Journal of International Technology and Information Management

Volume 18
Issue 3 *Double Issue 3/4*

Article 6

2009

PENTRAL: Pattern Based Logic Language

Manuj Darbari
Babu Banarasi Das National Institute of Technology and Management

Abhay Kumar Srivastava
Sherwood Business School

Sanjay Medhavi
University of Lucknow

PENTRAL: Pattern Based Logic Language

Manuj Darbari,
Babu Banarasi Das National Institute of Technology
INDIA

Abhay Kumar Srivastava
Sherwood Business School
INDIA

Sanjay Medhavi
University of Lucknow
INDIA

ABSTRACT

The article focuses on the urban traffic representation using patterns which are specifically developed for traffic situations. The framework PENTRAL (PEtri-Net TRAnsportation Language) provides conversion of these patterns into algebraic format. These formats are saved as an XML file following the three tier architecture of urban Traffic system.

INTRODUCTION

The increase of Motor vehicles in recent years has been phenomenal and importance of road traffic in supporting the social and economic activities every increase; as the result, road congestion, accident, air pollution is becoming serious year after year. In urban area, it has been difficult to implement a drastic plan like a new road construction as a resolution for traffic congestion (Asgekar, 2003; Wray, Markham, & Mathieu, 2003; Daribari, Medhavi, & Srivastava, 2007; Daribari, Medhavi, & Srivastava, 2008). So we are forced to find out other practicable solutions based on individual conditions of actual road configuration and available technologies. It is beneficial to develop some representation format which could support dynamic graphical representation. The Microscopic graphical representation could contribute to this requisite as a competent tool to explain details of relationships between traffic flows and proposed alternatives both from logical and experimental points of view.

Most of previous studies worked on architectural analysis but there is little being done on development of Language development of Workflow system with special emphasis on urban Traffic systems (Darbari & Bhaskar, 2008; Dvorak & Novak, 2003; Georgakopoulos, Homick, & Sheth, 1995; Panagos & Rabinovich, 1996; Knybel, 2005). We have shown the traffic flow by the help of Petri-net graphs Wang, 1998; Kepuska, Grubuz, rodriguez, Fiore, Carsten, converse, & Metcalf, 2008; Zhang, Lin, & Hsieh, 2008). The use of Petri-nets have been criticized for the use in modeling complex business processes, specially when dealing with multilayer graphical representation but Petri-net Transportation Language (PENTRAL) provides an extension to Petri-net theory providing a set of techniques to describe complex traffic management process. It could offer a technique with perfect verification capabilities and comprehensible models. The purpose of this paper is to show how Petri-net language can provide pattern interactions. PENTRAL will provide an easy representation of objects like route section and link in which

represent a part of the layered tree structure. The control flow aspects of urban Traffic system will define the alphabet of a labeled Petri-net as the set of activities of the process, the Petri-net generate a language over the task of the processes:

LITERATURE REVIEW

Urban TNs (Transport Networks) exhibit high degree of concurrency and are characterized by resource sharing and conflicts. Hence, appropriate models of these systems have to take into account such distinctive features, in order to result in efficient traffic management strategies. In particular, urban TNs can be viewed as event driven and asynchronous systems. Their dynamics depends on the complex interactions of the timing of various discrete events, such as arrivals or departures of vehicles at intersections and beginning or completion of the various phases in the signal timing plans of the traffic lights controlling junctions. Thanks to the well known ability of Petri Nets (PNs) to capture concurrency and asynchrony, PN based models may be suitably derived for urban traffic systems. More precisely, PNs can be employed both for describing traffic signals controlling urban signalized areas, as well as for modeling concurrent activities that are typical of TNs. Tensen (1992) present a traffic model in a timed PN framework, where tokens are vehicles and places are parts of lanes and intersections. Since in PNs tokens cannot distinguish among different vehicles and their associated routes, colours are introduced in (Murata, 1989), where a different number (colour) is assigned to each vehicle entering the system. The model is realized by defining appropriate subnets modeling links and intersections:

PATTERNS IN PETRI-NETS REPRESENTATION

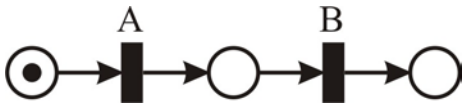
Traditionally most techniques used for the analysis of Business process originate from operation research. The focus is on performance analysis of the workflow model (Dvorak & Novak, 2003; Novak, 2000; Sheth, Joosten, Scacchi, & Wolf, 1996; Tensen, 1992) rather than correct and precise representation. In order to streamline our work we will specifically focus on basic control patterns related to urban Traffic system

PATTERN 1: SEQUENCE PATTERN

A sequence pattern contains two or more ordered activities that are performed sequentially, i.e. an activity starts after a previous activity has completed. This pattern is easily implemented by means of the basic Petri Net constructs: for each activity a transition is created and the transitions are connected with each other by means of arrows and places. The sequence flow direction is determined by the flow relation, e.g., the arrows in Figure 1.

Additionally, to define a Petri-net language, we need to specify the labeling function, the beginning and the final marking. Obviously, the labeling function shall rename the transitions with the names of the activities they represent. We define the begin and the set of final markings for this Petri-net as follows: $\mu 0 = (1,0,0)$ and $F = \{(0,0,1)\}$. The labeled Petri Net $PN = (N,\tau,\mu_0,F)$ defines the language $L(PN) = \{AB\}$, i.e. activity B is only executed after the completion of activity A.

Figure 1: The sequence pattern.

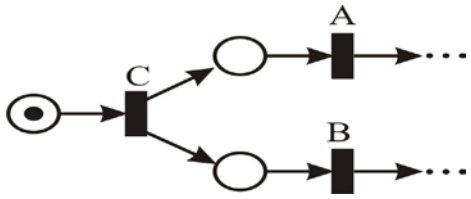


Pattern 2: Point to Many Points: Parallel Routing

The pattern is defined as being a mechanism that allows activities to be executed concurrently. The single thread of control is split into two or more threads, which means that the activities can be executed at the same time or in any order. In fact, the parallel split is used when there is no sequence constraint defined on a set of activities. This pattern is also easily implemented by means of the basic Petri-net constructs: a transition is connected to multiple (output) places, i.e. the firing of this transition will enable multiple transitions at the same time e.g., the firing of transition c enables transitions A and B, see Fig. 2.

Again, a Petri Net language is defined y specifying the labeling function and the beginning and final marking. There are no special requirements in the definition of the begin marking $\mu 0$ = (1,0,0,...,0) and the set of final markings F = {(0,0,0,....,1)}. The possible simultaneous execution of the same activity has no meaning. Moreover, whenever such a construction seems convenient, it in fact turns out that, in business terms, the activities have a different containment. The language defined by the Petri Net in figure 2 is L(PN) = {CAB..., CBA....}.

Figure 2: The parallel split pattern.

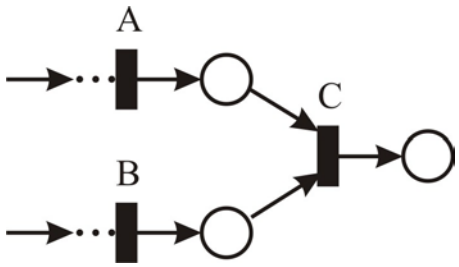


PATTERN 3: SYNCHRONISATION POINTS : ROUTE CONVERGENCE

The pattern is used to merge the different threads that are started by a parallel split. This means that all the threads of the parallel split must be completed before the process can continue. In Petri-net terminology it is implemented by connecting the places of each concurrent thread with one new transition. This means that each parallel thread needs to finish (add a token in the place) before the process can continue with the next activity, eg. transitions A and B need to fire to enable the transition C, as shown in figure 4.3.

The Petri Net language that we need to define for this pattern has no special features, and can be specified as follows : $\mu 0$ = (1,...,0,0,0) and the set of final markings F = {(0,...,0,0,1)}. Thus, L(PN) = {...ABC,...BAC}

Figure 3: The synchronization pattern.

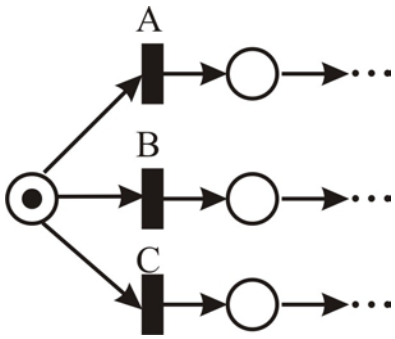


Pattern 4: Restrictive Point / Route

This pattern defines a place in the process where exactly one of multiple exclusive threads is executed. The pattern supports conditional behaviour and is also directly supported by basic Petri Net constructs. Connecting several transitions with one place results in a situation where multiple transitions are enabled but the firing of one will disable the others, eg., if transition A is fired, transitions B and C are disabled , shown in Figure 4. The begin marking and set of final markings are defined as follows:

μ_0 (1,0,0,0,....,0) and F = {(0,0,0,0,...,1)}. Some special attention is needed for the labeling function as the exclusive choice pattern defines a set of transitions hat are enabled at the same time. An additional restriction is specified: each activity in the exclusive choice pattern should be unique, i.e. the labeling of each transition in the pattern should be unique. The case where transitions with the same label are allowed breaks the determinism constraint. This, again, is a plausible constraint in business process terms. The language defined by the Petri Net is L(PN)= (A...,B...,C..).

Figure 4: The exclusive choice pattern.

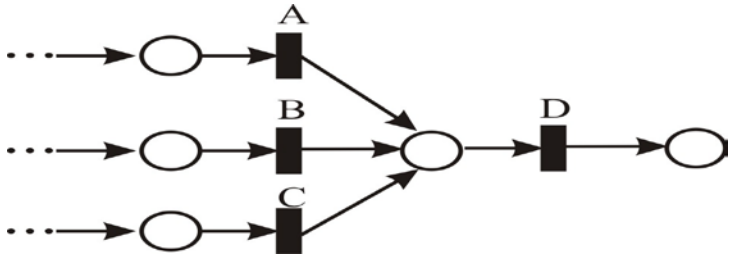


Pattern 5: Merge of Points : Multi To Urinary Routing

The pattern is used to bring together the paths of an exclusive choice pattern. This means that after the execution of one of the paths of the exclusive choice exactly one and the same activity needs to execute. This pattern is accomplished by connecting the last transitions of the different paths with the same place, the execution of the A, B or C branch will always enable transition D, as shown in Figure 5.

No additional constraints are required to define this pattern as a deterministic Petri net language, we just have to define the beginning marking and the set of final markings as follows : μ_0(1,.....0,0,0,0,0) and F = {(0,0,0,0,...,1)}. There is no restriction on the labeling function. L(PN) = {... AD, ..BD, ...CD}.

Figure 5: The simple Merge Pattern



FRAMEWORK FOR PENTRAL: PETRI-NET TRANSPORTATION LANGUAGE

Until now there is no formal language which is specifically designed to model Urban traffic flow. There exists number of papers in the field of Urban Traffic modeling using Petri-nets but none of hem could a customise Petri-nets specifically for urban traffic systems. Our approach provides customisation of Petri-nets into “Point” and “Route” Patterns and finally converts it into mathematical form providing an environment to model Traffic workflows. The advantage to convert eh entire Petri-net patterns into mathematical form as the graphical forms require a fool to be visualised and moreover these forms cannot be formally verified. PENTRAL provides a two way approach :

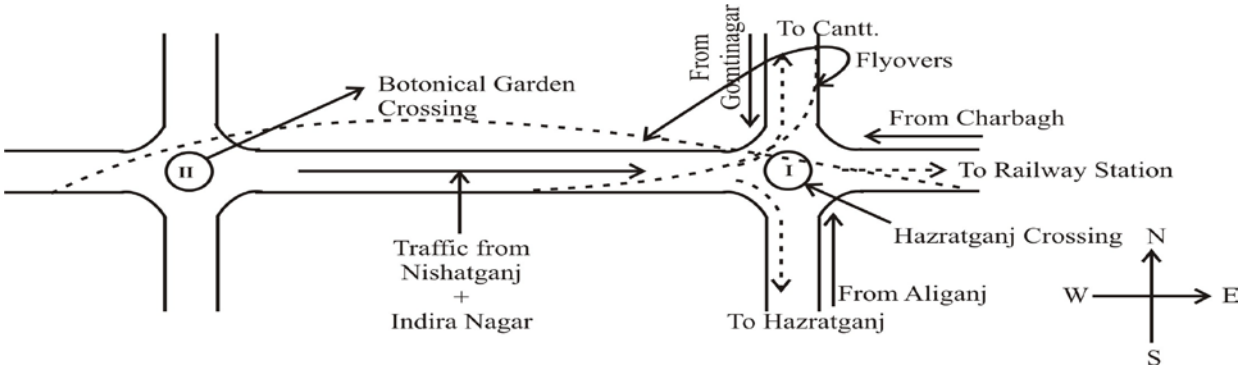
(i) Customisation of Petri-nets to "Point" and "Route" Patterns.

(ii) Mathematical representation of point and Route Petri-net patterns.

The Mathematical form of Petri-net is a linear equation format consisting of several statement separated by semi-colons places are enclosed within parenthesis add transitions within square brackets. Each token is represented by "@" co-reference variables are used for connecting various occurrences of a same place or transition. Commas are used for separating different branches of the graphs.

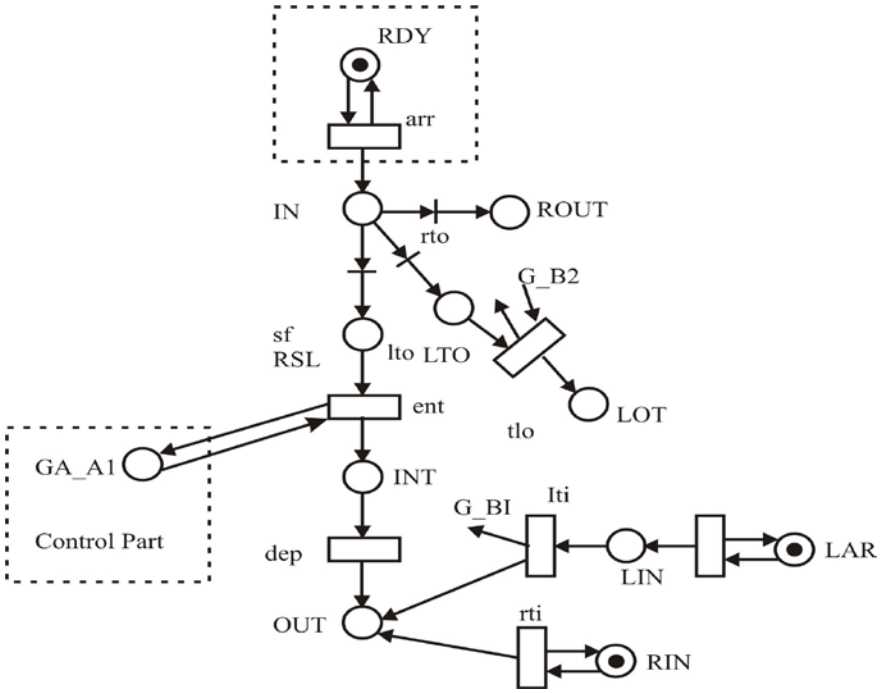
In order to apply Petri-net Transportation language let us consider a situation represent a simple graphical model with Hazratganj (Lucknow) as the Model point shown in figure 6.

Figure 6: Main Hazratganj Crossing at its interconnectivity. Traffic flow is from Crossing-II to Crossing-I.



We can represent the entire network by the help of 4-phase intersection model which represents the static nature of the flow of traffic shown in figure 7.

Figure 7: The Traffic flow model of 4-phase intersection.



PLACE:

RDY_A1	Incoming vehicles
IN_A1	Vehicles ready to enter intersection
RSL_A1	Ready for going straight or turning left
ROUT_A1	Right-turn-out vehicles
INT_A1	Vehicles entering intersection
MF_A1	Vehicles moveing forward
OUT_A1	Vehicles out
RIN_A1	Right-turn-in vehicles
LOUT_A1	Vehicles turn left out
LAR_A2	Left-turn-in incoming vehicles
LIN_A2	Left-turn-in incoming vehicles to enter intersection

TRANSITION:

arr_A1	Vehicles arrive
rto_A1	Vehicles right-turn out
Nrto_A1	Vehicles not right-turn out
lto_A1	Vehicles left-turn out
Nlto_A1	Vehicles not left-turn out
ent_A1	Vehicles enter intersection
Dep_A1	Vehicles depart intersection
lti_A1	Vehicles left-turn in
rti_A1	Vehicles right-trun in
alt_A1	Left-turn-in vehicles arrive

PENTRAL provides concise and readable textual notations as graphical model are hard to be visualised and need to be transformed into images before exchanging them.

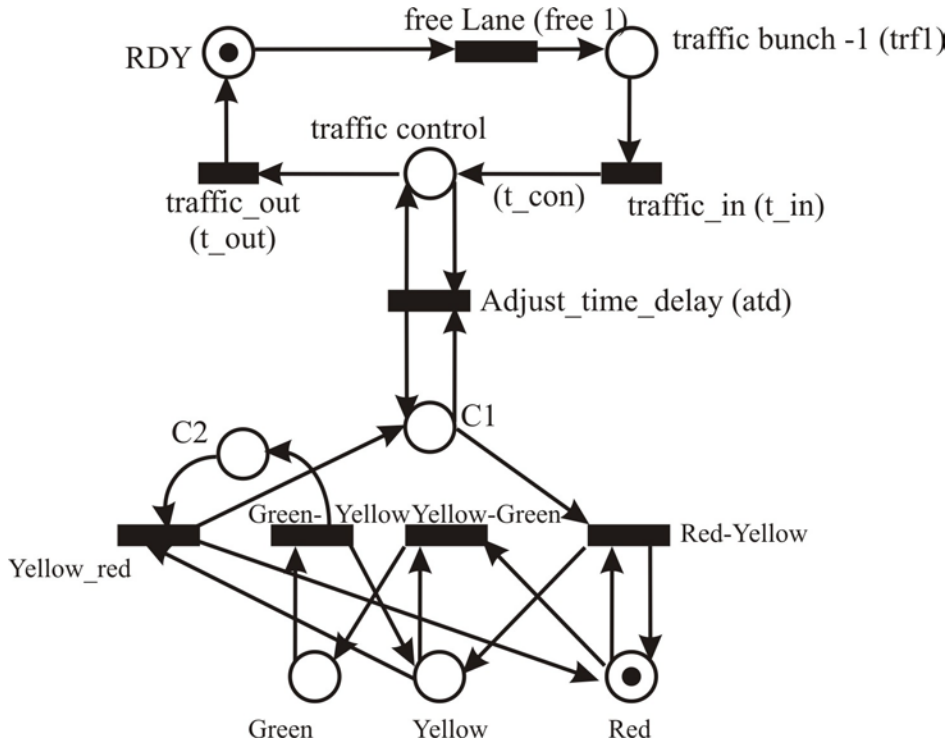
The format for PENTRAL can be represented by three basic steps:

Step 1 : (trf1*trf1@) → [freel1] → (trf2) → [freel2] → [*trf1]
Step 2 : (trf1*trf1) → [freel1] → (trf2@) → [freel2] → [*trf1]
Step 3 : (trf1*trf1) → [freel1] → (trf2) → [freel2] → [*trf1@]

Where: trf1 → represents traffic bunch No. 1 at particular time and place.
trf2 → represents traffic bunch No. 2 at particular time and place.

freel1 & freel2 → represents free lanes available for different traffic bunch @ represents the flow of token. These steps can be expanded for real time traffic simulation mode. The simplified version is a two layered structure (figure. 8) with traffic flow representation in the first layer and control section representing traffic lights in the second layer. The process starts from "RDY" showing the ready status. It is linked with free lane generation cycle which generates free lane for the traffic bunch. "Traffic bunch" represents the movement of traffic in store and forward mode occupying the road network of a particular section (figure 6) denoted by "traffic_in". In order to control the flow of traffic dynamically we link it with other layer represented as "control section". "Adjust_time_delay" control section provides a situational control mechanism which can dynamically change the sequence of operation of traffic lights and diversions. Traffic moving out of the particular section is represented by "traffic out". The entire module can be cascaded to form a real time Petri-net representation which is stored in Algebraic form as an XML file.

Figure 8: Simplified representation of Traffic flow and control in PENTRAL



The linear form of the above situation can be represented as:

(RDY@
*RDU { ←[freel1 *fr1]←(trf1 *trf1)← [t_in *t_in] ← (t_con *t-con) ←[atd *atd] {←(c1*c1@), →[*ry], →(yellow *y) {→[yellow_red *yr] {→(*c1), → (*r)},
→ [*yg] → (green) → [green_yellow]g
{→ (*y), → (C2) → [*yr]}
→ {[*atd]←(t_con) ←[t_out *t_out], [*t_out] { → (*t_out), →(*RDY).

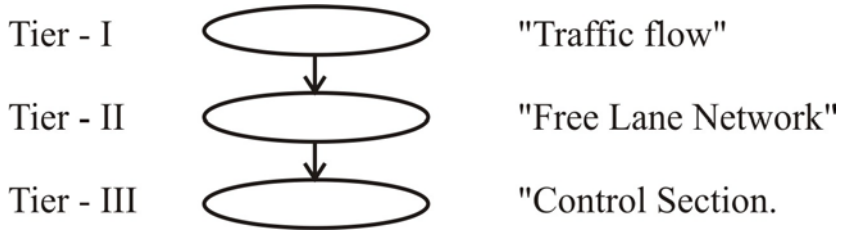
The entire algebraic form represents the movement of taken in Petri-net Traffic control system. The main purpose of developing a platform PENTRAL is to store the entire Petri-net notation in XML file, but it differs from PNXML in which the representation is only possible in static form. In our environment PENTRAL we can save the entire file in XML format with minimum tree level. The entire algebraic form represents the movement of token in Petri-net Traffic control system. The main purpose of developing a platform PENTRAL is to store the entire Petri-net notation in XML format but unlike PNXML where the representation is only in static form. In our environment PENTRAL we can save the entire file in XML format with minimum tree level.

<? Xml Version = "1.0"?>
<traffic flow> (RDY@ *RDY { ←[freel1 * frl 1] ←..... </traffic flow>

<control Signal> { ←(c1*c1@), → (ved*r@){ →[*ry], → [yellow_green *yg]} .... </ control Signal>

The dynamic aspect of modeling can be represented by algebraic format. It can be embedded with eclipse to support the linking with HPSIM 2.0. In order to define it in XML we have developed a simple three tier architecture The first tier relates to the route section which is depicted as “Traffic flow”. The second tier represents the necessary have structure which shows the “Free-lane” available. The third tier represents the “Control section”. This section represents the traffic light sequence and timings along with variable message signboards (VMS).

Figure 9: Three-tier Architecture for of Traffic Representation in XML.



CONCLUSION AND FUTURE SCOPE

In this paper, a new model for traffic movement is presented. The model is based on conversion of Petri-net flow graph into textual format. These equations describe the model behavior in both, the static mode and dynamic mode. A special Three-tier conversion factor is developed which makes the conversion and storage of models easier. PENTRAL framework provides all the functionalities which can support the conversion of Urban Traffic Petri-net representation into its algebraic form. In future we are planning to incorporate additional practical constraints in control section and traffic flow section

REFERENCES

Asgekar, V. (2003). Event Management Graduates with Distinction. *Supply Chain Review*, 7(5), 15-16.

Darbari, M., Medhavi, S., & Srivastava, A. K. (2007). Application of UML for Modeling Urban Traffic System Using Producer Consumer Theory to Generate Process Algebra Model. *Journal of International Technology and Information Management*, 16(4). 75-82.

Darbari, M., Medhavi S., & Srivastava A. K.(2008). Development of Effective Urban Road Traffic Management Using workflow Techniques for upcoming metro-cities like Lucknow (India). *International Journal of Highway and Information Technology*, Korea, 1(3).

Darbari, M. & Bhaskar, K. (2008). Enterprise Modeling using Unified Framework supporting Distributed Object Computing. *Journal of International Technology and Information Management*, 17(3-4), 205-218.

Dvorak, A. & Novak, V. (2003). The software package LFLC- 2000 - Its specificity, realization and perspective applications. *Computers in Industry,* 51, 269-280.

Georgakopoulos, D., Homick M., & Sheth, A. (1995). An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and parallel Databases,* 3(2), 119-153.

Këpuska, V. Z., Gurbuz, S., Rodriguez, W., Fiore, S., Carstens, D., Converse, P. D., & Metcalf, D. (2008). uC: Ubiquitous Collaboration Platform for Multimodal Team Interaction Support. *Journal of International Technology and Information Management,* 17(3-4), 263-284.

Knybel, J. (2005). Representation of Fuzzy IF-THEN rules by Petri-nets. In ASIS-Abstracts of Contributions to 2nd International Workshop on Data-Algorithms-Decision Making, December 10-12, 2006, Trest, Czech Republic.

Murata, T. (1989). Petri-nets: Properties, Analysis and Applications. *Proceeding of the Institute of Electrical and Electronics Engineers,* 77(4), 541-580.

Novak, V. (2000). *Fuzzy Sets and their applications.* Adam Hilger, Bristol.

Panagos, E. & Rabinovich, M. (1996). Escalations in workflow management systems, In DART Workshop, Rockville, Maryland.

Sheth, A., Joosten, S., Scacchi, W., & Wolf, A. (1996). A report from NSF workshop on workflow and process automation in information system, SIGMOD USA.

Tensen, K. (1992). Coloured Petri-nets: Basic concepts, Analysis, Methods and Practical Use, Springer - Verlag, Berlin.

Wang, J. (1998). Timed Petri-nets Theory and applications. K Kluwer Academic Publishers, Boston.

Wray, B. A., Markham, I. S., & Mathieu, R. G. (2003). An Artificial Neural Network Approach to Learning from Factory Performance in a Kanban-Based System. *Journal of International Technology and Information Management*, 12(2), 85-98.

Zhang, Y. J., Lin, Z., Lin, Q., & Hsieh,C-T. (2008). The Readiness For And Current Status Of E-Goverenment In China. *Journal of International Technology and Information Management*, 17(1), 75-84.