouped. Finally, they define an agent for each group.

A collaborative approach for workflow systems is presented in (Savarimuthu & Purvis, 2004) where agents collaborate by forming social network (societies), in (Savarimuthu et al., 2004) agents are embedded in a system that can monitor and control the overall functioning of a workflow process in an agent based WF system.

In (Wang et al., 2005), agent technology is used for the WF monitoring where various intelligent agents work together to perform flexible monitoring tasks in an autonomous and collaborative way.

This work presents a method for the specification and development of agent-based workflow processes controllers. The proposed two phase methodology consists of a) the modular modelling of the system (using a PN based formalism called n-LNS); and b) an implementation technique using the obtained models and the middleware JADE (Bellifemine et al., 2007) as a development tool, where the models allow the synthesis of agent-based software in which mobile agents guide the process through organizational units and execute different tasks.

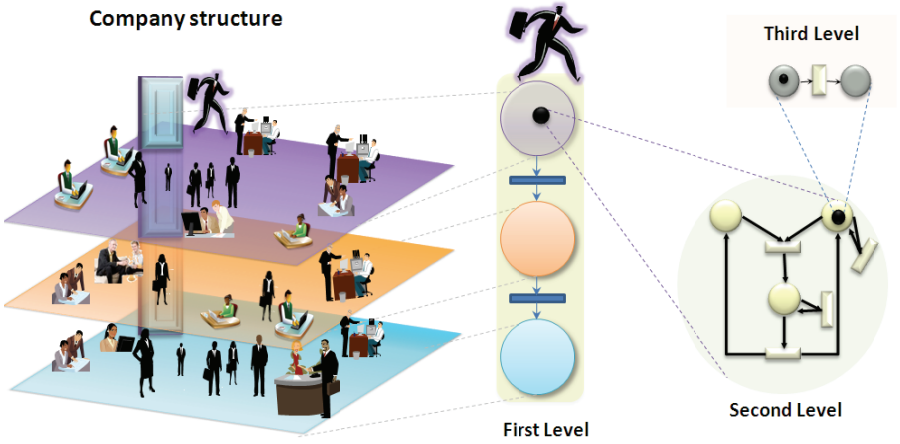



Fig. 1. General modelling strategy

A workflow model may be clearly represented using three levels of the n-LNS formalism (see Fig. 1), where the first (highest) level describes the company structure and the possible flows between organisational units; the second level net describes the general behaviour of a entity which guides the processes within the company according to the defined workflow process definition, which is represented by a net of the third level; also the third level nets describe the tasks to be performed to complete the workflow case. The remainder of this chapter is organized as follows. Section II introduces the n-LNS formalism. In Section III we present a methodology for the modelling of workflows using n-LNS; simulation results are included. Section IV presents a software development technique using JAVA and the middleware JADE for component definition. Finally, Section V presents some current research and trends.

## 2. A Multi-Level Net System

### 2.1 High level Petri Nets

Among the existing modelling formalisms for discrete event systems, Petri nets (PN) have been widely adopted as a description formalism in the automation community, because of its characteristics issued of its graphical nature and its simple mathematical support: clearness and compactness to represent complex behaviour, namely causality, concurrence, parallelism, decisions, synchronization, and information exchange (DiCesare et al., 1993).

The largeness of PN describing complex systems motivated the creation of high level PN, extensions to the original formalism that allow compact models of systems. First, Predicate-transition nets proposed in (Genrich & Lautenbach, 1979), and coloured PN (CPN), proposed in (Jensen, 1981), allow token identities (colours) represented by symbols in the marking. The expressiveness of CPN was increased by associating data structures to coloured tokens (Kasturia et al., 1988).

Later, high level modelling formalisms appeared to deal with object oriented software specification: concepts on high level PN and object oriented programming were merged leading to Cooperative Objects (Sibertin-Blanc, 1994) and Object Oriented PN (Lakos, 1995; Valk, 1991). In these works the tokens can be in turn other PN. These methods bring near the models to software implementation in despite of the loss of clearness of the description.

Recently, the idea of considering PN as tokens is retrieved by R. Valk (Valk, 1998) who proposed a two-level PN "code-clean" formalism independent to programming languages. Holding the same notion of nets into nets, K. Hiraishi in (Hiraishi, 2000) proposes PN2, a two level formalism similar to the Valk´s definition; also, I. Lomazova in (Lomazova, 2000) proposes Nested PN; also, in (Kummer, 2001) Reference nets are defined as a support of a simulation tool in which the tokens are references to other nets.

The proposed multi-level net system called *n*-LNS consists mainly of an arbitrary number of nets organized in n levels according to a hierarchy; *n* is chosen according to the degree of abstraction that is desired in the model. A net may handle similarly to tokens, nets of deeper levels and symbols; the nets of level n permits only symbols as tokens, similarly to CPN. Interactions among nets are declared through symbolic labelling of transitions.

### 2.2 n-Level Net System

The definition of *n*-LNS includes the description of the components (structure and marking), declaring of interaction (transition labelling), and the enabling and firing rules (embedding the synchronization mechanism). In this definition the basic notion of PN structure is used.

**Definition 1.** A PN structure is a triple $G = (P, T, F)$ where $P$ and $T$ are finite nonempty set of places and transitions respectively, $P \cap T = \emptyset$, and $F: P \times T \cup T \times P$ is a flow relation of the net. Pictorially, places are represented as circles and transitions as bars or boxes.

♦ ***Type Nets***

**Definition 2.** A type-net of level i is a tuple $typenet_i = (G, TOKEN_i, LABEL_i, VAR_i, \tau, \lambda, \pi)$ for $1 \leq i \leq n$, where:

- $G$ is a PN structure.
- $TOKEN_i$ is a finite non empty set of type-nets and symbols permitted into the places of a net level *i*: $TOKEN_i \subseteq \{typenet_{i+j,k} \mid j = 1,..,n-i, k = 1,..,r\} \cup SYMBOL_i$
  *n* is the number of levels of a multi level net system,
  *r* is the number of different type-nets allowed into places of a net of level *i*.
  $SYMBOL_i$ is a finite set of symbols allowed into the places of a net of level *i*.
- $LABEL_i$ is a finite set of labels defined for a net level *i*: $LABEL_i \subseteq LABELS$.
- $VAR_i = \{x, y, ...\}$ is a finite set of variables defined to net level *i*; $VAR_i \subseteq VARS$, where:
  $Type: VAR_i \rightarrow TOKEN_i$ is an assignment function of type nets to places.
  $Type(x) = \{Type(x) | \forall x \in VAR_i\}$ is the set of types associated to variables.
- $\tau: P_i \rightarrow 2^{TOKEN_i} - \emptyset$ is an assignment function of type nets to places.
- $\lambda: T_i \rightarrow 2^{LAB_i} - \emptyset$ is an assignment function of labels to transitions where:
  If $i = 1$ then $LAB_i = LABEL_i$
  If $2 \leq i \leq n-1$ then $LAB_i = (LABEL_i \cup \{\varepsilon\}) \times (LABEL_i \cup \{\varepsilon\}) \times (LABEL_i \cup \{\varepsilon\})$
  If $i = n$ then $LAB_i = (LABEL_i \cup \{\varepsilon\}) \times (LABEL_i \cup \{\varepsilon\})$
- $\pi: F_i \times LABEL_i \rightarrow M_{VAR_i \cup SYMBOL_i}$ is a weighting function that assigns to every arc, a multi-set of variables and symbols, with respect to transition labels. If $label \notin \lambda(t), \pi((p,t), label) = \pi((t,p), label) = \emptyset$. Moreover if $i = n$ then $VAR_i = \emptyset$, so that $\pi: F_n \times LABEL_n \rightarrow M_{SYMBOL_n}$.
- *LABELS* and *VARS* are the sets of all the symbols representing the labels and variables respectively used in a model.

A type-net $typenet_i$ is a PN structure with additional information that declares and handles data defined in $TOKEN_i$, according to the *pre* and *post* conditions established by $\pi$ and the labelling, specified by $\lambda$ for the interaction between nets.

The function $\tau$ assigns to every place a set of type-nets and symbols that belong to $TOKEN_i$.

The function $\lambda$ assigns to every transition a set of labels. The labels of the lowest level are pairs: the first element indicates that the transition must be synchronized locally (with a transition of other net of the same level), and the second element indicates that the synchronization must be external (with a transition of a net of the next upper level), unless this pair is $(\varepsilon, \varepsilon)$. In the intermediate levels *(i=2, .., n-1)* a label is a triple: the elements in the triple declare local, internal (with a transition belonging to a token-net) and external synchronization, respectively only if the label is different form $(\varepsilon, \varepsilon, \varepsilon)$. The transitions of the highest level net are labelled either with a symbol or $\varepsilon$, declaring internal synchronization.

The function $\pi$ determines on (p, t) arcs, the amount and type of token-nets and symbols needed into the input places to enable a transition (those that must be removed), and on *(t,*

*p)* arcs, the amount and type of token-nets and symbols that must be added into the output places.

♦ ***Nets of level i***

**Definition 3.** A net of level *i* is a tuple $NET_i = (typenet_i, \mu_i)$; $1 \leq i \leq n$, where:

- $typenet_i$ is a type-net of level i
- $\mu_i: P_i \rightarrow M_{NETS_{TOKEN_i} \cup SYMBOL_i}$ is a marking function for the type-net of level i; it assigns to every place $P_i$ a multiset of nets and/or symbols.
  $NETS_{TOKEN_i} \subseteq \{NET_{i+1}, NET_{i+2}, \dots, NET_n\}$

A net of level *i* is a type-net $typenet_i$ with a marking $\mu_i$, which provides the current distribution of symbols and nets of $NETS_{TOKEN_i}$ into the places.

♦ ***Net System***

A n-LNS model, called *Net System,* is the set of all nets defined at all levels.

**Definition 4.** A n-level Net System (n-LNS) is a n-tuple $NS = (NET_1, NET_2, \dots, NET_n)$ where:

- $NET_1$is the highest level net
- $NET_i = \{NET_{i,1}, NET_{i,2}, \dots, NET_{i,r}\}$ is a set of *r* nets of level *i.*

The Fig. 2 sketches pieces of the components of a 4-LNS. The level 1 is represented by the $NET_1$, the level 2 by the nets $NET_{2,1}$ and $NET_{2,2}$, the nets $NET_{3,1}$, $NET_{3,2}$, $NET_{3,3}$, $NET_{3,4}$ and $NET_{3,5}$ compose the level 3, and finally the nets $NET_{4,1}$, $NET_{4,2}$, $NET_{4,3}$ and $NET_{4,4}$ form the level 4.

### 2.3 Net System Evolution

The components of the model may interact among them through synchronization of transitions; the synchronization mechanism is included in the enabling and firing rules of the transitions. This mechanism establishes that two or more transitions labelled with the same symbol must be synchronized. In order to define the enabling conditions and firing of transitions we introduce first the notion of variable binding.

**Definition 5.** A binding $b$ on a variable set $VARS = \{x, y, \dots\}$ is a function $b: VARS \rightarrow NETS_{TOKEN_i}$; for a $v \in VARS$; $b(v)$ is a next lower level net whose the type is $Type(v)$. $b_t$ maps every variable defined on the weight of the input arc to the transition *t,* with respect to a label. $m_{<b>}$ denotes a multiset of nets resulting of instancing a multiset of variables *m* with the binding *b.*

♦ ***Enabling rule***

**Definition 6**. A transition *t* of a net of level *i* $NET_i$ is enabled with respect to a label $lab \in \lambda(t)$ if:

- There exists a binding $b_t: VAR_t \rightarrow NETS_{TOKEN_i}$, where $VAR_t$ is the set of variables appearing in all $\pi((p,t), lab)$, and it must fulfill that $\forall p \in \bullet t, \pi\big((p,t), lab\big)\langle b_t \rangle \subseteq \mu_i(p)$.
  (The binding $\langle b_t \rangle$ is omitted when the level net is n, since the arc weight does not have variables. Therefore the condition is simplified).
- The conditions of one of the following cases are fulfilled:
  *Case 1.* If $lab = (\varepsilon, \varepsilon, \varepsilon)$ then no additional condition is required. The firing of t is autonomously performed.
  *Case 2.* If $lab \neq (\varepsilon, \varepsilon, \varepsilon)$ one must consider one or a combination of the following situations:
  *i)* $lab = (l, \varepsilon, \varepsilon)$. It is required the simultaneous enabling of the transitions labelled with $l^{\equiv}$ belonging to other nets into the same place $p'$ of the next upper level net, marking

the input places of $t$. The firing of these transitions is simultaneously performed and all the (locally) synchronized nets remain into the same place $p'$.

*ii)* $lab = (\varepsilon, l, \varepsilon)$. It is also required the enabling of the transitions labelled with $l^{\uparrow}$ belonging to other next lower level nets into $\bullet\, t$. These transitions fire simultaneously and the lower level nets and symbols declared by $\pi\big((p,t), lab\big)\langle b_t\rangle$ are removed.

*iii)* $lab = (\varepsilon, \varepsilon, l)$. It is also required the enabling of at least one of the $t' \in p'\bullet$, labelled with $l^{\downarrow}$, of the upper level net where the $NET_i$ is contained. The firing of $t$ provokes the transfer of $NET_i$ and symbols declared into $\pi\big((p',t'), lab\big)\langle b_t\rangle$.

A label may involve a combination of any of these clauses. So, a label $(l, l, l)$ indicates that a transition must be synchronized locally, internally, and externally respect to the symbol $l^{\equiv\uparrow\downarrow}$.

These situations apply to all levels of net, except for the nets of levels 1 and $n$. In the case of a net level 1 it is only possible the *internal synchronization*, so the transitions may be labelled with $l^{\downarrow}$ or $\varepsilon$. In the case of a net of level $n$ *local* and/or *external synchronization* may be declared; the transitions may be labelled with $l^{\equiv}$, $l^{\uparrow}$, $l^{\equiv\uparrow}$, or $\varepsilon$.

♦ ***Firing rule***

The firing of transitions in all level nets modifies the marking by removing $\pi\big((p,t), lab\big)\langle b_t\rangle$ in all the input places and adding $\pi\big((t,p), lab\big)\langle b_t\rangle$ to the output places. The binding $\langle b_t\rangle$ is not necessary for nets of level $n$.

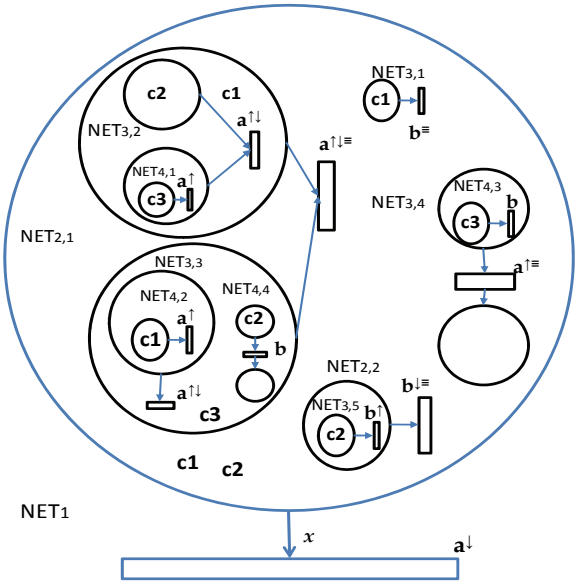



Fig. 2. Piece of a 4-Level Net System

In Fig. 2, $NET_1$ is synchronized through the transition labelled with $a^{\downarrow}$ with $NET_{2,1}$ and $NET_{3,4}$ by mean of the transitions (locally synchronized) labelled with $a^{\uparrow}$; also $NET_{2,1}$ is synchronized with nets $NET_{3,2}$, $NET_{3,3}$, $NET_{4,1}$ and $NET_{4,2}$ through transitions labelled with $a^{\uparrow\downarrow}$ and $a^{\uparrow}$; all these transitions must be enabled to fire. The simultaneous firing of the

transitions removes these nets from the input places; $NET_{4,1}$ is removed from the place of $NET_{3,2}$, and $NET_{4,2}$ is removed from the place of $NET_{3,3}$.
$NET_{2,2}$, $NET_{3,1}$ and $NET_{3,5}$ are synchronized through the transitions labelled with $b^{\downarrow}$, $b^{\equiv}$, and $b^{\uparrow}$ respectively; the firing of the transitions changes the marking of $NET_{2,2}$ and $NET_{3,1}$; $NET_{3,5}$ is removed from the place of $NET_{2,2}$.

## 3. Workflow process specification

### 3.1 General Strategy

The use of n-LNS induces a modular and hierarchical modelling methodology allowing describing separately the environment and the behaviour of all the involved components in a workflow problem, and then, integrated such models into a global one through transition synchronization.
As mentioned before, the workflow problem can be clearly described using only 3 levels of the n-LNS (see Fig. 1). The highest level net (1) describes the organization structure, i.e. the different organizational units involved in the solution of the workflow problem (*case*) and the relations (flow of information, tasks or documents) among them.
The second level net describes the general behaviour of an entity (software agent) which guides a process (current *case*) within the organizational units and executes the tasks within the departments expressed in the WF-process definition net (both types of nets modelled as nets of level 3 and described later); this entity net will be a token-net for the net of level 1 (Flores-Badillo et al., 2009a).
The method, for the sake of readability, is illustrated through a case study dealing with claim processes in an insurance company.
Consider the following description: "Define the WF for the claim processes in an insurance company in which a customer claims the insurance policy of a personal property (real state, car, life insurance). The company must receive the claim, request personal data from the customer (insurance policy number, etc.), and verify the insurance validity, payments, and beneficiaries. It must do the adjustment of real damages, validate the *case*, calculate the corresponding assessment, do the necessary payments to the customer if the complain is valid, or inform in case that the process has some invalid data".
The remainder of this section describes the definition of the different nets that compose the model at three levels.

### 3.2 Level 1 Net

This net describes the general structure of the company, where the WF processes are performed (environment net), and the possible workflow between every pair of *organizational units* or departments, considering all the possible cases. First, the different departments in which some tasks are executed are identified; then one must consider all the possible flows. The departments are represented by PN places and the flows by PN transitions (see Fig. 3), arc orientation corresponds to the direction of the flow. Firing of a transition means that the next task will be performed in the organizational unit that represents the output place of the fired transition. Furthermore, places for representing resources may be added in a classical structure allocation-release.

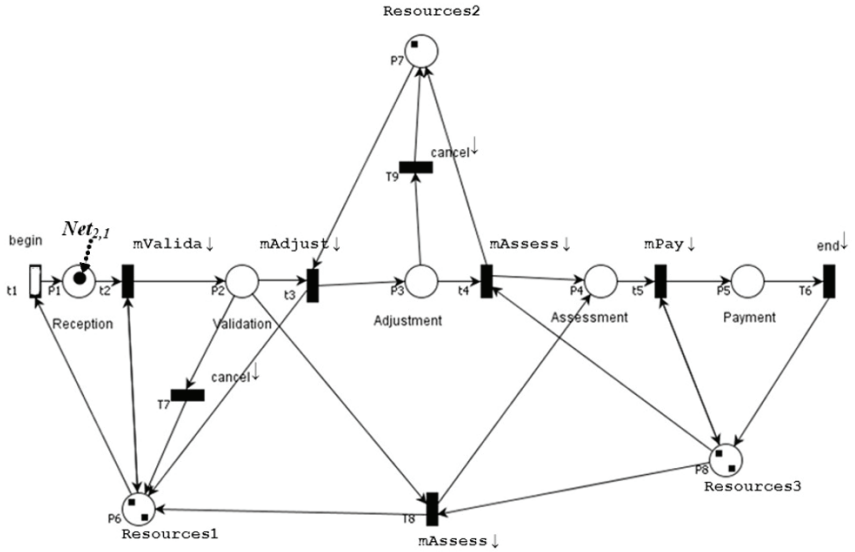



Fig. 3. $Net_{1,1}$ net describing the organizational structure

Tokens in this net are nested nets of level 2; for example in place *P1* of the model of Fig. 3, there is a token $Net_{2,1}$ which will be detailed later. If needed, transitions for allowing the cancelation of *cases* (see *t7* and *t9* in Fig. 3) can be added. Once this net is defined, synchronization attributes to the transitions must be declared. Also it is recommended using at least two transitions for representing the beginning and ending of a *case*. All transitions should include the *internal* (↓) *synchronization attributes*; it implies that the token-nets of level 2 must have some transitions labelled using the *same symbols*, with the external (↑) synchronization attributes.

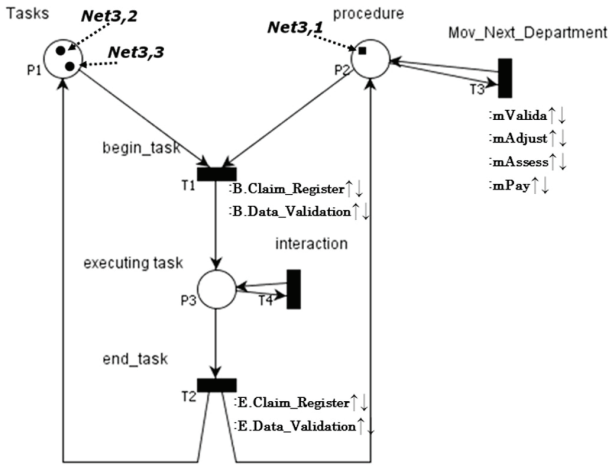



Fig. 4. $Net_{2,1}$, net corresponding to a Mobile entity which handles the current WF case.

The *case* begins with the firing of the transition *t1*, and ends with the firing of the transition *t6* according to Fig. 3. Finally define $Net_{1,1} = (G, TOKEN_{1,1}, LABEL_{1,1}, VAR_{1,1}, \tau, \lambda, \pi)$. We can also include in this model another kind of information, for example, the available resources for each department. This can be achieved adding places to the obtained net.

### 3.3 Level 2 Net

Define $Net_{2,1} = (G, TOKEN_{2,1}, LABEL_{2,1}, VAR_{2,1}, \tau, \lambda, \pi)$, which models the general behaviour of an entity (mobile agent) that handles the *case*. This agent should know which tasks are involved in the WF process, in which department these tasks will be performed, and the pertinent information of the current *case* (that modify the order of tasks execution). Fig. 4 shows a simple agent net that handles as tokens, level 3 nets describing tasks and the WF process plan.

Initially, task nets ($Net_{3,2}, Net_{3,3}$) reside in place *P1*, and the *WF-process definition net* ($Net_{3,1}$) is nested in place *P2* as initial marking. $Net_{3,1}$ determines which task (from all the available tasks in *P1*) has to be performed in the next step; when *T1* fires, both nets are placed into place *P3* where the task nets evolves performing the involved transactions. When task is finished, this net enables the transition synchronized with *T2* of $Net_{2,1}$; the firing of *T2* returns back the task net into *P1* and $Net_{3,1}$ into *P2*. Both transitions (*T1* and *T2*) must be declared with internal synchronization (↓). When the next task needs to be performed in a different department than the current, transition *T3 must fire externally* (↑) synchronized with the environment net ($Net_{1,1}$) controlling the agent migration, and *internally* (↓) synchronized with $Net_{3,1}$ because it defines the behaviour of the agent, i.e. the operations that it will perform. An additional transition (*T4*) can be included for interaction with other stationary or mobile agent nets.

### 3.3 Level 3 Nets

The nets of this level may be evolving within the agent net for providing a specific behaviour, namely the process plan, the tasks, or the agent interaction protocols. These nets may be within the environment model for specifying the state of resources or other stationary entities. For the case of study described, two types of level three nets are used: 1) those for describing the different tasks involved in the process of the case, and, 2) the net which describes the execution order of the tasks and the migration of the agent through the company (workflow process).

- ***$Net_{3,1}$ WF-process definition net***

The *WF-Process definition net* ($Net_{3,1}$), is a net having a simple structure and it must represent the possible sequences of tasks for accomplishing the handled cases of a WF process; the sequence include tasks executions and displacements between departments. Every stage in the plan is specified as place representing the tasks or the displacement operations; two transitions (as the labelled with *B.Claim_register*↑ in T0 and *E.Claim_register*↑ in T1 of Fig. 5) represent the beginning/ending of a task execution; each of this particular transitions must be synchronized with the corresponding net which models tasks and with the agent net $Net_{2,1}$.

The transitions representing displacements (as *mValida*↑) must be synchronized with *T3* of $Net_{2,1}$. All the transitions must be externally↑ synchronized. Finally define $Net_{3,1}$ =(G, $TOKEN_{3,1}$,$LABEL_{3,1}$,$VAR_{3,1}$, τ, λ, π). An example of this net is showed in Fig. 5.

♦ ***$Net_{3,i}$ tasks nets***

Each task is expressed as a simple sequence of transactions, where two transitions are used for emphasizes the beginning and the ending of a task execution. We can obtain each net identifying the operations required for each task and its precedence order. Pertinent labels are assigned for external synchronization with the agent net $Net_{2,1}$, and local synchronization with $Net_{3,1}$. For each task $i$ define the nets $Net_{3,i}$ = (G, $TOKEN_{3,i}$, $LABEL_{3,i}$,$VAR_{3,i}$, τ, λ, π), for $i$=2,3,….$r$, $r$ is the number of tasks plus 1. Fig. 6 shows the models of two tasks.

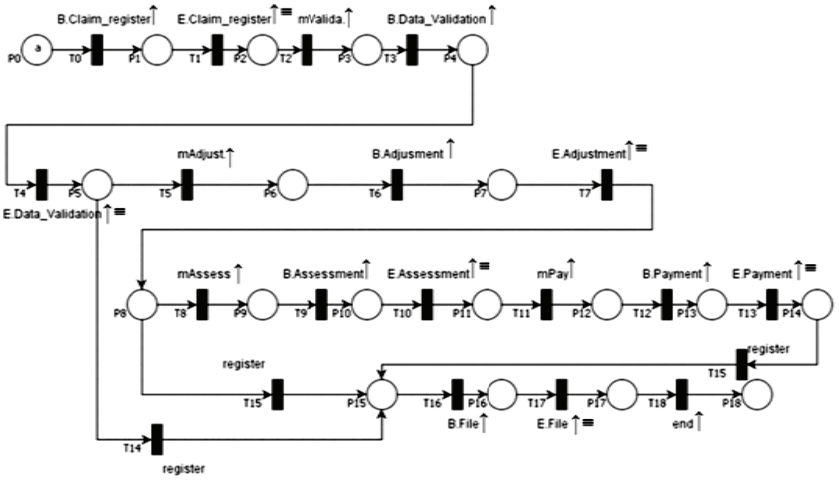



Fig. 5. $Net_{3,1}$ defining a simple WF-process

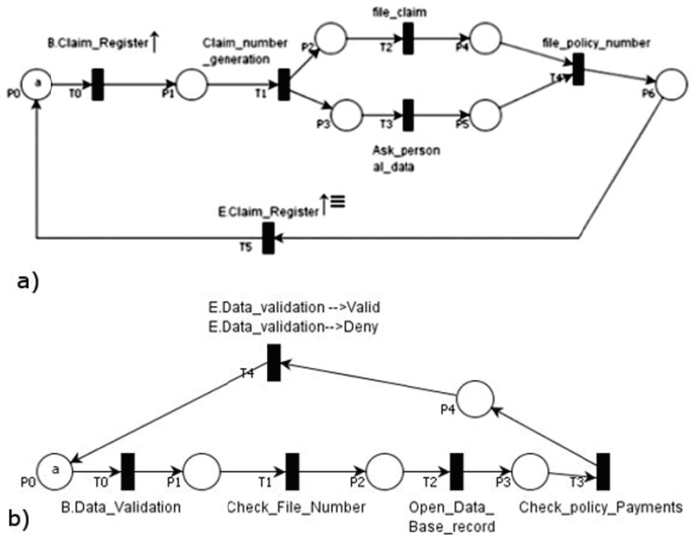



Fig. 6. Level 3 nets modelling two tasks: a) $Net_{3,2}$ for a Claim_Register task, b) $Net_{3,3}$ for a Data_Validation task.

### 3.4 Model Simulation

Although the model construction is easily performed by following the proposed modelling methodology, the size of models becomes large as the complexity and quantity of activities in the system grow. In the absence of analytical procedures for verifying the correct functioning of the resulting models, interactive simulation appears as a useful and suitable solution for validating the obtained models. This task is enlightened by the use of a simulation software tool. In this section we describe the simulation procedure for n-LNS models and the tool for editing and executing such models.

- ***Model Execution***

The simulation of n-LNS models consist in the interactive execution of the models according to the rules for enabling and firing transitions. We are going to briefly describe a procedure, usually named *token player*, which implements such rules.

The n-LNS models, edited through a visual interface, are coded into a XML file according to the standard format Petri Net Markup Language (PNML) (Weber & Kindler, 2003); such file is called *Data model*. Besides the PN structure, other information is stored in the Data model, namely the level of the net, transition labels and their attributes, weight functions, marking, and Java code associated to transitions for software generation purposes.

From the Data model, a set of objects representing every $Net_{i,j}$ are built. Every object evolves as the corresponding $Net_{i,j}$ by verifying the transition enabling conditions and the transition firing mechanism. Both enabling and firing procedures involve interactions among $Net_{i,j}$-objects for checking synchronizations declared on the labels.

- ***n-LNS Simulation Tool***

The simulation of the case of study has been performed through the execution of the 3-level net model described above. This task was possible with the help of MASGAS, a software tool that allows the visual edition and the interactive execution of multi level net models expressed in n-LNS.

The tool provides facilities for the interactive execution of the model: for a current marking the system indicates, through the highlighting of transitions, which transitions are enabled with respect to which label; then the user selects the transition to fire. After the firing the new markings of the involved nets are reached and displayed on the corresponding windows. Based on n-LNS, MASGAS provides a complete syntactic control over the different hierarchical levels of the net system; this avoids nesting of upper level nets allowed by other net within nets simulators (Kummer et al., 2002).

Below we are including several views of the edited model. Every net is built in a single window and it can be saved and updated for model adjustments.

In Fig. 7 it is showed the net which describes the company structure where the workflow tasks will be performed ($Net_{1,1}$), the token in *P1* is the nested net $Net_{2,1}$ which models the general behaviour of the mobile agent that guides the handled case. Notice that the transition *t2* is enabled (respecting to the label *mValida*); when it is fired, the agent migrates from one organizational unit (*Reception department* in *P1*) to another (*Validation department* in *P2*). In Fig. 8 the $Net_{2,1}$, is showed; the edited net shows the defined initial marking. At this situation the entity is ready to perform the initial task in the pertinent department. When transition *T1* of $Net_{2,1}$ is fired, it means that the net contained in its place *P2* (net $Net_{3,1}$ in Fig. 5) indicates that a particular task (one task net from the available in *P1*) need to be performed and will be selected (through transitions synchronization).

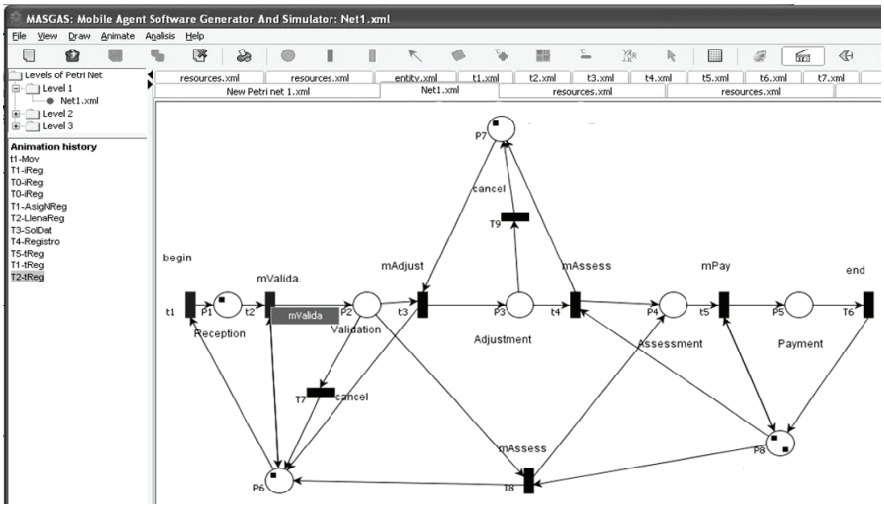



Fig. 7. $Net_{1,1}$ (Organizational Unit Net): t2 is enabled with respect to *mValida*

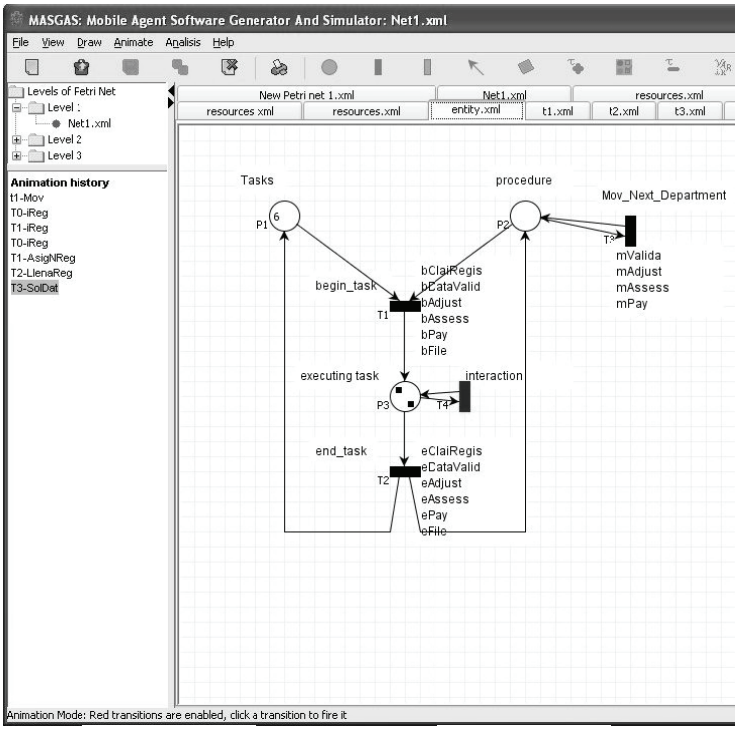



Fig. 8. Level 2 net (Mobile entity Net), after the firing of the transition *T1*: a token net of place *P1* and the token net of place *P2* are placed into *P3*

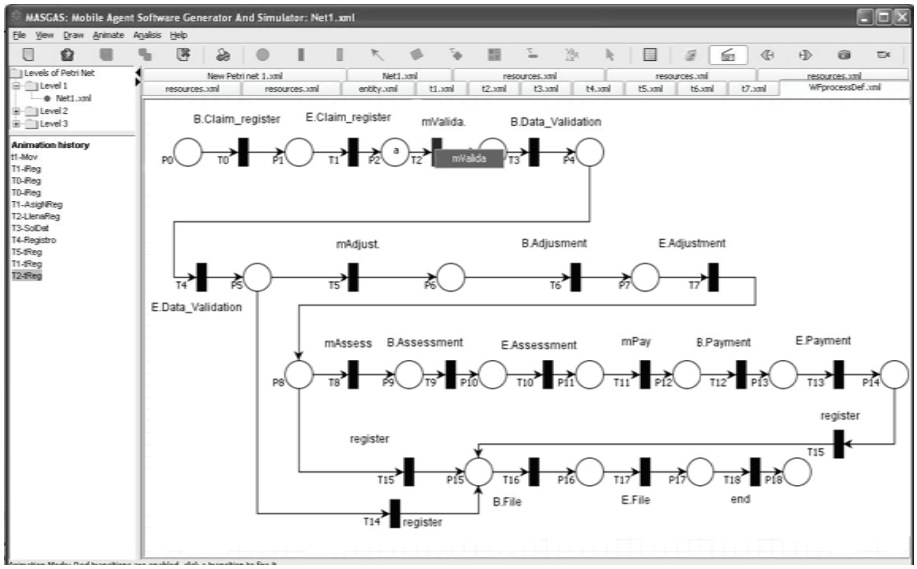



Fig. 9. $Net_{3,1}$; T2 (label *mValida*) is enabled: the entity need to migrate to the Validation Department (P2 in $Net_{1,1}$).

Fig. 9 shows the edited model for the *WF-process definition Net* $Net_{3,1}$. When the given task needs to be performed in a different department than the current one, the transition *Move_Next_Department* of the $Net_{2,1}$ in Fig. 8, is used; it indicates that the entity migrated to other department within the organization (label *mValida* in Fig. 9).

## 4. A Software Development Technique

### 4.1 Overview of JADE

Java Agent Development Framework (JADE) is a software framework completely implemented in Java language which simplifies Mobile Agent (MA) system implementations by using a middleware which fulfill FIPA (FIPA, 1997) specifications. The agent platform can be distributed through machines (which not necessary share the same OS) and the configuration can be managed by a remote GUI (Bellifemine et al., 2007).

The communication architecture offers flexible and efficient message passing where JADE creates and manages the incoming private ACL message queue for each agent. The complete FIPA communication model has been implemented and its components have been clearly distinguished. JADE integrates completely interaction protocols, ACL, ontology's, transport protocols, etc. Most of the FIPA defined protocols are available in JADE.

### 4.2 Software architecture

This section presents a technique for the development of Mobile Agent-based WFMS. The basic idea for conceiving such a system is that a MA guides the workflow process through the different organizational units in which several tasks are executed according to the

handled case. The application, obtained from an n-LNS model, runs concurrently in a set of networked computers where mobile agents execute concurrently the WF process for the different cases, interacting with stationary (software or human) agents.
During the design phase the components are described in a clear and compact way. The system is described as a set of interconnected organizational units that have a specific resource allocation. The agent behaviour is determined by two kinds of specifications: a) the description of the agent general behaviour and common knowledge for all the agents, namely, basic operations, and interaction protocols (collaboration, and resource competition); and b) particular descriptions of a specific behaviour such as the task plan and an accessibility roadmap, which describe the assigned process and the permitted access to the organizational units respectively.
The implementation phase is supported by a software development guideline allowing the definition of Java components (using also the middleware JADE) from agent systems specification from the design phase. The obtained software is distributed in a set of networked computers that manages MA migration. The modularity allows adaptations to system specification changes without difficulties.
Below an outline the main steps of the software synthesis methodology is presented, where the Mobile Agent Environment is first defined, and then both general and specific agent behaviours are described. All the system components are defined using the information given by the Nets obtained in the modelling phase.

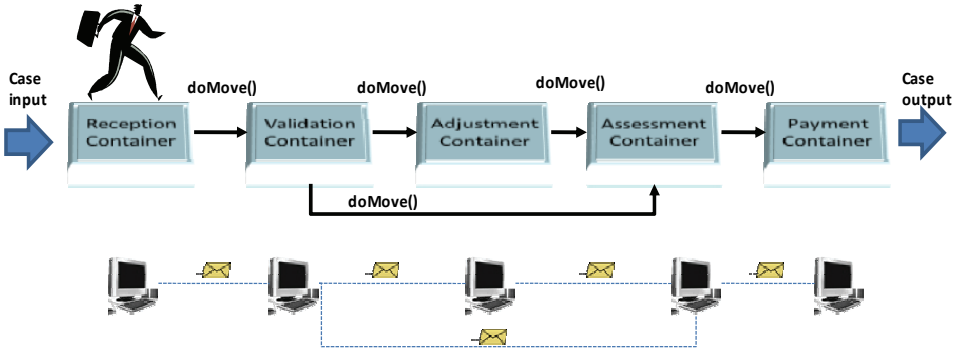



Fig. 10. Platform for the Multi-Agent based Workflow System

♦ ***Environment Structure***

This is defined using the information given by the $Net_{1,1}$, net which describes the general structure of the company; where each place of this net (representing a different department) can be seen as a site (host, container) of the Organization Local Area Network (LAN), placed each one in a different office; every site will contain an agent platform, so each transition of the $Net_{1,1}$ represents a migration of an agent from one site to another. In this way we can create a different JADE agent container for each place in that Net; a container can be created using: C:\ java jade.Boot –container Name_Host [-gui]. In Fig. 10 a strategy for the platform distribution is proposed, but this can have a different distribution due to the company's net organization.

♦ ***Defining Agents with JADE***

The class for deriving all the mobile agents is defined extending the basic JADE class *Agent*. The common base class *jade.core.Agent* provides all the necessary features to accomplish basic interactions with the agent platform, where the *setup()* method registers an agent in the Directory Facilitator (DF) and deregister it with the *takedown()* method.
Furthermore, since mobile objects are used, one must register the SLCodec language and the MobilityOntology by adding the lines:

*getContentManager().registerLanguage(newSLCodec());*
*getContentManager().registerOntology(MobilityOntology.getInstance());*

which are also included in the *setup()* method (see Fig. 11).
In Fig. 11 an Agent general structure is described, where the specific behaviour of this agent must be added in the highlighted part of the code, defining all tasks that this agent should perform is order to process the WF *case*. This agent functionality is defined using the level three nets defined in the modelling phase.

```
public class MobileAgent extends Agent {
  public void setup() {
      // Register in the yellow pages
          DFAgentDescription dfd = new DFAgentDescription();
          dfd.setName(getAID());
          ServiceDescription sd = new ServiceDescription();
          sd.setType(type);      sd.setName(name); dfd.addServices(sd);
          try {
           DFService.register(this, dfd);
          }catch (FIPAException fe) {fe.printStackTrace();}

        //Register the mobility ontology
     getContentManager().registerLanguage(new SLCodec());
getContentManager().registerOntology(MobilityOntology.getInstance());
            ////////////////////// // Add agent behaviour
          }
  public void takeDown() {
  // Deregister from the yellow pages
    try {  DFService.deregister(this);
     } catch ( FIPAException fe) {fe.printStackTrace();}
    }
}
```

Fig. 11. Fragment of code for Agents Definition in JADE

♦ ***Defining Agent functionality (agent behaviour) with JADE***

In JADE each functionality or service provided by an agent should be implemented as one or more *behaviours* that can be executed concurrently (Bellifemine et al., 2007). A *behaviour* is basically an event handler that describes how an agent reacts to an event; a behaviour is implemented as an object of a class that extends of: *jade.core.behaviours.Behaviour*, where one must to implement the *action()* method (the operations to be performed when the *behaviour* is executed) and the *done()* method (that specifies whether or not a *behaviour* has completed, through the *Boolean* value it returns).
The *behaviours* help to define the procedures specified by the level 3 nets in a n-LNS model. JADE provides ready to use complex *behaviours* that contain sub-*behaviours* and execute

them according to some policy. Deciding about which behaviour to use depends of the specification given in the level 3 nets; however the structure of these nets usually is close to a Finite State Machine (FSM), and then the JADE´s *FSMBehaviour()* can be used. In the used case of study, the $Net_{3,1}$ (modelled in Fig. 5) exhibits such a behaviour.
Other level 3 net structures may be closer to other *behaviours*, namely *ParallelBehaviour(), SequentialBehaviour(), CompositeBehaviour(),* and *SimpleBehaviour().*
Similarly, WF tasks can be programmed using the same behaviour. Fig. 12 shows the different methods used for defining the *FSMBehaviour()* corresponding to *WF process net* ($Net_{3,1}$ in Fig. 5); such methods declare (register) the transitions and the s*tates* of the equivalent FSM. Each *state* is registered using: *registerFirstState(new Behaviour_Name(), State_Name)* for the first state, and registerState(new Behaviour_Name(), State_Name) for the rest of them. Each *transition* is registered using: *registerDefaultTransition(State1_Name, State2_Name),* and *registerTransition(State1_Name, State2_Name, value)* used when a state have multiple outputs.

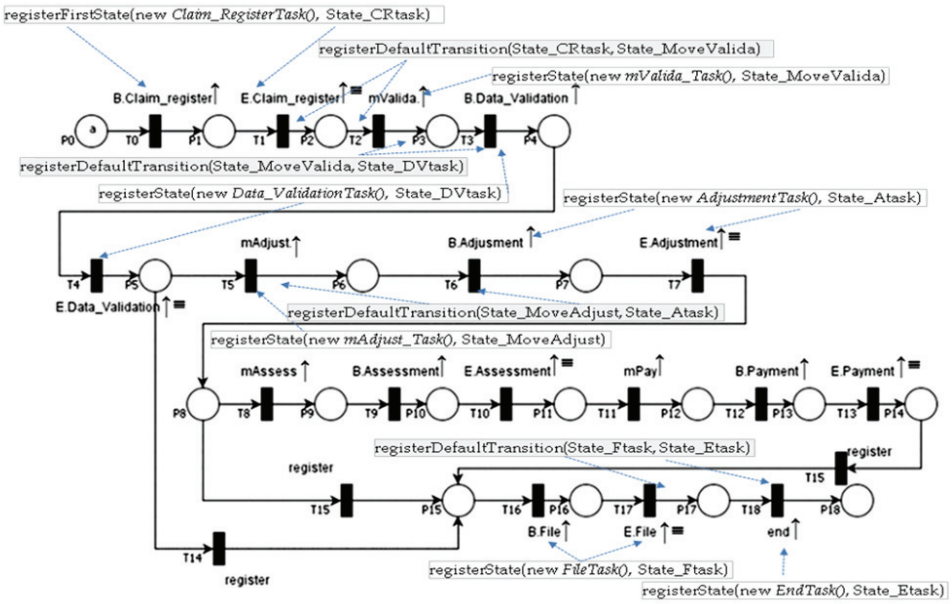



Fig. 12. Some states and transitions registrations for a JADE's *FSMBehaviour()*

```
//Agent Migration to the Validation Department
private class mValida_Task extends OneShotBehaviour{
   private int exit Value;
   public void action(){
            doMove(Host_Location_Name);
   }
  public int onEnd(){
        return exitValue;
  }
```

Fig. 13. Example of a *Behaviour* definition

In this way, each *state* of the *FSMBehaviour()* (for example state *State_MoveValida* from Fig. 12) will be a task or action that the agent will perform; the code for these tasks can be obtained from the nets modelling the involved WF tasks ($Net_{3,2}$, $Net_{3,3}$, ... ); it is placed in each *Behaviour_Name()* class added in the *states* registration. For example *mValidaTask()* behaviour registered in Fig. 12 represents the agent migration behaviour to the Validation Department; the appropriated code for the agent migration that must be added could be as showed in Fig. 13.

- ***Running the agent System with JADE***

In Jade, for executing the agent it is necessary

a) compile it (*javac –classpath* <*JADE-jars*> *AgentClassName.java*), and

b) start it from JADE runtime environment (*java –classpath* <*JADE-jars*> *jade.Boot Agent_Name:AgentClassName*).

## 5. Current research and trends

The proposed methodology for modelling and developing workflow automation systems allows addressing the problem in complex organisations including several business partners (Inter-organizational WF). The technological constraints are limited to holding in the computer equipment the Java virtual machine and JADE supporting the containers that handle the agents.

However new problems appear when the partners share resources and WF processes, and the agents must migrate from one company network to another one (through internet) in order to process a handled case (see Fig. 14). Reliability and security issues must be taken into account during the design of the agent management to provide fault tolerance capabilities. Delays and lose of agents must be distinguished and opportune detected.

Current research addresses these issues as a problem agent population control (Flores-Badillo et al., 2009b). A first protocol has been proposed including the tasks location, loss detection, and recovery of mobile agents; the protocol is supported by the handling of allowed timing in agent mission execution, allowing passive and active termination of agents, opportune localization of agents and orphan detection.

## 6. Conclusions

This work presented a method for developing workflow automation systems. A mobile agent based approach is proposed for obtaining distributed controller software. The first stage consists of a modelling methodology that uses n-LNS, yielding modular and hierarchical descriptions capturing both the agent environment and the agent behaviour. The programming stage allows creating straightforwardly from the obtained model, efficient and extensible software.

The proposed implementation technique uses JADE getting all the JAVA advantages. The mobile agent can interact with other agents for collaborate, negotiate or compete for resources.

Mobile agent based approach is suitable for distributed WF automation because it allows operating within an open platform architecture along local or remote locations.

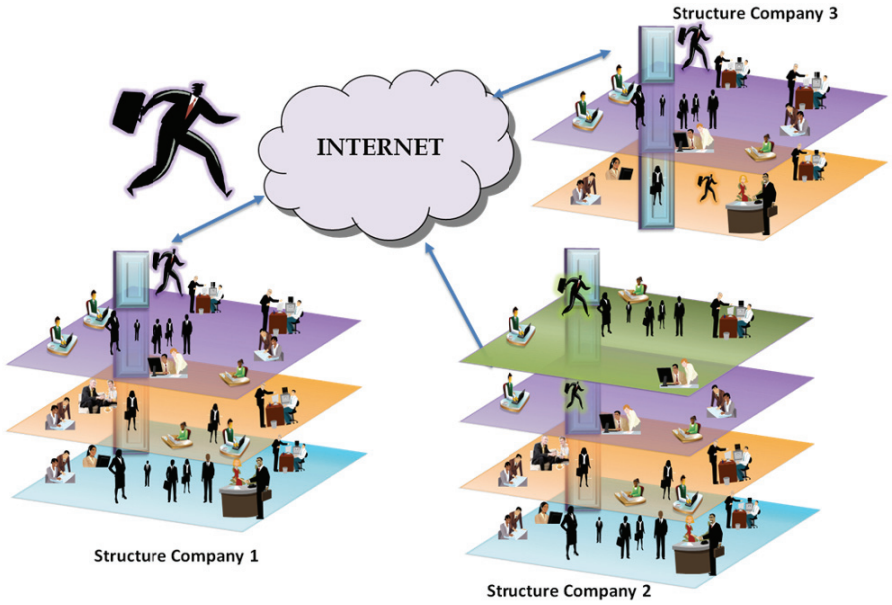



Fig. 14. Inter-Organizational Workflow System

## 7. References

Adam, N.R. ; Atluri, V. & Huang, W.K. (1998). Modeling and Analysis of Workflows Using Petri Nets. *Journal of Intelligent Information Systems.* Vol. 10, Issue 2, March 1998, pp. 131-158, ISSN : 0925-9902.

Bellifemine, F.; Caire, G.; Trucco, T. & Rimassa, G. (2007). *Jade programmer's guide*, JADE 3.5. 2007.

Dicesare, F.; Harhalakis, G.; Proth, J.M.; Silva, M. & Vernadat, F.B. (1993). *Practice of Petri Nets in Manufacturing*, Chapman & Hall.

Eshuis, R. & Dehnert, J. (2003). Reactive Petri Nets for Workflow Modeling. In *Application and Theory of Petri Nets 2003*, vol. 2679, of Lecture Notes in Computer Science, pp. 296-315, ISBN: 3-540-40334-5, Springer Verlag, Berlin.

FIPA (1997). Foundation for Intelligent Physical Agents, Specifications. Available at: http://www.fipa.org.

Flores-Badillo, M.; López-Mellado, E. & Padilla-Duarte, M.(2009a). Modeling and Simulation of Complex Workflow Processes using Multi-level Petri Nets, *International Journal of Simulation and Process Modelling (IJSPM)*, Vol. 4, No. 3/4, pp. 205-214, 2009. ISSN: 1740-2123.

Flores-Badillo, M.; Padilla-Duarte, A. & López-Mellado, E. (2009b). A Population Control Protocol for Mobile Agent Based Workflow Automation, In: *Proceedings of IEEE Int. Conf. on Systems, Man, and Cybernetics,* pp. San Antonio, Texas, USA, October 2009.

Genrich, H.J. & Lautenbach, K. (1979). The Analysis of Distributed Systems by Means of Predicate/Transition-Nets, In: *Semantics of Concurrent Computation*, *Lecture Notes in Computer Science,* Vol. 70, Gilles Kahn (Ed), pp 123-146, Springer Verlag, ISBN : 978-3-540-09511-8, Evian, France.

Hiraishi, K. (2000). A Petri-net-based model for the mathematical analysis of multi-agent systems, *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics,* Vol. 4, pp. 3009-3014, ISBN: 0-7803-6583-6, Nashville, Tennessee, USA, October 2000,

Jensen, K. (1981). Coloured Petri Nets and the Invariant Method, *Theoretical Computer Science,* Vol. 14. North-Holland, pp. 317-336.

Kasturia, E.; Dicesare, F. & Desrochers, A. (1988). Real Time Control of Multilevel Manufacturing Systems using Colored Petri Nets, In: *Proceedings of IEEE Int. Conf. on Robotics and Automation,* Vol. 2, pp. 1114-1119, Philadelphia, Pennsylvania, USA, May 1988. Washington: IEEE Comput. Soc. Press.

Kummer, O. (2001). Introduction to Petri nets and Reference nets, *Sozionik Aktuell*, No. 1, pp. 1-9, ISSN 1617-2477.

Kummer, O. ; Wienberg, F. & Duvigneau M. (2002) *Renew – User Guide*, University of Hamburg, Department for Informatics, Theoretical Foundations Group.

Lakos, C. (1995). From Coloured Petri Nets to Object Petri Nets, In *Proceedings of 16th International Conference on the Application and Theory of Petri Nets, LNCS,* Vol. 935, pp 278-297, ISBN: 3-540-60029-9, Torino, Italy, Springer-Verlag, London, Uk.

Li, J.Q. ; Fan, Y.S. & Zhou, M.C. (2004). Performance Modeling and Analysis of Workflow. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, Vol. 34, No. 2, 2004, pp 229-242, ISSN: 1083-4427.

Lomazova, I. (2000). Nested Petri nets –a formalism for specification and verification of multi-agent distributed systems, *Fundamenta informaticae,* Vol 43, Issue 1-4, August 2000, pp. 195-214, ISSN:0169-2968, IOS Press Amsterdam, The Netherlands.

Lu, R. & Sadiq, S. (2007). A Survey of Comparative Business Process Modeling Approaches. In *Proceedings of 10th International Conference on Business Information Systems BIS 2007,* LNCS, pp. 82-94, ISBN: 978-3-540-72034-8, Poznan, Poland, Springer-Verlag.

Nwana, H. (1996). Software Agents: an Overview, *Knowledge Engineering Review*, Vol. 11, No 3, pp. 205–244.

Reijers, H.A. & van der Aalst, W.M.P. (2005).The Effectiveness of Workflow Management Systems: Predictions and Lessons Learned. *International Journal of Information Management,* Vol. 25, No. 5, pp. 458-472. ISSN: 0268-4012.

Repetto, M.; Paolucci, M. & Boccalatte, A. (2003). A design tool to Develop Agent-Based Workflow Management Systems, In *Proc. Italian Workshop, from Objects to Agents: Intelligent Systems and Pervasive Computing (WOA2003),* pp. 100-107, ISBN: 88-371-1413-3, Villasimius, CA, Italy, September 2003. Pitagora Editrice Bologna.

Savarimuthu, B.T.R & Purvis, M. (2004). A Collaborative Multi-Agent Based Workflow System, *Knowledge-Based Intelligent Information and Engineering Systems KES'2004,* Lecture Notes in Artificial Intelligence (LNAI), vol. 3214, pp. 1187-1193. ISSN: 0302-9743.

Savarimuthu, B.T.R.; Purvis, M. & Fleurke, M. (2004). Monitoring and Controlling of a Multi-agent based Workflow System, In *Proceedings of the second Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation*, Vol. 32, pp. 127-132, Dunedin, New Zealand, 2004, Australian Computer Society, Inc. Darlinghurst, Australia.

Shi, M.L.; Yang, G.; Xiang, Y. & Wu, S. (1998). Workflow Management Systems: A Survey. In *Proc. IEEE International Conference on Communication Technology, ICCT'98*, pp. 1-5, Beijing, China, 1998.

Sibertin-Blanc, C. (1994). Cooperative Nets, In *Proceedings of the 15th International Conference on Application and Theory of Petri Nets, LNCS*, Vol. 815, pp. 471-490, ISBN:3-540-58152-9, Zaragoza, Spain, June 1994, Springer-Verlag, London, UK.

Zisman, M.D. (1977). *Representation, Specification and Automation of Office Procedures*. PhD Thesis. Wharton School of Business, University of Pennsylvania. 1977.

Valk, R. (1998). Petri nets as token objects: An Introduction to Elementary Object Nets, In: *Proceedings of the 19th Int. Conf. on Application and Theory of Petri Nets, LNCS*, Vol. 1420, pp. 1-25, ISBN:3-540-64677-9, 1998, Springer-Verlag, London, UK.

Valk, R. (1991). Modeling Concurrency by Task/Flow EN Systems, In *Proceedings of the 3rd Workshop on Concurrency and Compositionality,* pp. 207-215, GMD-Studien Nr. 191, Gesellschaft f. Mathematik und Datenverarbeitung, St. Augustin, Bonn.

van der Aalst, W.M.P. (1996). Three Good reasons for Using a Petri-net-based Workflow Management System. In *Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96),* pp. 179-201, Cambridge, Massachusetts, Nov. 1996.

van der Aalst, W.M.P. (1998). The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers,* Vol. 8, No. 1, pp. 21-66.

van der Aalst, W.M.P. & Anyanwu, K. (1999) Inheritance of Interorganizational Workflows to Enable Business-to-Business E-commerce. In *Proceedings of the Second International Conference on Telecommunications and Electronic Commerce (ICTEC'99)*, pp. 141-157, October 1999, Nashville, Tennessee.

van der Aalst, W.M.P. & Hee, K. (2002). *Workflow Management: Models, Methods and Systems*. MIT Press. ISBN: 0-262-72046-9, London.

van der Aalst, W.M.P. & Hofstede, A. (2005). YAWL: Yet Another Workflow Language. *Information Systems*. Vol. 30 No. 4, pp.245-275. ISSN: 0306-4379.

Wang, M.; Wang, H. & Xu, D. (2005), The design of intelligent workflow monitoring with agent technology, *Knowledge-Based Systems*, Vol. 18, Issue 6, pp. 257-266, ISBN: 0950-7051.

Weber, M. & Kindler, E. (2003). The Petri Net Markup Language. In *Petri Net Technology for Communication-Based Systems (LNCS),* Volume 2472, Springer Berlin, pp 124-144. ISBN : 978-3-540-20538-8

WfMC. (1999). Workflow Management Coalition - Terminology & Glossary. Technical report, *The Workflow Management Coalition, Document Number WFMC-TC-1011*, available at: http://www.wfmc.org/

Wooldridge, Michael (2001). Intelligent Agents: The Key Concepts, In *Proceedings of the Multi-Agent-Systems and Applications II, (LNCS),* Vol. 2322, pp 3-43, ISBN:3-540-43377-5, Berling, Heidelberg, 2002, Springer-Verlag, London, UK.

Wooldridge Michael (2002), *An Introduction to multiagent Systems*, John Wiley & Sons (Chichester, England), ISBN: 978-0-471-49691-5. 366 pp.

Yan, Y.; Maamar, Z. & Shen, W. (2001). Integration of Workflow and Agent Technology for Business process Management, In *Proceedings of the Sixth international Conference en CSCW in Design*, pp. 420-426, ISBN: 0-660-18493-1, London, Ontario, Canada. July 2001.

# An Application of Petri Nets to e/m-Learning Environments

Cristina De Castro and Paolo Toppan
*IEIIT-CNR, Italian National Research Council*
*Italy*

## 1. Introduction

Thanks to new technologies and devices, e/m-Learning scenarios are quickly evolving and are becoming more and more difficult to control. Besides educational factors and learning contents, a modern e/m-Learning system must also take into account services, kinds of devices and aspects related to the network. As a matter of fact, both learning conditions and activities at disposal depend on many factors, such as the network load of the location from where the user has logged on. Furthermore, not only has the nature of e-learning material changed deeply over time, but so have devices and connection technologies for their fruition, especially in context of mobile learning based on multimedia broadband services.
Learning paths themselves are developing: teaching is not an univocal process anymore, with fixed steps and tasks, since modern technologies make many alternative choices possible. Moreover, such tasks can be fulfilled or not on the basis of personal and environmental conditions. For instance, a lesson in videoconference can be followed or not depending on the kind of device and network load.
In more detail, as far as didactic motivations are concerned, methodologies are changing. The old "linear" teaching paths are evolving into more complex shapes, where parallel and cooperative activities come abreast of conventional ones. In addition, traditional tasks can turn into many kinds of activities that can be fulfilled using different digital formats and devices.
In the same way, modern communication technologies affect learning deeply from many points of view. One of the most important aspects is that they increase interaction and cooperation and reduce – if not cancel - problems related to distance. This allows to make use of many learning contents, remote lectures, cooperative work and remote use of instruments.
As far as m-Learning is concerned, it was born from the evolution of both mobile devices and network access technologies and made e-Learning independent from location and device. This leads to great advantages but also to many new problems. For instance, not every kind of content can be used with any kind of device. In order to make a content available in an m-learning context, it is necessary to design information properly, adapt it and scale it with respect to different devices.
Moreover, this scenario requires a constant monitoring of the user's conditions: as a matter of fact, some tasks can be feasible or not depending on network load conditions and devices

as well. This means that such factors become an integrant part of teaching modalities and offers.
On the one hand, this variety can turn out to be a positive issue in many situations: for instance, if network overload makes a videoconference impossible, the student can be suggested to make exercises on his own, etc. In the same way, broadband services allow to cooperate or use didactic structures and instruments otherwise inaccessible.
On the other hand, controlling the evolution of educative paths is complicated by this multiplicity of factors.
In this work, which proceeds from (De Castro & Toppan, 2008a; De Castro & Toppan, 2008b, De Castro, 2009), these features and their relationships are analysed and an integrated layered architecture is consequently proposed which aims at reaching a good compromise between quality of learning and studying conditions.
The proposed architecture consists of a user interface, a module for providing services, a decision engine for evaluating students' improvements and consequently deciding learning paths and a database storing learning and assessment contents. Due to the use of different devices and to the presence of different network conditions, the database stores contents in scaled formats, or ready to be scaled on the fly.
The whole system is controlled by a Petri Net defined on the basis of the above factors. The definition of the Petri Net by means of such dissimilar kinds of constraints is one the main novelties of the proposed approach.
The research activity described in this chapter originates from the experiences of the Teledoc2 project and Cooperative Telemeasurements (www.teledoc2.cnit.it/Teledoc2/home.htm), which are briefly summed up in Section 2.
Section 3 is devoted to the main architecture, its components and the data flow among them.
In Section 4 constraints are discussed about the user's activities, cooperation and network, in order to understand which factors affect the Petri Net.

## 2. The Teledoc2 Project and Cooperative Telemeasurements

The Teledoc2 project was financed by the Italian Ministry of Education, Universities and Research (MIUR) and carried out by CNIT (National Inter-University Consortium for Telecommunications). The project was active during 2003-2005 and aimed at building a complete, multimedia, interactive and fully-featured online learning service for ICT researchers and PhD students of Italian Research Centres, provided they were branches of CNIT.
As in many e-Learning systems, the main components were a web-based user interface, the network infrastructure, the e-Learning software and the courses.
Teledoc2 was planned for the diffusion of scientific and technological culture in the ICT field, and meant to allow students to attend specialist courses in the forefront of research. Such courses were at disposal broadcast from different Italian Research Centres.
The project aimed at building an efficient service of distance learning of third generation: the courses could be attended in real-time, just connecting to the CNIT proprietary packet communication network and using simple Personal Computers running a custom multimedia application.
The learning strategy, therefore, aimed at recreating a live virtual classroom environment, with a real-time face-to-face relationship and high levels of interactivity among users.

Furthermore, the concept of virtual classroom had to be extended to "ubiquitous distributed service", with no kind of limitation to the user's position.
The whole learning system was designed to be complete, efficient, user-friendly and characterized by fixed and suitable QoS levels.
In order to guarantee reliability, CNIT used all its experience in the ICT field both in the backbone connections and in the local ones.
One of the key network requirements was the support of multicast, a strong element of innovation and originality if compared with most of the other distance learning systems. CNIT decided to build multicast-enabled networks because this way of transmission seemed particularly appropriate for online learning services like Teledoc2.
These applications, in fact, required two basic network requirements: on the one hand, they needed one-to-many and many-to-many communications to reach all participants and to promote interaction; on the other hand, they needed high bitrates, since they had to transmit audio and video of fixed quality.
In context of this project, the WiLab (Wireless Communication Laboratories) research unit of CNIT and IEIIT/CNR (Institute of Information, Electronics and Telecommunications - Italian National Research Council) at the University of Bologna carried on further activities.
In particular, the definition, planning and development of the paradigm of "distributed cooperative telemeasure". The "Telemeasurement" concept (meant as remote control of instrumentation belonging to one single workbench) was described in (Roversi et al., 2004), where this methodology was applied to characterize communication systems based on instruments and programmable platforms with Digital Signal Processors (DSP).
The concept of telemeasurement was enhanced with the introduction of "cooperative telemeasurement" (Roversi et al., 2005), in which various resources are distributed in a network of different laboratories and can cooperate to set up augmented experiments. This extension of the telemeasurement concept wanted to increase measurement capabilities, since the user could access different remote laboratories and use remote devices without having all the needed instrumentation locally. The definition and implementation of this platform involved signal processing, management of distributed resources, development of aggregated user interfaces, transport of signals for measure, innovative remote controls, protocols for gaining access and control of specific laboratory instrumentation, prototypes for testing the schemes designed on the field and a proper communication network.
As it will be explained in the following, after the completion of such activities, research themes evolved into the definition and management of contents, access methodologies and network optimisation.

## 3. Main Architecture and Petri Net-Based Approach

The e-Learning model used in Teledoc is represented in Fig. 1, where users access learning material and services, such as slides and videoconference.
This model evolves into the Cooperative Telemeasurement paradigm (Fig. 2), where many laboratories put their instruments at disposal, and augmented experiments can consequently be performed.

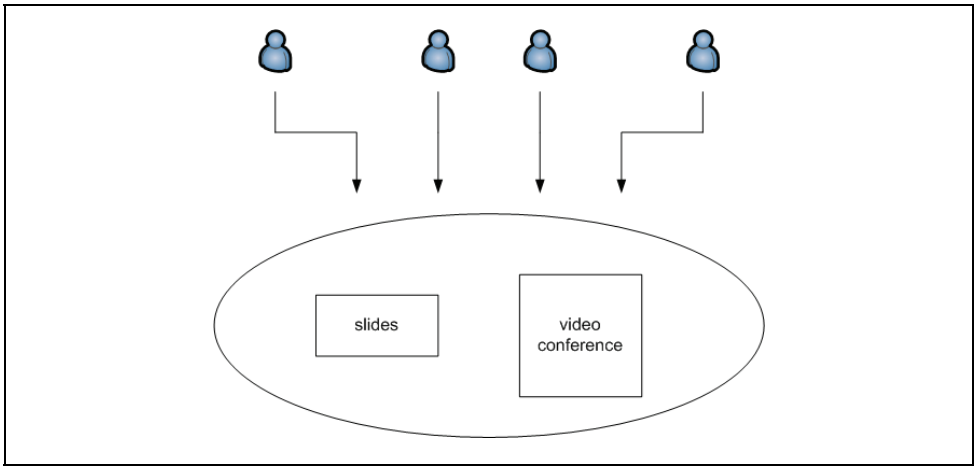



Fig. 1. The Teledoc2 e-Learning model

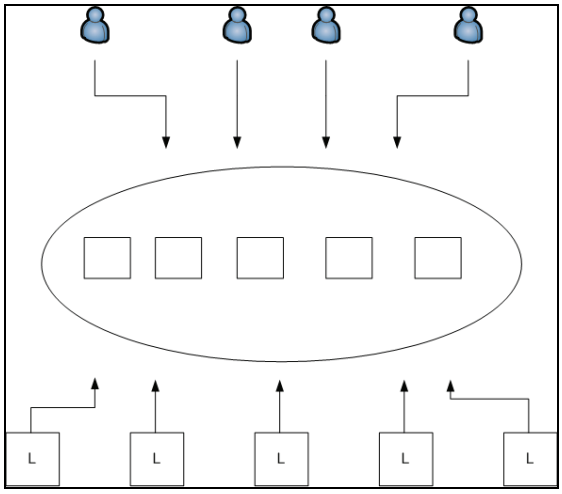



Fig. 2. The Cooperative Telemeasurements e-Learning model

The current viewpoint is to represent the evolution of the whole system, taking further factors into consideration: network conditions (network load and kind of device), didactic prerequisites and cooperative activities. Such a system and the learning paths can be controlled by means of a Petri Net.

In such a complex environment, technologies and feasibility factors play a fundamental role. For instance, videoconference can be feasible or not depending on the device and network load, so that the videoconference material may need to be scaled to audio-only format and adapted to present conditions.

In the same way, a user can be ready or not to take part to a given activity, considering at least two factors. First, he must have fulfilled all the necessary prerequisites; if not so, he must be stopped and proposed other activities. Second, the user must be ready, and so must his companions.
This approach can be depicted as in Fig. 3: the effective fruition of a content depends on many factors, which influence resources scheduling and data format as well. In more detail, if content n is requested, device and network load will indicate whether the request is feasible or not and thus influence scheduling. If the user requests to access content 1 with a PDA, the system will scale such content and adapt it to the device, provided the device is supported for such content. As a matter of fact, not every content can be adapted to any type of device.
The same applies to network load: if, between device and network, the bottleneck is the latter, contents will be adapted in consequence. As a matter of fact, accessing multimedia contents requires a high bandwidth; in case this is not available, contents will have to be scaled, for instance on the fly (De Castro & Toppan, 2008b; Donzelli et al., 2006).

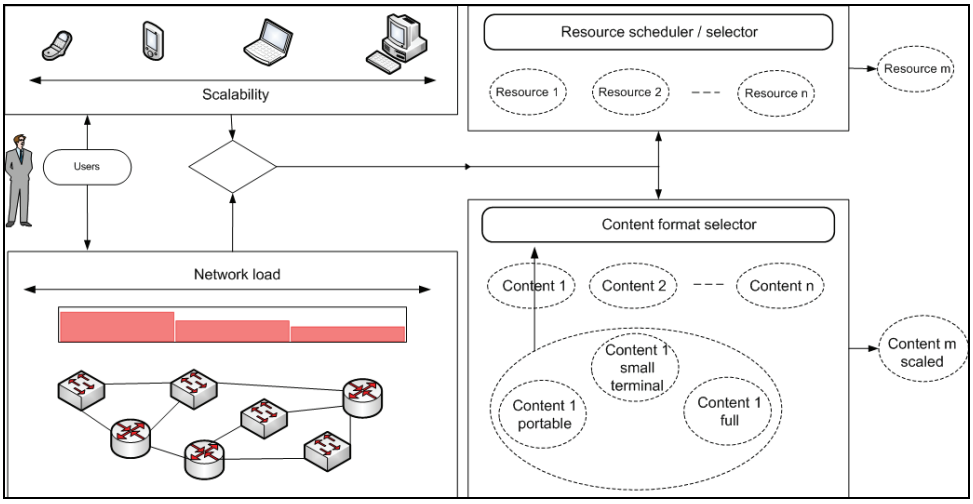


Fig. 3. Scheduling, scaling and data adaptation

In Fig. 4 all the discussed variables are represented: factors related to technology, didactic prerequisites, readiness to cooperate. Given such issues, the system will return the feasibility to use a given content in an appropriate format.
Let us now see how the system and its components can be represented from a single user's viewpoint (De Castro & Toppan, 2008b).
First of all, the overall architecture and didactic prerequisites are discussed. The proposed architecture is presented and its components discussed separately.

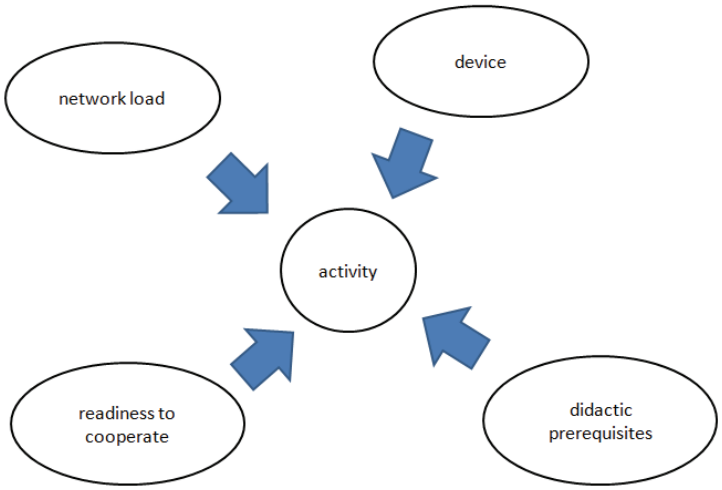



Fig. 4. Factors affecting the feasibility of an activity

The system can be described by means of the 5-layered architecture in Fig. 5, which is increasingly enriched from the left to the right.
It consists of five main blocks: the Query Layer, the Data Adaptation module, the Testing and Path Decision block, the Services and the Database.
The Query Layer allows the user to interact with the system, accessing services and contents scaled on the basis of the user's technology.
The Data Adaptation layer allows to adapt the data format and consequently optimise response time.
Given a knowledge level, the Testing and Path Decision module returns learning steps. In this module the first approach to the use of Petri Nets is defined. In this context, variables are didactic prerequisites, whereas in Section 4 the architecture of such module will be augmented taking into account network factors, as well as cooperative work ones.
The Services level indicates which multimedia broadband services are at disposal, such as booking of experiments, etc.
The Database level highlights the presence of three groups of information: testing and assessment material; learning material; user's data, bookings and access to services.
In Fig. 5, the vertical axis is a first classification; on the right, all the components are expanded and represented with respect to their interaction and to the data flow which takes place among them.
The main purpose and features of each block and its sub-modules are described in the following.
The User-System Communication Interface must carry out the following tasks:

1. receive the user's requests (initial target and successive ones) and forward them to the Testing Module. All such requests are meant to be events that must be notified to the system;
2. receive the suggested assessment tests from the Testing Module and send back the results;

3. receive the suggested learning steps from the Path Decision Module and allow the user to access the studying material;
4. manage access to auxiliary services;
5. transform all such requests and answers on the basis of the user's devices (mobile, PDA, laptop, etc.); more generally, make the user and the system communicate on the basis of the user's device *("Device Interpretation")*;
6. adapt learning data to the user's device and access technology, so as to meet the learner's requirements and optimise the overall process *("On the Fly Data Conversion")*.

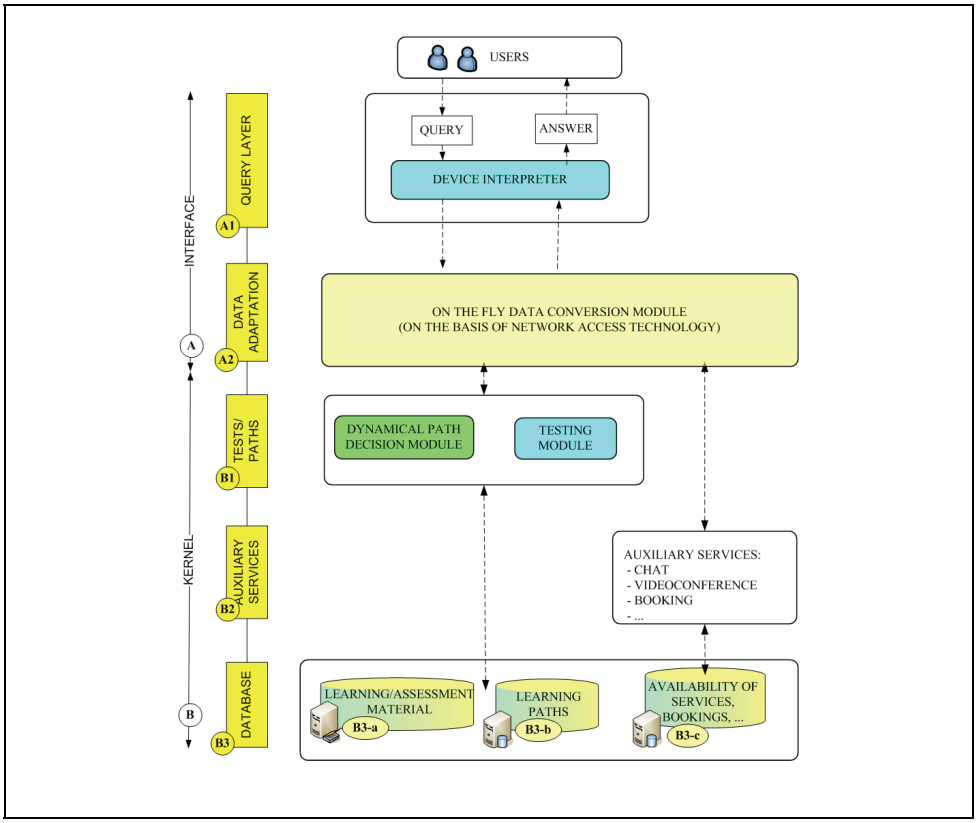


Fig. 5. Main components of each layer

Note that task 5 concerns the management of queries and data with respect to the user's *device*, and is accomplished by the Query Layer. Task 6 concerns data optimisation on the basis of *network access technology* and is fulfilled by the Data Adaptation module.
When talking of data format, two different - even not independent - operations must be distinguished. First, data transformation due to the type of device (*Data Interpretation* on the basis of device). Second, data format optimisation due to the type of network connection (*Data Adaptation*).

As for Data Interpretation, the system must communicate on the basis of the user's device, so the user's data must be converted in a format that both the front-end to the system and the database can understand. This process will last the whole lifespan of the learning process and can easily be done by means of XML conversions.
As a matter of fact, this is a straightforward, general and effective way for exchanging data between heterogeneous environments.
As far as the Data Adaptation module is concerned, in order to optimise response time in each scenario and in perspective of m-Learning extensions, the following guidelines must be taken into account: first, it is essential to reach the best trade-off among the user's actual needs, quality/quantity of data and response time; second, information must be adapted to available technologies (PDAs versus laptops, etc.) (Bronson et al., 1993; Yuang et al., 1994; Caouras et al., 2003).
Developing such a system involves at least two aspects: first, information must be represented at different levels using different formats. Second, an access methodology must be designed for filtering data on the basis of the above criteria. This is accomplished by the On the Fly Data Conversion Module, which retrieves data in its original format and scales it so as to adapt it to the kind of network access technology and device.
The main role of the Path Decision and Assessment Module is to be aware of the user's aims, check his learning levels, and consequently define a tailored studying path.
The main idea is the strict interaction between the Path Decision/Assessment Layer and the database. As a matter of fact, the database stores both assessment and studying material which is selected from the database on the basis of the user's goals and actual achievements.
The database schema can be represented by means of a network of *issues*, *levels* and *prerequisites*. Consider issues $I_l$ and $I_m$ and suppose they are related (such as derivatives and integrals) and meant to be faced at a given level (such as a course of Mathematics at a high school).
For the sake of simplicity, suppose $I_l$ and $I_m$ can be considered steps of the learning process.
Suppose that facing step $I_m$ after step $I_l$ requires prerequisites $p_1$, $p_2$, ..$p_k$.
This kind of algorithm can be represented by means of a Petri Net (Chen et al, 2001; Li et al, 2005) which acts as a traffic light between a learning step and the successive one.
In this approach, the Petri Net's places are knowledge to be tested (prerequisites), and its transitions are studying phases. A transition is enabled if all the required prerequisites have been fulfilled (Fig. 6).
Since the system is adaptive, contents will be put at the user's disposal on the basis of his device and access technology.
As far as the underlying e-learning information system is concerned, its architecture has been designed keeping in mind that two kinds of information are involved:

(i) e-learning and assessment material, which is not meant to be frequently updated (named *static data*);
(ii) personalised learning paths and assessment results, which are time-varying (named *dynamic data*);

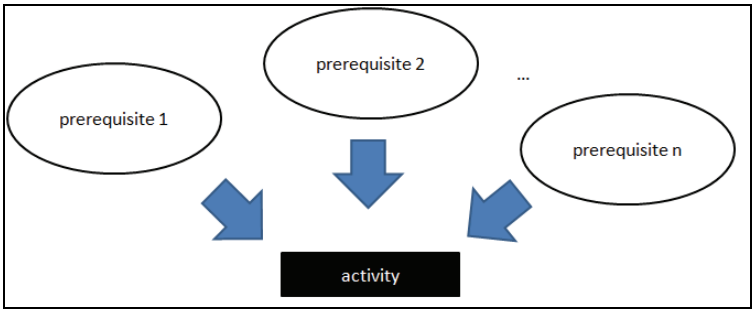



Fig. 6. Didactic prerequisites and activities

It must also be noticed, and considered as a requisite, that e-learning information can be represented by means of a hierarchy of subclasses. For instance, from a subject to its specific issues.

These requirements suggest the use of a hybrid database structure for data storage: an LDAP directory service (Howes et al., 2003) for static data and a relational DBMS for dynamic information.

LDAP provides both a model and an implementation tool which is particularly suitable for web-based e-learning applications, both from the data representation viewpoint and for an efficient web-based access.

As a matter of fact, it is scalable, extendable and optimised for reading operations, so it is particularly suitable for static data. It also supports standards and interfaces of many multimedia broadband applications and integrated access to e-learning services.

Another important feature is that LDAP represents information by means of a hierarchy of classes using very flexible schemata. In the considered environment, this implies at least three advantages.

First, the knowledge that a person acquires on a specific subject can be organized in an LDAP tree as follows: the $n^{th}$-level class describes the *subject* in general; the $n^{th+1}$-level classes represent related subjects, issues and related issues, documentation and assessment material, prerequisites, learning paths and assessment tests.

Learning and testing material, as well as paths and tests, are divided in as many subclasses as the number of target levels provided for. In this way, known such level, the middle layer can access the correct material.

As for the second advantage, LDAP was built for the integration of distributed environments, so it also suits the distributed location of documentation very well. As a matter of fact, for applications such as international remote education, e-learning information is distributed by nature.

The third advantage concerns schema management. The schema of the e-learning database is likely to be modified or augmented during its life cycle, for instance due to the addition of new kind of media or information described by means of different properties. A relational system, in traditional settings, does not allow efficient schema revision. Such operations involve high costs in terms of redesigning existent schemata, reloading data and verifying that original constraints and relationships on data are preserved. LDAP, on the contrary, offers high flexibility in modifying data structures.

As far as the dynamic part of the database is concerned, it mainly concerns the time-varying personalised learning paths. In more detail, the dynamic database stores information about the user and his learning phases, such as targets, suggested steps and actual achievements.
Other information involve auxiliary services and their booking.
In this case, an SQL database is more suitable. As a matter of fact, such models are optimised for reading/writing operations and time-varying data.
The connection between the LDAP and the SQL databases are LDAP object identifiers which, as identifiers of subjects, issues, etc., are used as key information in the definition of dynamical paths and assessment tests results. They are also used in the joint navigation of LDAP and SQL data.

## 3. Factors Affecting the Petri Net: Network, Didactic Prerequisites, Cooperation

In this section (De Castro, 2009), the previous approach is revised in order to represent learning paths and their control in advanced e/m-Learning systems. Such environments put heterogeneous data and services at disposal, and different connection technologies can also be adopted.
As already discussed, due to such diversity, the studying paths are not univocal: for instance, some exercises can be made first and cooperative work with fellow students afterwards or viceversa (“*alternative paths*”). Another important consequence is that learning processes have constraints dictated by download time, the network status, other activities and synchronisation with fellow students.
The proposed model proceeds from such observations, states some rules that the learning processes must obey in order to take place regularly and its core is based on graphs and Petri Nets.
Both *contents* and *access modalities* influence the process of learning and, thus, the evolution of the learning paths over time, so they must become an integrant part of the control system.
Some observations about such features will be made in the following, in order to understand their role in the learning process and consequently discuss the basics of the discussed approach.

### 3.1 Carrying Out Activities: Heterogeneity and Synchronisation

As far as functionalities and contents are concerned, it is becoming more and more important to involve the user actively, and adapt the studying process to his needs. It must be noticed, though, that such requirement implies a tailored and increasingly complex definition of learning paths, assessment phases, as well as an efficient control of the whole course of action.
As for contents, modern learning systems put a wide range of heterogeneous data and activities at disposal, such as videoconferencing, remote laboratory, chat for cooperative work, textual exercises, tests, storytelling and so on.
Some observations can consequently be made that will be of help in the definition of the learning process, by giving birth to constraints which will be progressively added to a first, rough version of the overall architecture.

1. The data format of learning material is intrinsically heterogeneous, ranging from plain documents to multimedia files;
2. Some activities need to be synchronised, while other do not. For instance, an interactive remote laboratory experience must obey precise timetables, and the same applies to cooperative work with fellow students or examinations. On the contrary, some laboratory training or exercises can be made with a certain degree of independence from other people and other tasks;
3. There can be many ways to achieve the same target, each corresponding to different data formats and technologies. For instance, a lesson can be followed using on-line videoconference or studying the course's material (slides, etc.).

The main architecture can be roughly revised as in Fig. 7, where the user asks to access the system and perform an activity (*query*). On the basis of the *synchronisation constraints*, he can be allowed to carry this task out or not. Furthermore, his background and the *didactic prerequisites* for the requested activity are analysed and he is submitted to a testing phase (*assessment*) before accessing new *learning steps*. The whole architecture lies on a database storing learning contents and their prerequisites, as well as assessment material and is guided by a "*Control Module*".

As for network access modalities and m-learning facilities, they should be as diversified as possible, on the basis of the student's aims, ties and timetables, as well as his location and kind of technologies at his disposal. All such factors are decisive in order to reach a good compromise between quality and studying conditions. In more detail, the following scenarios are taken into account:

- UMTS: this can be the case of accessing the system through PDAs, for instance if some exercises must be made;
- DSL: it is quite typical when users access the system from home;
- WiFi: this is quite common within a campus, in study lounges and libraries;
- wired/fiber: it is generally the case of universities, for instance within laboratories

Some rules are now stated, which are the counterparts of (1)-(3) with respect to connection technologies, and mean to be of help in the design of the Control Module of the proposed architecture.

Among all the activities that a student can undertake, only those can be performed that obey to the following constraints:

a. A connection technology can not always be chosen freely; as a matter of fact, there are tasks that require a minimal connection speed. For instance, a remote laboratory experiment in cooperative modality can only be fulfilled using at least a good DSL technology. On the contrary, simple exercises can be made using a PDA;
b. Only those activities must be made available which are:
    i. compatible with the student's current technology;
    ii. compatible with the e-learning plan;
    iii. asynchronous or
    iv. can be synchronised with other tasks;
c. If a task can be faced by means of different technologies, data must be organised in such a way that the format is adapted to the technology itself (e.g. video vs text), or

a choice can be made between material of different nature (e.g. lesson on videoconference vs slides).

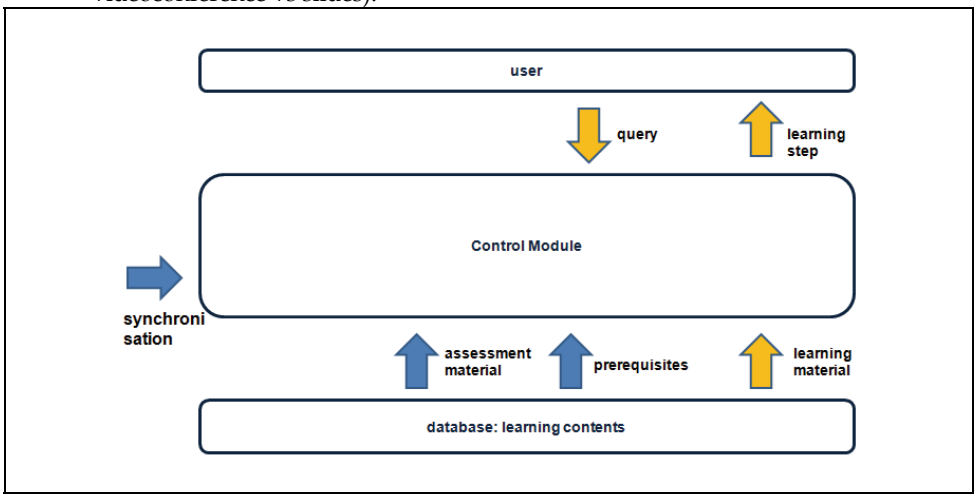



Fig. 7. Constraints on the basis of contents and cooperative work

These rules define alternative paths and state whether an activity can be currently carried out or not by a given person who is using a given network access technology and device.
Fig. 8 represents the further parameters that must be added to the architecture in Fig. 7: the synchronization factor had already been represented, but its role has been more precisely defined. The *device* and *minimal connection required* have become new input data for the Control Module.

### 3.2 Control System

First of all, the graph structure will be defined, since it is the core of the e/m-Learning model. Afterwards, the whole architecture will be completed and its components represented and discussed, first with respect to I/O, then with respect to the overall process.
First of all, let us give the following definition:
**D1 -** due to (1-3) and (a-c), learning paths are time-labelled graphs, whose nodes' are tasks associated to the following information: data, data format, minimal connection technology required, possible alternative contents of different format, prerequisites.
**D2 -** Compatibility controls can be defined by the edges themselves: there is an edge from $N_i$ to $N_j$ at time t if $N_i$ and $N_j$ obey (a-c). This can be achieved by guiding the graph through an appropriate Petri Net.

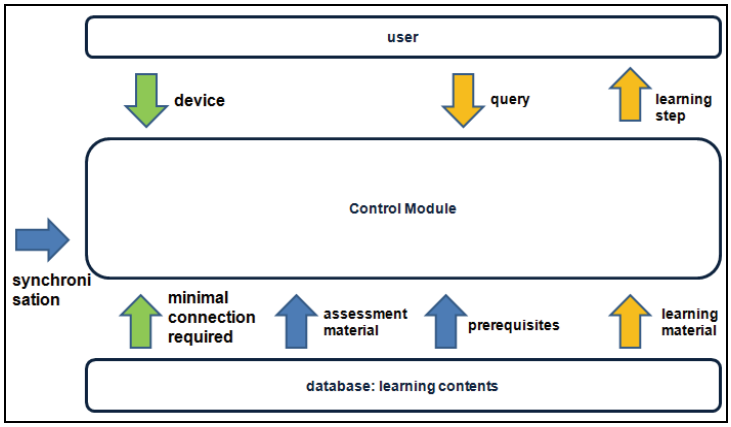



Fig. 8. Constraints on the basis of access technologies

**D3 -** Network load is a constraint itself in the progress of a learning activity. On the basis of the user's IP, his location is known and the available bandwidth determined. Only those tasks will be made available whose minimal bandwidth required is compatible with such conditions.

Taken all such factors into consideration, the proposed architecture consists of four layers. These components and their interaction are described in the following (Fig. 9).

**User-System Communication Module**: this module makes the user and the system communicate and, in particular, through different devices. The user sends his target to the Network Analyser Module by means of a query and is localised through his IP. The device is also determined.

**Network Analyser:** the network load and available bandwidth of the user's current location are determined on the basis of his IP and so is his device. All such data are forwarded to the Petri Net Module.

Petri Net Module (Control Module): this layer is a Petri Net which guides access to activities: given the user's query and all the constraints discussed above, this module returns a learning step among all the available ones. A transition is enabled if (a-c) are met, if the available bandwidth is greater or equal to the minimal bandwidth required and if the user has passed the necessary assessment tests.

**Database layer:** the data repository is a multimedia database whose contents are represented through a hierarchy of subjects and tasks. Each task is associated to the following information: data, data format, minimal connection technology required, possible alternative contents of different format, prerequisites and assessment phases.

### 3.3 Places and Transitions

As discussed above, the Petri Net will have three kinds of places: network prerequisites, didactic prerequisites and cooperation prerequisites. In order to define the generic places-transitions block which, from prerequisites, guides access to activities, the following considerations can be made.

Let us consider a single activity, available in more scaled formats. Each activity has at most three prerequisites and some alternatives. Suppose it is available in two formats. It gives birth to two different parallel transitions having the same input places. On the basis of marking and weights, the one or the other will be feasible. For instance, given the same didactic and cooperative prerequisites, if the device is a PDA, the only activity will be at disposal which requires less bandwidth.

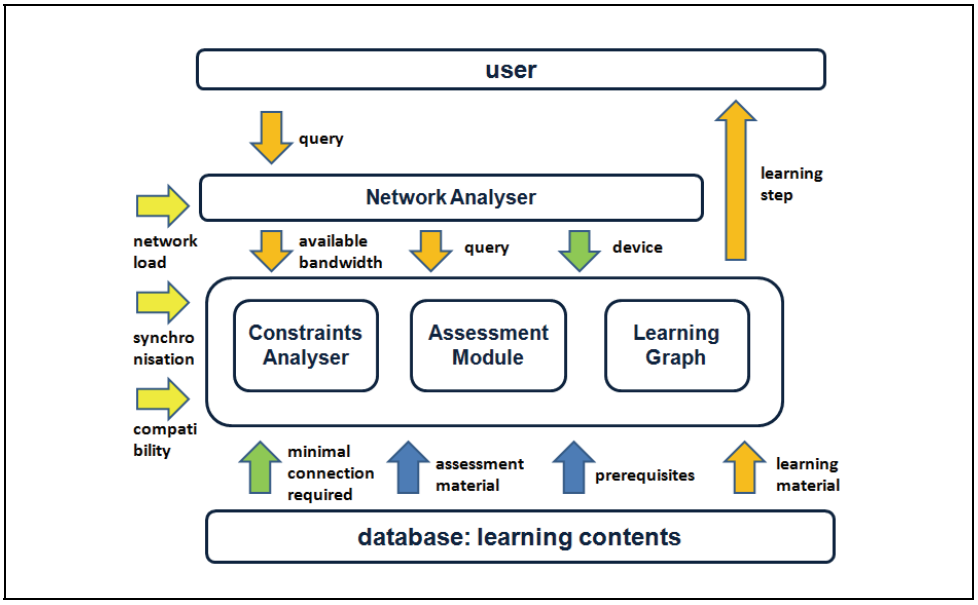



Fig. 9. Main architecture

The second viewpoint is that of alternative activities: the same places are connected to different kinds of activities. Each of such activities will be at disposal in many different formats, as above. Suppose the previous activity has one alternative, at disposal in one format.

The above situation is depicted in Fig. 10; the Petri Net will have as many blocks of such type as the number of activities at disposal. Not every transition will have all the prerequisites and will be at disposal in every format possible (there are activities that can not be scaled). In short, the generic block is made of three places connected to many transitions, each representing the same activity in different format or a different activity.

As for the marking of network places, it is a function of two variables: network load and device. Its value is the bottleneck between the two.

The didactic prerequisites places contain all the issues necessary for facing a new activity. Marking can be thought as a sort of result of the testing phase, leading to an activity or a different one. If transitions represent activities of scaled formats, the weighting of didactic prerequisites will be the same. It will be different in the case of different activities.

As for cooperation places marking, it is simply a boolean function representing readiness of participants or not.

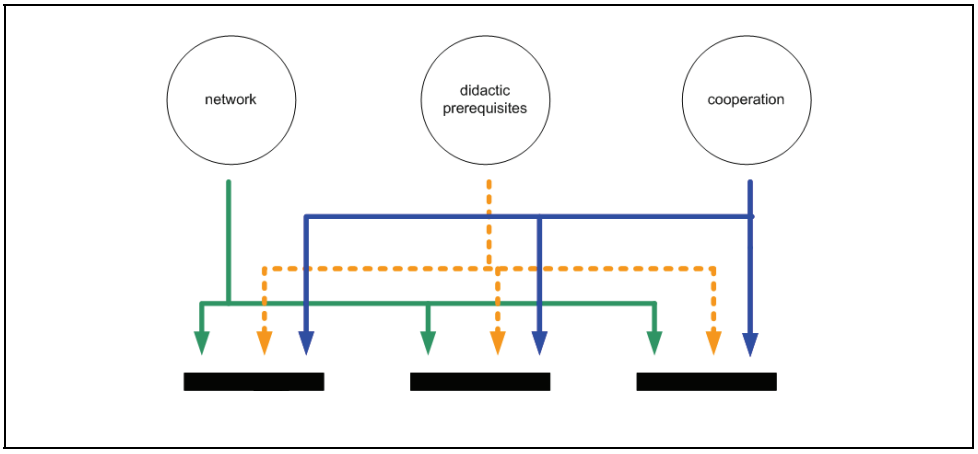



Fig. 10. Places and transitions

## 6. Conclusion

Modern e/m-learning systems must take many factors into consideration, such as functionalities offered to the user, organisation of contents, service fruition modalities, as well as network load. As a matter of fact, not only has the nature of e-learning material changed deeply over time, but so have devices and connection technologies for accessing services. Furthermore, in context of mobile learning based on multimedia broadband services, the network load of the location from where the user has logged on must also be taken into account.

In this work, these features and their relationships were analysed and an architecture was consequently proposed which aims at reaching a good compromise between quality of learning and studying conditions. The whole system is controlled by a Petri Net defined on the basis of the above factors. The definition of the Petri Net by means of such dissimilar kinds of constraints is one the main novelties of the proposed approach.

Future work will be devoted to the application of the discussed architecture to the Teledoc2 and Cooperative Telemeasurements environments, which aim at defining specialised learning paths in Information and Communications Engineering for PhD students and accessing laboratory experiences and equipment remotely through multimedia services.

## 7. References

De Castro, C. & Toppan, P. (2008a). Dynamical Target-Oriented e-Learning Networks, *Proc. of International Conference on Technology, Education and Development (INTED 2008)*, Valencia, Spain, March 2008.

De Castro, C. & Toppan, P. (2008b). An Architecture for Interactive Target-Oriented e-Learning Systems, *Proc. of 50th International Symposium ELMAR-2008*, Zadar, Croatia, September 2008.

De Castro, C. (2009). An e/m-Learning Architecture Based on Alternative Studying Paths, *Proc. of International Conference on Technology, Education and Development (INTED 2009),* Valencia, Spain, March 2009.

Roversi, A.; Conti, A.; Dardari, D. & Andrisano, O. (2004). Telemeasured Performances of a DSP based CDMA Software Defined Radio, *Proc. of International Conference on Engineering Education and Research (iCEER 2004)*, Czech Republic, June 2004.

Roversi, A.; Conti, A.; Dardari, D. & Andrisano, O. (2005). A Web-based Architecture Enabling Cooperative Telemeasurements, *Proc. of Thyrrenian International Workshop on Digital Communications 2005*, Sorrento, Naples, Italy, 2005.

Donzelli, C., Fontana, C., Ravaioli, A., Toppan, P., Patella, M., De Castro, C. (2006). An LDAP/SQL-based Architecture for Broadband Services, *Proc. of IASTED Int. Conference on Communication Systems and Applications (CSA 2006)*, Banff, Canada, pp. 96-101, July 3rd-5th 2006.

Bronson, G., Pahlavan, K. & Rotithor, H. (1993). Performance evaluation of wireless LANs in the indoor environment, *Proc. of 18th Conference on Local Computer Networks*, 452 – 460, 1993.

Yuang, M.C. & Hsu, S.J (1994). LAN protocol modelling and performance evaluation, *Communications, ICC 94, SUPERCOMM/ICC '94, IEEE International Conference on Serving Humanity Through Communications*, 2, 685 - 689.

Caouras, N., Freda, M., Monfet, F., Aldea, V.S., Naeem, O., Tho, L. & Champagne, B. (2003). Performance evaluation platform for xDSL deployment in a complex multi-segment environment, *Proc. of Canadian Conference on Electrical and Computer Engineering, 2003*. IEEE CCECE 2003, 1, 61 – 64.

Chen, C. S.; Ke, Y. L. & Wu, J. S. (2001), Coloured Petri nets approach for solving distribution system contingency by considering customer load patterns, *IEE Proc.-Gener Transm. Distrib.* Vol. 148, No. 5, September 2001.

Li, D.; Cui, Y. & Xu, K. (2005). *Improvement of Multicast Routing Protocol Using Petri Nets*, Slezak et al (Eds.), RSFDGrC 2005, LNAI 3642, pp. 634–643, 2005, Springer-Verlag Berlin Heidelberg.

Howes, T.; Smith, M. & Good, G. (2003), *Understanding and Deploying LDAP Directory Services*, Addison Wesley, 2^ ed.

# Petri nets-based Models for Web Services Composition*

Huaikou Miao[1,2] and Tao He[1]
[1]School of Computer Engineering and Science, Shanghai University
[2] Shanghai Key Laboratory of Computer Software Testing and Evaluating
[1,2] P. R. China

## 1. Introduction

Web services receive significant research recently from both academia and industry due to its broad applications and flexible architecture supporting recomposition and reconfiguration lately. Research on semantic web brings new energy to Web services; combining semantic web with Web service will be a kind of intelligent Web service which is the trend of Web service development. An individual service, whose functionality is limited, cannot meet practical application's needs, so composing Web services and generating a new value-added Web service, more functionalities can be provided and the potential of Web services can be showed. Composing Web services is to find available Web services, integrate interfaces between Web services, combine several autonomous Web services together in term of requirements of applications, and provide a more powerful composite service.

In order to allow flexible automation and composition of semantic representations of Web services, OWL-S (OWL for Services) was proposed. OWL-S service descriptions (e.g., the service process model) provide the needed information for a priori analysis and verification of service invocations and compositions. Due to the lack of formal semantics in the OWL-S specification, McIllraith and Narayanan use Petri nets to test and verify the composition of Web services based on OWL-S (Martin D.etal.2004). Some researches are currently being done on automated provision and reasoning about Web services. (Cordoso,J &Sheth, A. 2002)

Rachid Hamadi (Rachid Hamadi, Boualem Benatallah, 2003) proposed a Petri nets-based algebra for composing Web services; any service that is expressed using the algebra constructs, can be translated into a Petri nets model. This model without semantics belongs to the composition based on service level, particle is too large. Zhang Jia (Zhang Jia et al.,2004) presented a WS-net model based on colored Petri nets, described Web service components from three aspects which are the interface net, the interconnect net and the

* This work was supported in part by a grant from National Natural Science Foundation of China(NSFC) under grant No. 60673115; National High-Technology Research and Development Program(863 Program) of China under grant No. 2007AA01Z144; National Grand Basic Research Program(973 Program) of China under grant No. 2007CB310800; Research Program of Shanghai Education Committee under grant No. 07ZZ06; Shanghai Leading Academic Discipline Project, Project Number: J50103.

interoperation net. This model that supports the object-oriented paradigm and the component-based concept, is convenient to verify and stimulate service composition, but preconditions and effects of services are not considered.
Narayanan et al. (S.Narayanan &S.Mcllraith, 2002) defined semantics of DAML-S atomic processes in terms of a set of situation calculus axioms, and every basic service was represented by a situation calculus formula whose operational semantics was provided using Petri nets. Moreover, verification of the composite service can be realized by reachability of Petri nets. Nevertheless, it did not discuss about composite planning and process model. S. Narayanan and S. A. Mcllraith proposed a model based on Petri nets to specify a Web service of OWL-S (J.P. Thomas et al., 2003). They supposed that the service was composed of several atomic services, and the supposition was not suitable in open-environment composite service. Furthermore, the model was not described clearly in their proposition.
This chapter mainly discusses how to model and analyze the composite processes of semantic Web services using Petri nets. In this model, input, output and precondition are represented through different kinds of tokens; effect is represented by the change of the token number during firing the transition. The composition of two or more services generates a new service providing both the original individual behavioral logic and a new collaborative behavior for carrying out a new composite task. A composite service consists of a collection of Web services related by data and control flow. Composition models can be described unambiguously and composite processes can be analyzed and verified conveniently.

## 2. Background

### 2.1 OWL-S

OWL-S is defined as a W3C standard to provide a computer-interpretable description of the services, service access and service composition using OWL ontologies. It is an upper ontology for modelling web-service composition which offers a process-based perspective. OWL-S provides declarative publications of services properties and capabilities, API for Web services, specifications of prerequisites and consequences of individual services and descriptors for the state of services execution for automatic services discovery, invocation and execution monitoring.

OWL-S process model describes formation of services by composition

(i) Service profile –which presents what function the service computes. This information is expressed in terms of the transformation that the service produces.

(ii) Process model – which describes the service behavior providing a view of the service in terms of process compositions. OWL–S defines three types of processes:

atomic processes, which have associated inputs and outputs and can be directly invoked by the client,

composite processes, which consist of other composite and atomic processes, and

simple processes, which are abstract and simplified view of a composite Process.

(iii)Service grounding – which offers all details about their invocation.

An atomic process cannot be decomposed further and it executes in a single step (similarly to a black box providing a functionality), while a composite process is built up by using a

few control constructs: Sequence, Split, Split-Join, Any-Order, Iterate, If-Then-Else, Choice, repeat-while and repeat-until. Hence, for instance, an if-then-else process is a bag of two processes out of which one is chosen for execution according to the value of a condition, an any-order process is a bag of processes to be executed in some unspecified order but not concurrently, and a repeat-until process is a process to be executed at least one, until a condition becomes true.

### 2.2 Petri nets concept

Petri nets model has a strong capability to model events and states in a distributed system and to capture sequential, concurrency and event-based control. Petri net is also a powerful tool for analyzing and verifying certain properties such as reachability, liveness, and deadlocks (Maurice ter Beek et al., 2007). For examples, the Marking Reachability relation identifies the reachability between any two markings; the construct inclusion relation identifies the inclusions between two OWL-S construct by analyzing their Petri nets representations.

The Petri nets model is a bipartite graph containing places representing states and transitions representing actions. Places hold tokens that represent predicates of the state. A transition will be trigger when all the places pointing to the transition obtain an adequate number of tokens.

A place/transition Petri nets is represented as a quintuple PN = (P, T, F, W, $M_0$), where:

$P=\{p_1,p_2,\ldots,p_m\}$ is a finite set of places.

$T=\{t_1,t_2,\ldots,t_m\}$ is a finite set of transitions that represents the set for tasks, processes, activities, or events.

F is a set of arcs used to represents flow paths, where $F\subseteq(P\times T)\cup(T\times P)$

$W : F\rightarrow (\mathbb{N}\setminus\{0\})$ is the arc weight mapping. $\mathbb{N}$ is natural numbers set.

$M_0$: $P\rightarrow\mathbb{N}$ is an initial marking, where every place has a number of tokens.

Given a marking M, a transition t is *enabled* in M If and only if $M(p)\neq 0$ for each $p\in\bullet t$. t is fired in M If and only if it is enabled in M and M is transformed into M' such that (i)$\forall p\in\bullet t$:$M'(p)=M(p)-1$, (ii) $\forall p\in t\bullet$:$M'(p)=M(p)+1$, and (iii) $\forall p\notin\bullet t\bullet$: $M'(p)=M(p)$. In this case, M' is *directly reachable* from M via t, denoted as M[t>M'. M' is directly reachable from M, denoted as M[>M', If and only if M[t>M' for some $t\in T$. Given $\sigma\in T^*$, M' is *reachable* from M via$\sigma$, denoted as M[$\sigma$>M', If and only if (i) M'=M when$|\sigma|=0$, or (ii) $\sigma=t_1t_2\ldots t_k$, k>0 and there exists a sequence $M_0[t_1>M_1[t_2>\ldots M_{k-1}[t_k>M_k$ such that $M_0=M$ and $M_k=M'$. In this case, $\sigma$is called a firing sequence from M to M'. M' is said to be a reachable marking in $(N,M_0)$, and$\sigma$is called a firing sequence of M. The set of trackable markings in $(N,M_0)$ is denoted as $RM(N,M_0)$. The corresponding reachability graph is denoted as $RG(N,M_0)$.

Given a Petri nets PN=$(N,M_0)$, PN is *bounded* If and only if $RG(N,M_0)$ is finite, i.e., $\exists k\geq 0$ such that $\forall M\in RG(N,M_0)$ $\forall p\in P$:$M(p)\leq K$. In this case, we also say PN is K-bounded. PN is safe If and only if it is 1-bounded. PN is *live* (or $M_0$ is a live marking) If and only if $\forall M\in RG(N,M_0)$ $\forall t\in T$, $\exists M'\in RM(N,M_0)$:M[>*M' and t is enabled in M'. A reachable marking M is a deadlock marking If and only if there are no transitions are enabled in M.

To represent an OWL–S process model with Petri nets, we consider atomic processes as transitions and an atomic process can be executed only if the following two conditions occur:

(i) all of its inputs are available, and

(ii) all processes to be executed before it have been completed.

## 3. P/T Petri nets-based models for Semantic Web services composition

### 3.1 From OWL-S to Petri nets

The Web Service application logic described by OWL-S is first transformed into a Petri nets model to provide a formal representation of the structure and behavior of the service. Petri nets ontologies are defined to carry the operational semantic as well as the inputs, outputs, precondition and effects (IOPE) semantics of OWL-S service functionality.

The processes organized by OWL-S control constructs are mapped to the Petri nets by analyzing their execution semantics, the Perform actions, and the IOPE of each Perform. Its inputs are mapped to the places holding tokens pointing to the transition. Its preconditions are mapped to corresponding arc labels that must be valid in order to enable the transition. The Perform effects and output are mapped to output arcs and places of the transition that will be triggered after the occurrence of the Perform transition.

OWL-S distinguishes between atomic and composite processes. Atomic processes are indivisible processes that result in a message exchange between the client and the server. Composite processes are used to describe the control flow relation between processes.

The more general the model, the less amenable it is to analysis, so we extend classic Petri nets to be Web Service Petri nets for Web services verifications. The followings are definitions:

**Definition 1** The Petri nets model for an atomic process described by OWL-s is a Petri nets $BN = (S, T; F, W, M_0)$ as shown in Fig. 1, where

$S=\{s\}$. It represents the service to be run when $s$ includes tokens.

$T=\{t_b,t_e\}$, where, tb represents beginning of service and $t_e$ represents accomplishment of service, corresponding to Precondition and Effects respectively,

$F=\{( t_b,s), ( s, t_e)\}$,

$M_0=0$,

$W(t_b,s)$ and $W(s, t_e)$ represent Input and Output respectively,

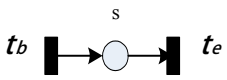



Fig. 1. Atomic process model

Definition 2 The Petri nets model for sequence composition of Web services is a hierarchical Petri nets $CBN= (S, BN; F, W, M_0)$ as shown in Fig. 2, where

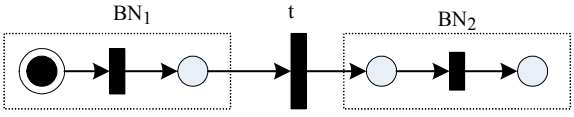



Fig. 2. Sequence model

$S=\{s_1,s_2,\ldots,s_k\}$,

$BN=\{BN_1,BN_2,\ldots,BN_k\}$,where $BN_i$ is a subnet of the i-th service.

$F=\{(BN_i,t_i),(t_i,BN_{i,i+1})\}$,

$M_0=\{M_0(s_i)\mid i=1,2,\ldots,k-1\}=0$,

$W=\{W(BN_i, s_i),W(s_i, ,BN_{i+1})\mid i=1,2,\ldots,k-1\}$,

**Definition 3** The Petri nets model for split composition of Web services is a hierarchical Petri nets BN= (S, BN; F, W, $M_0$) as shown in Fig. 3, where

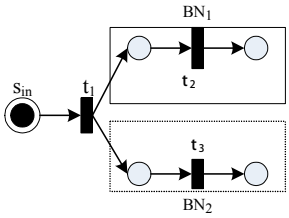



Fig.3. Split model

$S=\{s_{in}\}$,

$T=\{t_1\}$

$BN=\{BN_1,BN_2,\ldots,BN_k\}$,where BNi is a subnet of the i-th service.

$F=(s_{in},t_1)\cup(t_1,BN_i)$,

$W= W( s_{in}, t_1)\cup W(t_1, BN_i)$, i=1,2,…,k,

$M[t_1>M_1\rightarrow M_1[t_2>$,and $M[t_1>M_2\rightarrow M_2[t_3>$.

**Definition 4** The Petri nets model for Split+Join composition of Web services is a hierarchical Petri nets CBN= (S, BN; F, W, $M_0$) as shown in Fig. 4, where,

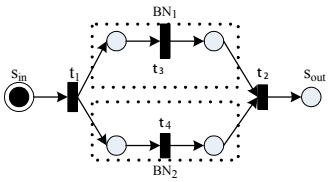



Fig. 4. Split-join model

$S=\{s_{in},s_{out}\}$,

$T=\{t_1,t_2,t_3,t_4\}$

$BN=\{BN_1,BN_2,\ldots,BN_k\}$,where $BN_i$ is a subnet of the i-th service.

$F=(s_{in},t_1)\cup(t_1,BN_i)\cup(BN_i,t_2)\cup(t_1,s_{out})$,

$W= W( s_{in}, t_1)\cup W(t_1, BN_i)\cup W(BN_i,t_2)\cup W(t_1,s_{out})$, i=1,2,…,k,

$M[t_1>M_1\rightarrow M_1[t_2>$,and $M[t_1>M_2\rightarrow M_2[t_3>$

Then $M_1[t_3>$ and $M_2[t_4>\rightarrow M_3[t_4>$.

**Definition 5** The Petri nets model for Choice composition of Web services is a hierarchical Petri nets CBN= (S, BN; F, W, $M_0$) as shown in Fig. 5, where

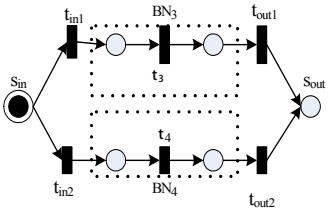



Fig. 5. Choice model

$S=\{s_{in},s_{out}\}$,

$T=\{t_{ini},t_{outi}\}$

$BN=\{BN_1,BN_2,\ldots,BN_k\}$,where $BN_i$ is a subnet of the i-th service.

$F=(s_{in},t_{ini}) \cup (t_{ini},BN_i) \cup (BN_i ,t_{outi}) \cup (t_{outi} ,s_{out})$,

$W= W(s_{in},t_{ini}) \cup W(t_{ini},BN_i) \cup W(BN_i ,t_{outi}) \cup W(t_{outi} ,s_{out})$, i=1,2,…,k,

$M[t_{in1}>M_1 \rightarrow \neg M_1\ [t_2>$ or $M[t_{in2}> M_2 \rightarrow \neg M_2[t_{in1}>$ .

**Definition 6** The Petri nets model for If-Then-Else composition of k Web services is a hierarchical Petri nets BN= (S, BN; F, W, $M_0$) as shown in Fig. 6, where

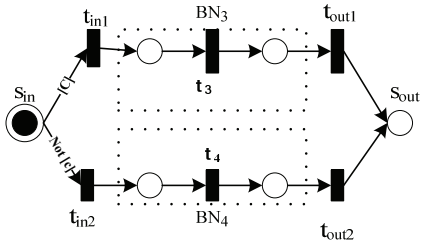



Fig. 6. If-then-else model

$S=\{s_{in},s_{out}\}$,

$T=\{t_{ini},t_{outi}\}$

$CBN=\{BN_1,BN_2,\ldots,BN_k\}$,where $BN_i$ is a subnet of the i-th service.

$F=(s_{in},t_{ini}) \cup (t_{ini},BN_i) \cup (BN_i ,t_{outi}) \cup (t_{outi} ,s_{out})$,

$W= W(s_{in},t_{ini}) \cup W(t_{ini},BN_i) \cup W(BN_i ,t_{outi}) \cup W(t_{outi} ,s_{out})$, $W(s_{in},t_{in1})=c$, $W(s_{in},t_{in2})\neq c$, i=1,2,…,k,

$M[t_{in1}> \rightarrow \neg M[t_2>$ or $M[t_{in2}> \rightarrow \neg M[t_{in1}>$

**Definition 7** The Petri nets model for Repeat-While composition of k Web services is a hierarchical Petri nets CBN= (S, BN; F, W, $M_0$) as shown in Fig. 7, where

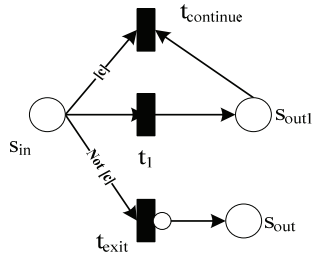



Fig. 7. Repeat-while model

$S=\{s_{in},s_{out1},s_{out}\}$,

$T=\{t_1, t_{continue}, t_{exit}\}$

$CBN=\{BN_1,BN_2,\ldots,BN_k\}$,where $BN_i$ is a subnet of the i-th service.

$F=(s_{in},t_1) \cup (t_1, s_{out1}) \cup (s_{out1}, t_{continue}) \cup (t_{continue}, s_{in}) \cup (s_{in},t_{exit}) \cup (t_{exit}, s_{out})$,

$W= W(s_{in},t_1) \cup W(t_1, s_{out1}) \cup W(s_{out1}, t_{continue}) \cup W(t_{continue}, s_{in}) \cup W(s_{in},t_{exit}) \cup W(t_{exit}, s_{out})$, $W(s_{in},t_1)=c$, $W(s_{in},t_{exit})\neq c$, $i=1,2,\ldots,k$,

$M[t_1> \rightarrow \neg M[t_{exit} >$ or $M[t_{exit} > \rightarrow \neg M[t_1>$

$(t_{exit}, s_{out})$ is an inhibitor arc. An inhibitor arc connects a place to a transition and is represented by a line with a small circle instead of an arrowhead at the transition. The inhibitor arc disables the transition when the input place has a token and enables the transition when the input place has no token and other input places have at least one token per arc weight. No tokens are moved through an inhibitor arc when the transition fires.

**Definition 8** The Petri nets model for Repeat-Until composition of k Web services is a hierarchical Petri nets $CBN= (S, BN; F, W, M_0)$ as shown in Fig. 8, where

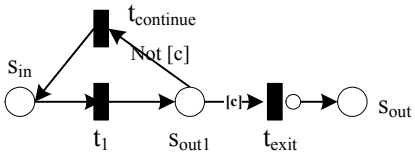



Fig. 8. Repeat-until mode**l**

$S=\{s_{in},s_{out1},s_{out}\}$,

$T=\{t_1, t_{continue}, t_{exit}\}$

$BN=\{BN_1,BN_2,\ldots,BN_k\}$,where $BN_i$ is a subnet of the i-th service.

$F=(s_{in},t_1) \cup (t_1, s_{out1}) \cup (s_{out1}, t_{continue}) \cup (t_{continue}, s_{in}) \cup (s_{out1},t_{exit}) \cup (t_{exit}, s_{out})$,

$W= W(s_{in},t_1) \cup W(t_1, s_{out1}) \cup W(s_{out1}, t_{continue}) \cup W(t_{continue}, s_{in}) \cup W(s_{out1},t_{exit}) \cup W(t_{exit}, s_{out})$, $W(s_{out1},t_{exit})=c$, $W(s_{out1}, t_{continue})\neq c$, $i=1,2,\ldots,k$,

$(t_{exit}, s_{out})$ is an inhibitor arc.

The Petri nets model captures the structure and operational semantics of composite WS described by OWL-S process model. OWL-S Processes can be mapped to Petri nets and generalized as follows:

Process ::= AtomicProcess … | CompositeProcess CProcess …

CProcess ::= AnyOrder PerformanceList |

Sequence PerformanceList |
Split PerformanceList |
SplitJoin PerformanceList |
Choice PerformanceList |
IfThenElse Performance|
RepeatWhile Peformance |
RepeatUntil Performance | Connect …

Performance ::= Perform Process.

### 3.2 Composition Net

Most systems that arise from practical applications are very complex and practically unmanageable.(Zhijun Ding et al.,2005) For this reason, modular construction methods provide a mechanism to manage the complexities of a large system that can be built out of well understood smaller subsystems. One way to do this is through Petri nets synthesis based on some prescribed construction rules which preserve certain logical properties as the construction progresses. These subsystems are then combined through common places and/or transitions into a larger subsystem at each synthesis step. Each subsystem is modeled separately while ignoring interactions with other subsystems. These subsystems are then combined through common places and/or transitions into a larger subsystem at each synthesis step.

Every interface place represents either messages from the service to a partner, or messages from a partner to the interface. The service is connected to the interface place in only one direction. The interface net identifies each subsystem as a unique functional object, and the interconnection net specifies the relationships between subsystems. As a result, we can visualize the entire topological view of a system by interconnecting each of the interconnection nets according to our unique module-interconnection technique. Furthermore, we assume that a service reads or writes only one message per transition. It may, however, perform transitions that do not interact with the interface at all. The concept of module formalizes our view on Web services as workflow modules equipped with an interface.

**Definition 9** (Module) MD=(P, T, F,W,$m_0$) is a module if

(i) P=(Place, Port), Place=($P_s$,$P_e$,$P_M$), $P_s$ is the start and $P_e$ is the end place respectively. $P_M$ is the set of internal places. Port=($Port_I$, $Port_O$), $Port_I$ is the set of input ports, and $Port_O$ is the set of output ports, which are pairwise disjoint.

(ii) (P, T, F,W, $m_0$) is a Petri nets with $m_0(p_s)=1$ and $m_0(p)=0$ for all other places p,

(iii) for all places and transitions x, ($p_s$,x) and (x, $p_e$) are in the reflexive and transitive closure of F,

(iv) every write transition is connected to exactly one output port and no input port, every read transition is connected to exactly one input port and no output port, every internal transition is connected to neither an input nor an output port.

We model the system behavior w.r.t abstraction BN by a Petri nets N=(P,T) as follows. Let J to be a sub-system, and sub-system J ∈ M is modeled as a subnet $BN_i$=($BP_i$,$BT_i$) of N, called the blackbox Petri nets of J. Suppose J has m inputs and n outputs, the corresponding $BN_i$ consist of five parts.

Given an abstraction subsystem BN, let BNi be the corresponding Petri nets. We conduct reachability analysis for BN based on some initial marking $M_0$. Denote $RG(N,M_0)$ as the resulting reachability graph. We check that the following conditions hold for $RG(N,M_0)$:

(1) $RG(N,M_0)$ is finite.

(2) $M_0(p_M)=0$ for each $BN_i$ in N.

(3) For each reachable marking M, for each blackbox Petri nets $BN_i$ in BN with m inputs and n outputs, the following two conditions hold: (1)$\forall i \in [1\ldots m]:M(p_i)<1$ (2) If $\exists i \in [1\ldots m]:M(p_i)=1$, then $\forall j \in [1..m], j \neq i : M(p_j)$ $\forall j \in [1\ldots m]$, $j \neq i:M(p_j)$

Web service net provides an interconnection mechanism across different levels of component diagrams. Interconnections can be visualized by: (1) interoperation nets of sender and receiver components, and (2) the interface net of the sender, receiver, and channel components. We believe that this is a very important feature to visualize very large systems. By applying such visual abstractions, such as replacing large interoperation nets with simpler interconnection nets or even with interface nets, complicated nets can be effectively visualized at various levels of abstraction.

**Theorem 1** Let $PN=(P,T;F,W,M_0)$ be a Petri nets model of Web service composition. $R(M_0)$ is the reachability set of $M_0$, then $R(M_0)$ is a finite set.

Proof: According to the definition of P/T Petri nets, for each reachability marking $M:M \in R(M_0)$, M(p) is a set of tokens residing in a place $p \in P$, including a tuple of symbolic individuals or structure terms constructed from individuals and operations. Moreover, for a service composition, the number of individuals and variables is always finite, so the number of tokens in M(p) is finite. At the same time, P is a finite set of places, and obviously the combination of a finite number of tokens with a finite number of places is always finite. Hence the number of reachable marking is finite, that is, $R(M_0)$ is a finite set.

**Theorem 2** Let M be a marking in $RG(N,MB_{0B})$, $MB_{1B}[\sigma>MB_{2,B}$, p is an internal place of $JB_{.B}$, For each $J \in BN$, M is reachable in $RG(N,MB_{0B})$ via a canonical execution sequence.

Proof: Suppose $MB_{2\,B}$ is reachable from $MB_{1B}$ via σ in $RG(N,MB_{0B})$, where $MB_{1B}(p)=0$. Let $k=|\sigma P^{outP}|$. Then $k \leq |\sigma P^{inP}| \leq k+1$. In addition, $MB_{2B}$ is reachable from $MB_{1B}$ in $RG(N,MB_{0B})$ via $\eta=\eta B_{0B}\eta B_{1B}\ldots\eta B_{kB}\eta B_{k+1B}$. When $MB_{1B}=MB_{0B}$, it is a canonical execution sequence for reachable marking $MB_{2B}$ w.r.t BN. Since $MB_{1B}(p)=0$, by the structure of BN, there must be at least k input transitions of BN in σ, and for each $l \in [1\ldots k]$, the *l*-th input transition of BN must occur before the *l*-th output transition of BN in σ, which can be written as $\eta B_{0B}\sigma B_{1B}\ldots\sigma B_{kB}\eta B_{k+1B}$. So an execution sequence σ from $MB_{1B}$ to $MB_{2B}$ is canonical execution sequence. Since $MB_{0B}(p)=0$, any execution sequence for a reachable marking M can be rewritten into its canonical form w.r.t BN.

Given two abstractions BN and BN′, BN′ is called a one-step refinement of BN, denoted as $BN \prec BN'$, If and only if $BN'=(BN\backslash\{J\}) \cup \{J_1,J_2,\ldots,J_k\}, k\geq 2$, where $\{J_1,J_2,\ldots,J_k\}$ is the set of component subsystems of J via one step decomposition.

**Theorem 3** Given Petri nets $N \prec N'$. Let $RG(N,M_0)$ and $RG(N',M_0')$ be the corresponding reachability graphs of N and N', respectively. The following statements are true:

- Deadlock: $RG(N,M_0)$is deadlock free If and only if $RG(N',M_0')$ is deadlock free.
- Liveness: A transition $t \in T$ is live in $RG(N,M_0)$ If and only if it is live in $RG(N',M_0')$.

Proof: Deadlock: Suppose M is a deadlock marking in RG(N, $MB_{0B}$). Let σ be a firing sequence for M. Then no transition in T is enabled in M. In particular, M(p)=0, and $|\sigma P^{inP}|=|\sigma P^{outP}|$. There is a marking M' in RG(N',M'$B_{0B}$) reachable via σ' such that M'(P\{P})=M(P\{p}) and σ' =σ. Thus no transition from T\$TB_{outB}$ is enabled in M'. Moreover, $|\sigma' P^{inP}|=|\sigma' P^{outP}|$. Thus no transition from $TB_{outB}$ is enabled in M' either. Hence, M' is a deadlock marking in RG(N',M'$B_{0B}$). On the other hand, suppose M' is a deadlock marking in RG(N',M'$B_{0B}$). Let M be a marking of N such that M(P\{p})=M'(P\{p}) and M(p)=0. By similar argument, we can also show M ∈ RG(N, $MB_{0B}$).

Liveness: Suppose a transition t ∈ T is enabled in M ∈ RG(N, MB0B). Let M[t>MB1B in RG(N, $MB_{0B}$) and σ be a firing sequence for M. Then σt is a firing sequence for $MB_{1B}$, and there is a marking M'$B_{1B}$ ∈ RG(N',M'$B_{0B}$) reachable via σ' such that σ'=σ. As a result, t is also enabled in some marking M' in RG(N',M'$B_{0B}$) in the path σ' from M'$B_{0B}$ to M'$B_{1B}$. On the other hand, suppose t ∈ T is enabled in M' ∈ RG(N', M'$B_{0B}$). By similar argument, we can also show that t is enabled in some M ∈ RG(N, $MB_{0B}$). As a result, a transition t ∈ T is enabled in RG(N,$MB_{0B}$) if and only if it is enabled in RG(N', M'$B_{0B}$).

We construct the Petri nets model for application composed of semantic Web services using following algorithm.

**Algorithm 1** Map OWL-S process model to the Petri nets model for application composed of semantic Web services.

(1) For every Web service ontology, whose behavior is represented by OWL-S, the structure and operational semantics of composite WS process are captured; the structure and operational semantics of composite WS are described by OWL-S process models.

(2) Utilize resolution principle to resolve the existential quantifier and universal quantifier by using skolem function and kripke structure. Then resolve the inference rule of OWL-S process model.

(3) According to dependent relation of all Web services, OWL-S process are mapped to Petri nets, and Petri nets model for WS is constructed as a tree using definition 1.

(4) Then the Petri nets models for composition of all Web services are constructed by using definition 2-8. The nodes of the tree correspond to composite processes that represent different control constructs such as Choice for non-deterministic choices, Sequence for deterministic sequences of processes, and If conditionals. Atomic processes are represented as the leaves of the tree.

(5) For those services without preceding service, the transitions representing their beginning are combined to one transition, named as tb. Then, introduce a place $s_0$, such that $\bullet t_b=\{s_0\}$, $t_b\bullet=\{s_i$, where service i has no preceding service $\}$, $\bullet s_0=\varphi$, $s_0\bullet=\{t_b\}$, $W(s_0,t_b)=1$, $M_0(s_0)=1$.

(6) For those services without succeeding service, the transitions representing their accomplishment are combined to one transition, named as $t_e$. Then, introduce a place $s_e$, such that $\bullet t_e=\{s_i$, where service i has no succeeding service $\}$, $t_e\bullet=\{s_e\}$, $\bullet s_e=\{t_e\}$, $s_e\bullet=\varphi$, $W(t_e, s_e)=1$, $M_0(s_e)=0$.

In order to use Petri nets as a process model for control purposes, the analysis of their corresponding reachability graphs has turned out to be a suitable analysis technique. Algorithm 2 can generate the reachability and coverability graph.

**Algorithm 2** Reachability and coverability graphs Generation.

Reachability-Graph(<P, T, F,W,$M_0$>)

```
1 <V, E, v0> := <{M0},Φ,M0>;
2 Work : set := {M0};
3 while Work ≠ Φ;
4 do select M from Work;
5    Work := Work \ {M};
6    for t ∈ enabled(M)
7    do M':= fire(M, t);
8        if M'∉V
9            then V:= V ∪ {M'}
10       Work := Work ∪ {M'};
11       E := E ∪{<M, t,M0>};
12 return <V, E, v0>;
```

The algorithm makes use of two functions: The set Work may be implemented as a stack, in which case the graph will be constructed in a depth-first manner, or as a queue for breadth-first. Breadth first search will find the shortest transition path from the initial marking to a given (erroneous) marking. Some applications require depth first search. (Huaikou Miao et al.,2008).

## 4. Colored Petri nets-based models for Web services composition

In this section, a colored Petri nets (CPN) (Jensen K.1994) based algebra for modeling Web services is proposed. The model is expressive enough to capture semantics of complex service combinations and their respective specificities.

The Web service is formally defined and the obtained framework enables declarative composition of Web services (Zhaoli Zhang, et al.2008). Within the model, availability, confidentiality and integrity of the composite service can be analyzed.

### 4.1 Web services as colored Petri nets

By a Web service we mean a software component that is described via WSDL and is accessible via standard network protocols such as but not limited to SOAP over HTTP. Web services should be based on open standards, platform independent, application independent, and enable to share data and resources. Web service composition is a task of combining and linking existing Web services to create new web processes in order to add value to the collection of services.

For the sake of fast computation, many researchers prefer Petri nets(Thomas J P,et al.2005),since they are well suited for capturing flows in Web services, modeling the distributed nature of Web services, representing methods in a Web service and reasoning about the correctness of the flows.

A Web service behavior is basically a partially ordered set of operations. Therefore, it is straightforward to map it into a Petri nets. Operations are modeled by transitions and the state of the service is modeled by places.

The arrows between places and transitions are used to specify causal relations. Web services can be categorized into material services (e.g., delivery of physical products), information services (create, process, manage, and provide information), and material information

services, the mixture of both. Therefore, information is modeled by tokens and the types of information are modeled by the colors of the tokens.

It is assumed that a Petri nets, which represents the behavior of a service, contains one input place (i. e., a place with no incoming arcs and one output place (i. e., a place with no outgoing arcs.A Petri nets with one input place for absorbing information, and one output place for emitting information, will facilitate the definition of the composition operators and the analysis as well as the verification of certain properties (e.g., reachability, availability, and security. At any given time, a Web service can be in one of the following states: NotInstantiated, Ready, Running, Suspended, or Completed(Schuster H,et al.2000). When a Web service is in the Ready state, it means that tokens in their corresponding input place enable postset (set of transitions) of input place to fire.

Whereas the Completed state means that preset (set of transitions) of output place has fired and has generated tokens in corresponding output place.

**Definition 10** (Service net) SN=(P, T, F, C, $I_-$, $I_+$,$M_0$, i ,o, l) is called a service net if and only if: (i) $\Sigma$=(P, T, F, C, $I_-$, $I_+$, $M_0$) is a colored Petri nets: (ii) i∈P is the input place with $^\bullet x=\varnothing$ that is $\neg\exists o\in T$: (t,i) ∈F; (iii) o∈P is the output place with $x^\bullet=\varnothing$ that is $\neg\exists o\in T$: (t,i) ∈F; (iv) l:T→A∪τ is a labeling function where A is a set of operation names. It is assumed that $\tau \notin A$ denotes a silent operation.

Silent operations are transition firings that cannot be observed. They are used to distinguish between external and internal behavior of the services. The sample service net is shown in Fig.9. The incoming arcs of a transition are marked with the number and color of tokens to enable the transition, and the outgoing arcs are marked with number and color of tokens that the transition generates. The color of token represents the type of information but not the content of information. Thus, two tokens with same color may be different. Obviously, transition t1 is enabled, and this service net is in Ready state. Subsequently, when this service net reaches Completed state, the output place o will contain four tokens: two colored "a", one colored "e", and one colored "f".

Now we give a formal definition of a Web service.

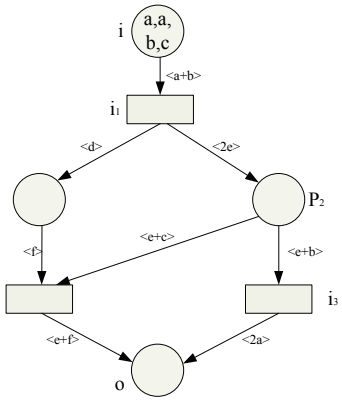



Fig. 9. Sample service net

**Definition 11** (Web service) A Web service is a tuple S= (ID, Desc, Loc, URL, CS, SN), where ID is the unique identifier of the service which can be a name or a global unique number; Desc is the description of the service provided which summarizes what the service offer: Loc is the server in which the service is located:
URL is the invocation of the Web service: CS is a set of its component services. If CS=S.ID then S is a basic service. Otherwise S is a composite service; SN=(P, T, F, C, $I_-$, $I_+$, $M_0$, i, o, l) is the service net modeling the dynamic behavior of the service.

### 4.2 Composing Web services

Researchers have discussed various composite constructs (Narayanan S, Ren Z H, et al.,2003). In this chapter, we take sequence, concurrent, choice and loop constructs as basic constructs specified in the control flow, and take replace as advanes constrmct.We also give a formal semantics to the proposed algebra in terms of Petri nets.

#### 4.2.1 Composite constructs

Below we describe syntax and informal semantics of the service algebra operators. The constructs are chosen to allow basic and advanced Web service composition.
The services can be defined as:
**Definition 12** S::= $\epsilon$ | X | Seq(S,S) | Conc(S,S) | Choice(S,S) | Loop(S) | Rep(S,a,S), where $\epsilon$ represents an empty service, i.e., a service performs no operation.
*X* represents a service constant, used as an atomic or basic service in this context.
Seq($S_1$,$S_2$) represents a composite service that performs the service $S_1$ followed by the service $S_2$. Seq(.) is an operator of sequence. If a composite service that performs either the service $S_1$ followed by the service $S_2$, or $S_2$ followed by $S_1$, it is called unordered sequence. In practice, we can decide the order by any condition since the order is not important, then unordered sequence could be treated as sequence.
Conc($S_1$,$S_2$) represents a composite service that performs the services $S_1$ and $S_2$ independently. Both services are concurrently enabled and the overall composite service waits until both services are completed. Conc(.) is a concurrent operator.
Choice($S_1$,$S_2$) represents a composite service that behaves as either service $S_1$ or service $S_2$. Once one of them is executed, the other is discarded. Choice(.) is a choice operator. The choice is not arbitrary (in fact, there is no absolute arbitrariness), and depends on conditions. Thus, the condition construct with Boolean variants could be treated as choice.
Loop(S) represents a composite service that performs a certain number of times of the service S. Loop(.) is a loop operator.
Rep($S_1$, a, $S_2$) represents a composite service that behaves as $S_1$ except for operation in $S_1$ with label a that is replaced by the nonempty service $S_2$. Rep( · ) is a replace operator.
The proposed algebra verifies the closure property. It guarantees that each result of an operation on services is a service to which one can again apply algebra operators. Software engineers thus are able to build more complex services by aggregating and reusing exfisting services through service algebra.

#### 4.2.2 Formal semantics

Let $S_i$=($ID_i$, $Desc_i$, $Loc_i$, $URL_i$, $CS_i$, $SN_i$) with $SN_i$=($P_i$, $T_i$, $F_i$, $C_i$, $I_{i}$, $I_{+I}$, $M_{0i}$, $i_i$, $o_i$, $l_i$) for i=1,…,n, be n Web services such that $P_i \cap P_j = \varnothing$ and $T_i \cap T_j = \varnothing$ for $i \neq j$.

It is important to note that service composition, as will be described below, applies to syntactically different services. This is due to the fact that the places and transitions of the component services must be disjoint for proper composition (Zhaoli Zhang, et al.2008). However, a service may be composed with itself. In this case, the overlapping must be resolved prior to composition. This can be accomplished by renaming the sets P and T of one of the equal services. The two services remain equal up to isomorphism on the names of transitions and places. Note also that, in case of silent operations, we represent graphically the corresponding transitions as black rectangles.

**Empty service** The empty service $\epsilon$ is a service that performs no operation. It is used for technical and theoretical reasons.

**Definition 13** The empty service is defined as $\epsilon$ =(ID, Desc, Loc, URL, CS, SN), where ID= Empty; Desc=Empty web service; Loc=Null, which means that there is no server for the service; URL=Null, which means that there is no URL for the service; CS={Empty} and SN=({p},∅,∅,C,0,0,{0}, p, p,∅).

Graphically, $\epsilon$ is represented by CPN of Fig.10(a) containing only one place.

Except for the empty service, in definitions below, ID is the ID of the new service; Desc is the description of the new service; Loc is the location of the new service; URL is the invocation of the new service.

**Sequence** The sequence operator allows execution of two services $S_1$ and $S_2$ in sequence. $S_1$ must be completed before $S_2$ can start. This is typically the case when a service depends on the output of the previous service.

**Definition 14** The service $Seq(S_1,S_2)$ is defined as $Seq(S_1,S_2)$=(ID, Desc, Loc, URL, CS, SN) where $CS=CS_1\cup CS_2$, SN=(P, T, F, C, I-, $I_+$, $M_0$, i, o, l) where $P = P_1 \cup P_2$, $T = T_1 \cup T_2 \cup \{t\}$, $F = F_1 \cup F_2 \cup \{(o_1, t), (t, i_2)\}$, $i=i_1$, $o=o_2$, $I_- = I_-\cup I_-(o_1,t)$, $I_+=I_+\cup I_+(i_2,t)$, $M_0=M_{01}\cup M_{02}$, and $l=l_1\cup l_2\cup\{(t,\tau)\}$.

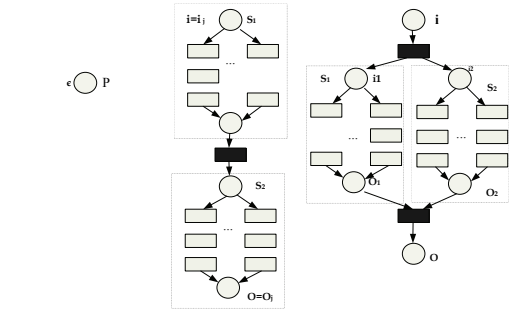
Given $S_1$ and $S_2$, $Seq(S_1,S_2)$ is represented graphically by CPN shown in Fig.10(b). As mentioned above the second service depends on the output of the first service thus $C(i_2)\subseteq C(o_1)$ must be satisfied, or else, the second service cannot be enabled and the new composite service cannot work.

**Concurrent** The concurrent operator permits concurrent execution of two services $S_1$ and $S_2$. This is typically the case when some small (atomic) services without interfering with each other are merged into a bigger composite service.

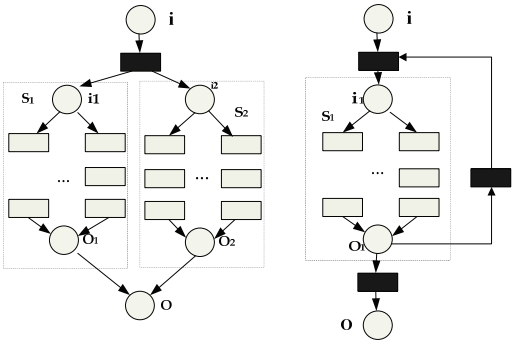
**Definition 15** The service $Conc(S_1,S_2)$ is defined as $Conc(S_1,S_2)$=(ID, Desc, Loc, URL, CS, SN) where $CS=CS_1\cup CS_2$, SN={P, T, F, C,I_,$I_+$,$M_0$,i, o, l) where $P = P_1 \cup P_2 \cup \{i, o\}$, $T = T_1 \cup T_2 \cup (t_i, t_o)$, $F= F_1 \cup F_2 \cup \{(i,t_i), (t_i,i_1), (t_i,i_2), (o_1,t_o), (o_2,t_o), (t_o,o)\}$, $I_-=I_-\cup I_-(i,t_i)\cup I_-(o_1,t_o)\cup I_-(o_2,t_o)$, $I_+=I_+\cup I_+(i_1, t_i) \cup I_+(i_2,t_i) \cup I_+(o.t_o)$, $M_0=M_{01}\cup M_{02}$, and $l=l_1\cup l_2\cup \{(t_i, \tau), (t_o, \tau)\}$.

Given $S_1$ and $S_2$, $Conc(S_1,S_2)$is represented graphically by CPN as shown in Fig10(c).

**Choice** The choice operator performs either service $S_1$ or service $S_2$. Once one of them executes, another service is discarded.




(a) Empty (b) Sequence (c) Concurrent




(d) Choice (e) Loop

Fig. 10. Colored Petri nets of basic constructs

**Definition 16** The service Choice($S_1$,$S_2$) is defined as Choice($S_1$,$S_2$)=(ID, Desc, Loc, URL, CS, SN) where CS =$CS_1 \cup CS_2$, SN=(P, T, F, C, I_, $I_+$,$M_0$,i, o, l) where P=$P_1 \cup P_2 \cup$ { i, o}, T = $T_1 \cup T_2 \cup$ {$t_{i1}$, $t_{i2}$, $t_{O1}$,$T_{O2}$}, F=$F_1 \cup F_2 \cup$ { ($i,t_{i1}$), ($i, t_{i2}$), ($t_{i1},i_1$), ($t_{i2},i_2$), ($o_1,t_{o1}$),($o_2, t_{o2}$), ($t_{o1}$,o), ($t_{o2}$,o)}, I_= I_ $\cup$ I_(i, $t_{i1}$) $\cup$ I_(I,$t_{i2}$) $\cup$ I_($o_1,t_{o1}$) $\cup$ I_($o_2,t_{o2}$), $I_+=I_+ \cup I_+(i_1,t_{i1}) \cup I_+(i_2,t_{i2}) \cup I_+(o,t_{o1}) \cup I+(o,t_{o2})$, $M_0=M_{o1} \cup M_{02}$, and $l=l_1 \cup l_2 \cup \{(t_{i1},\tau), (t_{i2},\tau), (t_{o1},\tau), (t_{o2},\tau)\}$.

Given $S_1$ and $S_2$, Choice($S_1$,$S_2$) is represented graphically by CPN as shown in Fig.10(d).

**Loop** The loop operator allows that the service S performs a certain number of times. Typical examples where loop is required are communication and quality control where services are execrated more than once.

**Definition 17** The service Loop ($S_1$) is defined as Loop($S_1$)=(ID, Desc, Loc, URL, CS, SN), where CS=$CS_1$,SN=(P,T,F, C, I_, $I_+$, $M_0$, i, o, l) where P = $P_1 \cup \{i, o\}$, T=$T_1 \cup \{t_i,t_o,t\}$, F=$F_1 \cup$ {($i,t_i$), ($t_i,i_1$), ($o_1,t_o$), ($t_o$,o), ($o_1$,t), (t,$i_1$)}, I_=I_$\cup$ I_(i, $t_i$) $\cup$I_($o_1,t_o$) $\cup$I_($o_1$,t), $I_+=I_+ \cup I_+(i_1,t_i) \cup I_+(o,t_o) \cup I_+(i_1,t)$, $M_0=M_{01}$, and $l=l_1 \cup \{(l_i,\tau),(t_o,\tau),(l,\tau)\}$.

Given $S_1$, Loop($S_1$) is represented graphically by CPN as shown in Fig.10(e).

**Replace** The replace construct, in which operations are replaced by more detailed nonempty services, is used to introduce additional component services into a service.

Replace is the transformation of a design from a high level abstract form to a lower level more concrete form hence allowing hierarchical modeling.

**Definition 18** Let a∈ A. The service Rep($S_1$,a, $S_2$) is defined as Rep($S_1$,a,$S_2$)=(ID, Desc, Loc, URL, CS, SN).

If a ∈ $I_1(T_1)$ then CS = $CS_1 \cup CS_2$, otherwise CS = $CS_1$.

SN=(P, T, F, C, $I_-$, $I_+$, $M_0$, i, o, l) where $P=P_1 \cup (P_2 - \{i_2, o_2\})$, $T=T_1 \cup T_2 - l_1^{-1}(a)$, $F=(F_1 - \{(x,y) \mid x \in l_1^{-1}(a) \text{ or } y \in l_1^{-1}(a)\}) \cup F_2$, $I_-=(I_{-1} - \{I_{-1}(p,t) \mid t \cup I_1^{-1}(a)\}) \cup I_{-2}$, $I_+=(I_+ - \{I_{+1}(p,t) \mid t \cup I_1^{-1}(a)\}) \cup I_{+2}$, $M_0=M_{01} \cup M_{02} - M(i_2) - M(o_2)$, if $t \cup (T_1 - I_1^{-1}(a))$ then $l(t)= l_1(t)$, otherwise $l(t)= l_2(t)$.

Given $S_1$, a and $S_2$, Rep($S_1$,a,$S_2$) is represented graphically by CPN shown in Fig.11. From the definition and Fig.11, we could find that the labeled transition, which to be replaced, should have only one incoming arc and one outgoing arc. If each transition in a Petri nets has only one incoming arc and one outgoing arc, the Petri nets is an ordinary Petri nets. We recommend high-level design is the ordinary Petri nets for allowing replace and hierarchical modeling.

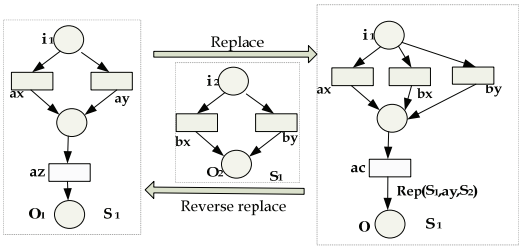



Fig. 11. Colored Petri nets of replace construct

### 4.3 Composing Web services

A composite Web service is a system that consists of several conceptually autonomous but cooperating units. It is difficult to specify how this system should behave and ensure that it behaves as required by the specification. However, the model based CPN could do something.

#### 4.3.1 Closure property

As mentioned above, the proposed algebra verifies the closure property.

**Theorem 4** The services compositions presented in Definition 12 are closed.

**Proof** The closure property of one-step composition is an immediate consequence of Definition 12, and the closure property of multiple-step composition can be proved by mathematical induction.

Not only it guarantees that each result of an operation on services is a service to which one can again apply algebra operators described in 4.2.1, but also it allows hierarchical modeling. For high-level abstract model, via replace, it can be transformed to lower-level model, which also can be transformed to higher-level model via reverse replace (see Fig.11). Behavioral equivalences are useful in verification as they lay the conceptual basis for

deciding that the behavior of two Web services can be considered to be "the same". They can also be used as a tool for reducing verification effort by replacing the CPN of a service by a smaller (in size) but "equivalent" one. Hence, analysts can analyze and verify the model in different levels.

### 4.3.2 Availability

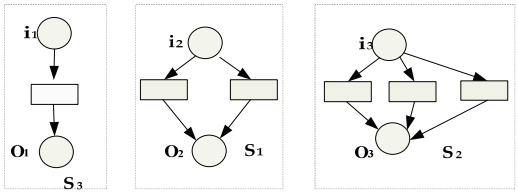
In a CPN model, the data (information) type is distinguishable, thus the designer can figure out which services can be composed together and which cannot. The data types are defined as classes, including ordinary types (such as integer, real, Boolean, etc.), same as in object-oriented programming. In a CPN model, colors of the tokens represent the classes (data types).

Since subclass inherits from superclass, color should be redefined to reflect inheritance. For instance, there are two superclasses: $sc_1$ and $sc_2$, and there are two subclasses: $bc_1$ inherits from $sc_1$, and $bc_2$ inherits from $sc_2$.

If $sc_l$ is colored "a" and $sc_2$ is colored "b", then $bc_l$ and $bc_2$ should be colored "a-c" and "b-d" to reflect the inheritance relation. Consequently, when a service input type requires color "a", both $sc_l$ and $bc_l$ are accept able. However, when a service input type requires color "a-c", only $bc_l$ is acceptable and $sc_l$ will be denied.

Consider three services as shown in Fig.12. We note that service $S_1$ cannot be composed with itself by loop construct because its output place cannot provide token colored "a", which is needed by its own input place $I_1$. For the same reason, service $S_2$ cannot be composed with itself by loop construct, but service $S_3$ can. We can also find that service $S_1$ can be composed with service $S_2$ by sequence construct only by order that $S_2$ is followed by $S_1$,and service $S_1$ can be composed with service $S_3$ by sequence construct only when $S_1$ is followed by $S_3$.

However, service $S_2$ can be composed with service $S_3$ by sequence construct by any order.




$C(i_1)=\{a\}, C(i_2)=\{b\text{-}f,c\}, C(i_3)=\{b,c\}$

$C(o_1)=\{b\text{-}d,c\}, C(o_2)=\{b\text{-}e,a,c\}, C(o_3)=\{b\text{-}f,c\}$

Fig. 12. Availability of composition

From the above analysis, we can reach the following conclusions:

If composite service $S = Seq(S_1,S_2)$ is available, $C(i_2) \subseteq C(o_1)$ must be satisfied.

If composite service $S = Conc(S_1,S_2)$ is available, the following must be satisfied: $C(i_1) \cup C(i_2) \subseteq C(i)$ and $C(o) = C(o_1) \cup C(o_2)$.

If composite service $S = Choice(S_1,S_2)$ is available, the following must be satisfied: $C(i_1) \cup C(i_2) \subseteq C(i)$ and $C(o) = C(o_1) \cup C(o_2)$.

If composite service $S = Loop(S_1)$ is available, $C(i_1) \subseteq C(o_1)$ must be satisfied.

In an elementary Petri nets, the data type is not distinguishable; hence the elementary Petri nets-based model for web composition cannot verify the availability feature of the composite service (Hamadi R,2003).

### 4.3.3 Security

Security is an important issue in information system. Confidentiality policies emphasize the protection of confidentiality (Zhaoli Zhang, et al.2008). Multilevel security (MILS) has a long tradition in military environments and is an important requirement in the trusted computer system evaluation criteria (TCSEC) for the security classes.

Subjects and objects of a system are assigned security classes (e.g. "high" and "low")with a specific order (high>low). A well known MILS model is the BellLaPadula model[ll]. The two most prominent rules are No-read-up and No-write-down which state that a low-level subject is not allowed to read high-level objects, and high-level objects can only be written by low-level subjects. These two rules result in an information flow from "low" to "high".

If the component services, transitions and places are assigned security level, the security feature of the Petri nets-based model for Web service composition can be verified by coverability graph(Knorr K.,2001).

The simplest type of confidentiality classification is a set of security clearances arranged in a linear (total) ordering. Let L(S) be the security clearance of service S. Expand the security clearances by adding a set of categories to each security classification. Each category describes a kind of information in all of those categories.

These sets of categories form a lattice under the operanon $\subseteq$ (subset of). Let G(S) be the category set of service S. Each security clearance and category forms a security level. Define the relation dom (dominates) as follows. (Bell D &Lapadula L.1996)

**Definition 19** The security level (L,G) dom-inates the security level (L', G') if and only if $L' \leq L$ and $G' \subseteq G$.

**Definition 20** (Simple security condition) S can read O if and only if S dom O, and S has discretionary read access to O.

To preserve confidentiality, following properties must meet while composing Web services。

**Property 1**. For composite service S, if S = Seq($S_1$, $S_2$), $S_2$ dom $S_1$.

**Proof** If service $S_1$and $S_2$ can be composed with the sequence construct by the order that S1 is followed by $S_2$,then $S_2$ can read $S_1$. For satisfying Definition 20, this can happen if and only if $S_2$ dom $S_1$.

**Property 2**. For composite service S, if S = Conc($S_1$,$S_2$), $S_1$ dom S, and $S_2$ dom S.

**Proof** If service $S_1$ and $S_2$ can be composed with the concurrent construct to form composite service S, then $S_1$ and $S_2$ can read S. For satisfying Definition 20, this can happen if and only if $S_1$ dom S, and $S_2$ dom S.

**Property 3**. For composite service S, if S=Choice($S_1$,$S_2$),$S_1$ dom S, and $S_2$ dom S.

**Proof** If service S and $S_2$ can be composed with the choice construct to form composite service S, then $S_1$ and $S_2$ can read S. For satisfying Definition 20, this can happen if and only if $S_1$ dom S and $S_2$ dom S.

Integrity policies focus on integrity rather than confidentiality, because most commercial and industrial firms are more concerned with accuracy than disclosure. An integrity policy is a security policy dealing only with integrity. The strict integrity policy is most commonly called "Biba's Mode1", in which integrity labels are assigned to the objects and subjects in a

system. This model is the mathematical dual of the Bell-LaPadula model. Obviously, if the component services, transitions and places are assigned integrity level, the integrity feature of the Petri nets-based model for Web service composition can also be verified by coverability graph.

Let I(S) be the integrity level of service S. To preserve integrity, the following rules must be met while composing Web services:

For composite service S= Seq($S_1$,$S_2$), $I(S_2) \leq I(S_1)$.

For composite service S=Conc($S_1$,$S_2$),$I(S_1) \leq I(S)$, and $I(S_2) \leq I(S)$.

For composite service S=Choice($S_1$,$S_2$),$I(S_1) \leq I(S)$, and $I(S_2) \leq I(S)$.

Because the Biba's model is the mathematical dual of the Bell-LaPadula model, the above conclusions can be proved like their counterparts in Bell-Lapadula model

## 5. Related work

In the research related to Web services, several initiatives have been conducted with the intention to provide platforms and languages that will allow easy integration of heterogeneous systems. In particular, such languages as UDDI, WSDL, SOAP and part of DAML-S ontology (ServiceProfile and ServiceGrounding), define standard ways for service discovery, description and invocation (message passing). Some other initiatives such as BPEL4WS and DAML-S ServiceModel, are focused on representing service compositions where flow of a process and bindings between services are known a priori(Rao J & Dustdar S, 2005). Ontology-driven Web services composition is used to discover and assemble services into processes for easier and better quality workflow executions given increasing number and complexity of Web services(Budaka I,et al.2005).

| Model | Petri type | Data type distinguishable | Auto composition | Additional message | Availability verification | Security analysis |
|---|---|---|---|---|---|---|
| CPWSC | Colored | Yes | Yes | No | Yes | Yes |
| Guo 2006 | Colored | Yes | No (pre-defined rule and conditions needed) | No | Yes | Not mentioned |
| Qian 2006 | Elementary | No | Yes | Yes | Yes | Not mentioned |
| Hamadi 2003 | Elementary | No | Yes | No | No | Not mentioned |

Table 1. Comparison of Petri nets-based models for Web service composition

Besides that, current solutions for Web service composition include web components, pi-calculus, Model check-ing/FSM and Petri nets(Milanovic N &Malek M.2004). In 2003, Hamadi proposed a Petri nets-based model for Web service composition (Hamadi R,2003), in which the data types cannot be distinguishable because an elementary Petri nets model is used. In a recent research, a CP-net model for Web service composition is proposed (Guo 2006) (Guo Yubin,et al.2006).However the rules and procedures of composition must be defined previously, and the services composition chain cannot be generated automatically without pre-defined conditions. In the message oriented activity based Petri nets model (Qian 2006) (Qian Zhuzhong, et al.2006),the Web service composition relies on messages,

which increase complexity of com-position. A comparison of Petri nets-based models for Web service composition is shown in Table 1, in which CPWSC represents the model proposed in this chapter.

## 6. Conclusion

In this chapter, we proposed Petri nets-based algebra for composing Web services. The formal semantics of the composition operations is expressed in terms of P/T Petri nets and CPNs by providing a direct mapping from each operator to Petri nets construction. In addition, the use of a formal model allows verification of closure, availability, and security properties and detection of inconsistencies both within and between services (Zhaoli Zhang, et al.2008) . There are other issues in B2B E-commerce which can be successfully addressed by extending the framework presented in this chapter. Further work will include a more thorough analysis of the field in addition to practical testing experiments with the methods.

## 7. References

Bell D, Lapadula L. (1996). The Bell-LaPadula model [J] *Journal of Computer Security,* 1996, 4(2-3): 239-263.

Budaka I, Aleman-Meza B, Zhang R, et al. (2005). Ontology-driven web services composition platform [C]*// Proceedings of IEEE International Conference on E-commerce Technology,* San Diego. Los Alamitos: IEEE Computer Society Press, 2005: 146-152.

Cordoso, J.; Sheth, A. (2002). Semantic e-Workflow Composition, *Technical Report,* LSDIS Lab, Computer Science, University of Georgia, July 2002

Dustdar S, Schreiner W. (2005). A survey on web services composition [J] *International Journal of Web and Grid Services.* 2005, 1(1): 1-30.

Guo Yubin, Du Yuyue, Xi Jianqing. (2006). A CP-net model and operation properties for web service composition [J]. *Chinese Journal of Computers,* 2006, 29(7):1067-1075 (in Chinese).

Hamadi R, Benatallah B.(2003). A Petri nets-based model for web service composition [C] *//Proceedings of the 14th Australasian Database Conference,* Adelaide. Darlinghurt: Australian Computer Society, 2003: 191-200.

Huaikou Miao, Tao He, Zhongsheng Qian,(2008). Modeling and Analyzing Composite Semantic Web Service Using Petri nets, *4th Workshop on Service-Oriented Applications, Integration and Collaboration (SOAIC'08),* Oct 24, 2008, Xi'an, China

J.P. Thomas, M. Thomas, and G. Ghinea,( 2003). "Modeling of Web Services Flow", *Proceedings of the IEEE International Conference on ECommerce (CEC'03).*

Jensen K. (1994). An introduction to the theoretical aspects of colored Petri nets [J]. *Lecture Notes in Computer Science.* 1994. 803: 230-272.

Knorr K. (2001). Multilevel security and information flow in Petri nets workflows [C]*//Proceedings of the 9th International Conference on Telecommunication Systems-Modeling and Analysis, Special Session on Security Aspects of Telecommunication Systems,* Dallas. Los Alamitos: IEEE Computer Society Press, 2001: 9-20.

Martin, D. et al. (2004). Bringing Semantics to Web Services: The OWL-S Approach, *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition,* San Diego, July 2004

Maurice ter Beek, Maurice; Bucchiarone, Antonio; Gnesi, Stefania; (2007). Web Service Composition Approaches: From Industrial Standards to Formal Methods Internet and Web Applications and Services, *2007. ICIW '07. Second International Conference* on 13-19 May 2007 Page(s):15 – 15 Digital Object Identifier 10.1109/ICIW. 2007.71

Milanovic N, Malek M. (2004). Current solutions for web service composition [J]. *IEEE Internet Computing,* 2004, 8(6): 51-59.

Narayanan S, Mcilraith S. (2003). Analysis and simulation of web services [J]. Computer Networks, 2003, 42(5): 675-693.

Qian Zhuzhong, Lu Sanglu, Xie Li. (2006). Automatic Composition of Petri nets based web services[J]. *Chinese Journal of Computers,* 2006, 29(7): 1057-1066 (in Chinese).

Rachid Hamadi, Boualem Benatallah, (2003). A Petri nets-based Model for Web Service Composition, *Proceedings of the Fourteenth Australasian database conference on Database technologies 2003,* ACM Press, Adelaide, Australia, February 1, 2003, pp.191-200.

Rao J, Su X. (2005). A survey of automated web service composition methods [J]. *Lecture Notes in Computer Science,* 2005, 3387: 43-54.

Ren Z H, Cao J N, Chan T S, (2003). Composition and automation of grid services [J]. *Lecture Notes in Computer Science,* 2003, 2834: 352-362.

S. Narayanan and S. Mcllraith. (2002). Simulation, Verification and Automated Composition of Web Services, *Proceedings of the 11th World Wide Web Conference,* Honolulu, HI, USA, July, 2002, pp.77-88.

Schuster H, Georgakopoulos D, Cichocki A, et al. (2000). Modeling and composing service-based and reference process-based multi-enterprise processes [J]. *Lecture Notes in Computer Science,* 2000, 1789: 247-263.

Thomas J P, Tomas M, Ghinea G. (2005). Modeling of web services flow [C]*//Proceedings of IEEE International Conference on E-commerce,* San Diego, California. Los Alamitos: IEEE Computer Society Press, 2005: 391-398

Zhang Jia, Chung Jenyao, Chang C.K., Kim S., (2004). "WS-Net: A Petri-net based specification model for web services", *Proceedings of the second IEEE International Conference on Web Servics,* IEEE Press, San Diego, California, USA, pp.420-427.

Zhaoli Zhang, Fan Hong, Haijun Xiao,(2008). A colored Petri nets-based model for web service composition,*Journal of Shanghai University (English Edition),* 2008,12(4):323–329

Zhijun Ding; Junli Wang; ChangJun Jiang; (2005). Semantic Web Service Composition Based on OWL-S. *Semantics, Knowledge and Grid, 2005. SKG '05.* First International Conference on Nov. 2005 Page(s):98 - 98