MODELLING PERFORMANCE MONITORING OF IT INFRASTRUCTURE COMPONENTS USING TIMED PETRI NETS

Ammu Qudsiya .A*

Research Scholar, Department of Mathematics,
Mother Theresa Women's University, Kodaikanal, India.
qudsiya_ansari@yahoo.com

Dr. K. Rangarajan

Director, Aryabatta Centre for Discovery Analysis,
Department of Master of Computer Applications,
Bharath University, Chennai, India.
kr2210@hotmail.com

Abstract

This paper focuses on utilization of Petri Nets as a modelling and simulation tool for analyzing performance monitoring of IT infrastructure. In order to make informed fine tuning decisions, modelling and simulation based performance analysis is essential. Petri Nets allows study of performance parameters and dynamic system behavior graphically. Mapping of performance monitoring scenarios in IT infrastructures to Timed Petri Nets and its structural analysis is illustrated in this paper

***Keywords*:** Petri Net, Timed Petri Net, Performance Monitoring, Performance Analysis.

1. Introduction

Today even small organizations comprises of a variety of IT infrastructure components (servers, routers, switches, firewalls, printers, software components, etc). With the ever increasing dependency of Organizations on their IT backbones for their everyday activities, availability and performance have become a critical ingredient for their success. With such critical role, IT infrastructure performance monitoring and management plays a vital role. The key challenge faced by IT infrastructure managers is predicting, analysing and isolating problems. Hence a variety of performance monitoring (predictive, real-time and historical) and modelling techniques are used to analyse and isolate performance issues and failures.

Conventional modelling tools (Network Diagrams, PERT, CPM, PDM, etc) provide limited scope for modelling such complex IT infrastructures with multiple interdependencies. Performance degradation of one layer may ripple and affect the performance of other layers. Hence the chosen modelling tool should allow depiction of interdependencies, cause-effect relationships, conditional branching, hierarchical representations etc for successful modelling. This paper is a novel effort to show how Timed Petri Nets can be used to model such dynamic environments.

The aim of this paper is to develop a theoretical framework and modelling approach for performance monitoring of IT infrastructure components. In section I we have introduced IT infrastructure components and the

importance of monitoring. Monitoring will then be defined in terms of the problem that we are going to address. In section II we introduce Petri Nets and Timed Petri nets. In section III mapping of monitoring onto timed Petri nets is done. In Section IV Petri net representation of monitoring is done by taking a small illustration. Structural analysis is done using the Petri net analysis techniques.

Note*:
Partial order for certain pairs of elements in the set one of the elements precedes the order.
$(C \nrightarrow D)$ denotes the set of all partial functions from C to D.
$\mathcal{P}$ (A) is the power set of A.

Performance Monitoring is one of the key activities in complex IT environments. Monitoring is checking something over a period of time in order to see how it works. Monitoring IT infrastructure components helps to make the necessary changes to improve/regulate. Monitoring IT infrastructure components is useful in predicting, analyzing and isolating etc of performance issues. These tasks are taken care of by some resources. Some of the resources are managers, agents, alert interfaces etc. There may be some orders in which these tasks are carried out. The tasks may be taken care on priority basis, which satisfies constraints. All the tasks consume time. Tasks are carried out by capable resource sets.

This leads to the following definition:

Definition.1.1

A **monitoring** is a 5 tuple Mg = (T,R,PRE,TS,MT) satisfying the following requirements.

- T is a finite set of tasks.
- R is a finite set of resources.
- PRE $\subseteq$ T X T is a partial order, the precedence relation*.
- TS is the time set.
- MT $\in$ (T x $\mathcal{P}$ (R)) $\nrightarrow$ TS* defines for each task t :
  The resource sets capable of monitoring task t and
  The monitoring time required to monitor t by a specific resource set.

The definition specifies the data required to formulate performance monitoring of IT infrastructure.

The tasks are denoted by T and the resourced are denoted by R. The precedence relation PRE is used to specify precedence constraints. If a task t has to be monitored before task t' then (t, t') $\in$ PRE. The execution of task t has to be completed before the execution of t' may start. TS is the time set. N and R+ U {0} are typical choices of TS.

Monitoring a task (MT) needs two things.

- The resource sets capable of monitoring task t: $\{rs \in \mathcal{P}(R) / \langle t, rs \rangle \in dom(MT)\}$
- The monitoring time required to process t by a specific resource sets: MT{<t, rs>}

Assumptions:

We have made a number of assumptions about the structure of a monitoring problem.
(1) No resource may monitor more than one task at a time.
(2) Each resource is continuously available for monitoring.
(3) No pre-emption: (i.e.,) each operation, once started must be completed without interruptions.
(4) The monitoring times are fixed and known in advance.

2. Timed Petri Nets

Introduction to Petri nets:

A brief overview of Petri net is provided in this section, A detail definition can be found in [1][2][3].The classical Petri net is a directed bipartite graph with two node types called places and transitions. The nodes are connected via directed arcs. Connections between two nodes of the same type are not allowed. Places are represented by circles and transitions by bars. A place p is called an input place of a transition t if there exists a directed arc from p to t. Places may contain zero or more tokens. Tokens are denoted by black dots. A transition is enabled if each of its input places contains atleast w(p,t) tokens, where w(p,t) is the weight of the arc from p to t.

Definition 2.1 Petri Nets

A Petri net with n places and m transition can be represented by the multi-tuple.

$PN=(P,T,I,O,M_0)$ where

$P=\{p_1,p_2,p_3,p_4.....p_n\}$ is the set of places.
$T=\{t_1,t_2,t_3.....t_n\}$ is the set of transitions.
I is the input arc function, which is a mapping of $PxT \rightarrow N$, such that if there exists k input arcs connecting p_i to t_j, then $I(p_i,t_j)=k$;
O is the output arc function which is a mapping of $TxP \rightarrow N$, such that if there exists k output arcs connecting t_j to p_i, then $O(t_j,p_i)=k$;
$M_0 : P \rightarrow \{0,1,2,...\}$ is the initial marking of the Petri net, such that if initially there are k tokens in place i then $M_0(p_i)=k$, for $p_i \in P$ and $k \in N$. The marking of the Petri net represents the state of the net.

Definition 2.2 Reachability graph

The reachability graph has the marking of the Petri net (or state of the Petri net) as a node. An arc of the graph joining M_i with M_j represents the transition when firing takes the Petri net from the marking (state) M_i to the marking M_j. By computing the reachability graph it is possible to analyse all possible firing sequences.

This part covers some of the most important properties of Petri nets such as reversibility, liveness, boundedness and reachability. Boundedness and liveness are related to deadlock avoidance.

Definition 2.3 Reachability

A marking M_j is said to be reachable from marking M_i if there exists a sequence of transition that takes the Petri net from state M_i to M_j. The set of all possible markings that are reachable from M_0 is called the reachability set and is defined by $R(M_0)$.The concept of reachability is essential for the study of the dynamic properties of a Petri net.

Definition 2.4 Liveness

A Petri net is said to be live for a marking M_0, if for any marking in $R(M_0)$ it is possible to fire a transition. The liveness property guaranties the absence of deadlock in a Petri net. This property can also be observed from the reachability graph. If the reachability graph contains an absorbent state, then the Petri net is not live at that state and it is said to have a deadlock.(1)

Definition 2.5 Boundedness

A Petri net is said to be bounded or k-bounded if the number of tokens in each place does not exceed a finite number k for any marking in $R(M_0)$.Furthermore a Petri net is structurally bounded if it is bounded for any finite initial marking M_0.A Petri net is said to be safe if it is 1-bounded.

Definition 2.6 Reversibility

A Petri net is reversible, if for any marking in $R(M_0)$,M_0 is reachable. This means that the Petri net can always return to the initial marking M_0.

Structural analysis

The liveness and boundedness of the net will be assessed by using P- invariants and T-invariants. These invariants are obtained from the incidence matrix of the net and they are used to assess the overall liveness and boundedness of the net.

Definition 2.7 Incidence matrix

Let a_{ij}^{+} =w(i,j) be the weight of the arc that goes from the transition ti to place pj and a_{ij}^{-} =w(j,i) be the weight of the arc from place pj to transition ti. The incidence matrix A of a Petri net has |T| number of rows and |P| number of columns. It is defined as A=[aij] where aij = $a_{ij}^{+} - a_{ij}^{-}$.

Net invariants

Let A be the incidence matrix. A P-invariant is a vector that satisfies the equation Ax=0 and a T-invariant is a vector that satisfies the equation $A^T y=0$

Boundedness assessment:

A Petri net model is covered by P-invariants if and only if, for each place p in the net, there exists a positive P-invariant x such that x(p)>0.Furthermore, a Petri net is structurally bounded if it is covered by p-invariants and the initial marking M_0 is finite.

Liveness assessment:

A Petri net model is covered by T-invariants if and only if, for each transition t in the net, there exists a positive T-invariant y such that y(t)>0.Furthermore, a Petri net that is finite is live and bounded if it is covered by T-invariants. This is a necessary condition but not sufficient.

For real systems we need to model durations and delays. So a timing concept is introduced. There are many ways to introduce time into classical Petri net [4][5].In this paper a timing mechanism is used where time is associated with transitions. Each transition has a time delay associated with it. Because of the firing delay it takes some time before the produced tokens become available for consumption.

Definition 2.8

A **Timed Petri net** is a six tuple

TPN=(P,T,I,O,TS,D) satisfying the following requirements,

- P is the finite set of places.
- T is a finite set of transitions.
- $I \in T \rightarrow \mathcal{P}(P)$ is a function which defines the set of input places of each transition.
- $O \in T \rightarrow \mathcal{P}(P)$is a function which defines the set of output places of each transition.
- TS is the time set.
- $D \in T \rightarrow TS$ is a function which defines the firing delay of each transition.

The state of a timed Petri net is given by the distribution of tokens over the places and corresponding time stamps. Firing a transition results in a new state. A sequence of states $s_0,s_1,s_2....s_n$ can be generated such that s_0 is the initial state and s_{i+1} is the state reachable from s_i by firing a transition. Many firing sequence are possible. Let s_0 be the initial state of a timed Petri net. A state is called reachable state if and only if there exists a firing sequence $s_0,s_1,s_2,s_3....s_n$ which visits this state. A terminal state is a state where none of the transitions is enabled (a state without successors).

3. Mapping monitoring onto Petri Nets.

To show that Petri Nets can be used to model and analyse monitoring problems. We provide a translation from a monitoring problem to a suitable timed Petri net. This means that we have to map concepts such as tasks, resources and precedence onto places and transitions [6].

Given a task t we identify three stages:

- t is waiting to be monitored
- t is being monitored
- t has been monitored

The Fig.(1) shows how we model a tasks in terms of a timed Petri Nets. Transition st_t and ct_t represents the beginning and termination of t respectively. The places sm_t, bm_t and cm_t correspond to the stages mentioned.

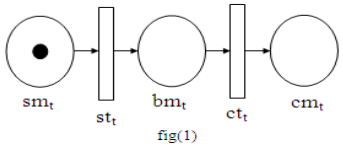

Fig.(1) Task t

Initially there is one token in sm_t. The firing delay is a delay time of task t to go to the next stage.

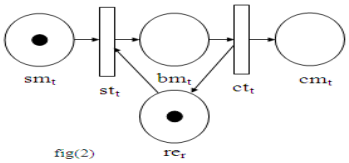

Fig.(2) Resource r:

Each resource r is modelled by a place re_r. Initially re_r contains one token. Fig.(2) shows a resource r which can be used to monitor a task t. Transition st_t claims the resource when the execution of t starts, transition ct_t releases the resources when t terminates.

Precedence constraints are modelled by adding extra places. Fig.(3) shows the situation where task t precedes task t' (ie) The execution of task t has to be completed before the execution of task t'. Place pre<t,t'> prevented $st_{t'}$ from firing until ct_t fires. Note the places are used to model the stages of a task, resources and precedence.

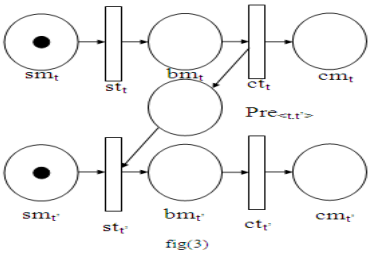

Fig.(3) Precedence constraint <t,t'>

A task with three possible resource sets.

Thus far we ignored the fact that a task may be processed by one of multiple resource sets. Fig.(4) shows how to model this situation. For each resource set rs capable of processing task t, we introduce a place bm<t,rs> and two transitions st<t,rs> and ct<t,rs>.Fig(4) shows that task t can be processed by one of the following resource sets $\{r_1\}$,$\{r_2\}$ and $\{r_1,r_2\}$. Note that there is only one start place sm_t and one completion place cm_t.

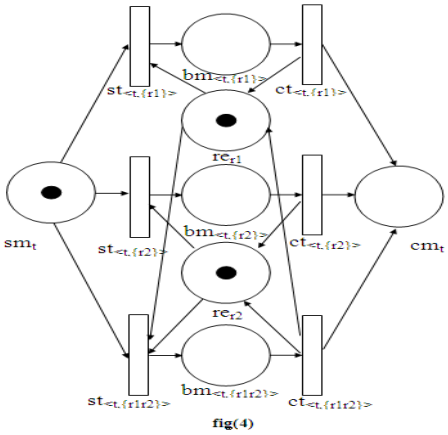

Fig.(4): A task with three possible resource sets.

Definition 3.1

Mapping:

Given monitoring problem Mg=(T,R,PRE,TS,MT)

we define the corresponding timed Petri net

TPN=($\overline{P}$,$\overline{T}$,$\overline{I}$,$\overline{O}$,$\overline{TS}$,$\overline{D}$) as follows.

$\overline{P}$ = $\{bm_{<t,rs>}/<t,rs> \in dom(MT)\} \cup \{sm_t/t \in T\} \cup$

$\{cm_t/t \in T\} \cup \{re_r/r \in R\} \cup \{pre_{<t,t'>}/<t,t'> \in PRE\}$

$\overline{T} = \{st_{<t,rs>}/<t,rs> \in dom(MT)\} \cup \{ct_{<t,rs>}/<t,rs> \in dom(MT)\}$

and for any task $t \in T$ and resource set $rs \in \mathcal{P}(R)$ such that $<t,rs> \in dom(MT)$.

$\overline{I}(st_{<t,rs>}) = \{sm_t\} \cup \{re_r/r \in R \text{ and } r \in rs\} \cup \{Pre_{<t,t'>}/t' \in T \wedge <t,t'> \in PRE\}$

$\overline{I}(ct_{<t,rs>}) = \{bm_{<t,rs>}\}$

$\overline{O}(st_{<t,rs>}) = \{bm_{<t,rs>}\}$

$\overline{O}(ct_{<t,rs>}) = \{cm_t\} \cup \{re_r/r \in R \wedge r \in rs\} \cup \{pre_{<t,t'>}/t' \in T \wedge <t,t'> \in PRE\}$

$\overline{TS} = TS$

$\overline{D}(st_{<t,rs>}) = MT(_{<t,rs>})$

$\overline{D}(ct_{<t,rs>}) = 0$

This definition shows how to model a monitoring problem in terms of a TPN. The initial state of the net is as follows. For each task t, place sm_t contains one token. For each resource r, place re_r contains one token.

We have some assumptions stated about the above model. Those assumptions can be relaxed. First of all, we assumed that each resource can process only one task at a time. If this assumption is dropped, then we have to deal with resources having a specific capacity and tasks requiring only a part of this capacity. Place re_r contains more tokens. If we allow pre-emption we have to split tasks into smaller tasks. Each sub-task corresponds to a phase in the monitoring of task t. This can be handled by hierarchical Petri Nets.

We assumed the monitoring times are known and fixed. The monitoring problem is deterministic. This can easily be extended to nondeterministic monitoring problems by using another timed Petri net model.

4. Timed Petri net model of performance monitoring of IT infrastructure

Performance monitoring has become ultra critical for organizations to have peak performance IT infrastructures. Vendors of hardware and software has understood this and have adopted and exposed a variety of interfaces like SNMP, WMI, Perfmon, JMX, Queries, Command interfaces, Custom APIs etc. Choice of the interface will depend on availability, resource overhead, depth of details needed etc. A majority of the monitoring solutions follow the manager-agent architecture. As per this architecture, software agents deployed on the various hosts of a network environment make periodic measurements that are reported to a central manager. To collect measurements the agents use various tests.

In this section, we provide a Petri net representation of monitoring based on the definition and concepts developed in section II and III. The objective of the work presented in this section is to present a proof of concept on how Petri nets can be used to model and analyse performance monitoring of IT infrastructure components. The example taken for the case here will monitor CPU usage on a system and alerts the user when it crosses a pre-determined threshold limit [7]. However, in a real world scenario, for a full fledged monitoring system, a variety of parameters at different levels may have to be monitored. The example taken may be extended to accommodate more parameters as needed in a real time system. In this paper, for simplicity of depiction and ease of understanding of the concepts, we have taken a single attribute CPU monitoring. The Algorithm and flow chart for the model is defined first and the Petri net model is presented.

4.1 Algorithm for a simple performance monitoring model, monitoring CPU usage levels in a computer.

1. Agent reads the value of the current CPU usage on the system
2. Agent Sends the value to the manager
3. Manager reads the acceptable threshold limit for CPU usage
4. Check if the CPU usage value is more than the set threshold limits.
5. If CPU usage is more than set threshold then raise alert else go to next step directly. Alerts can be via email, snmp traps, sms, event logs, user interface, etc or custom/external alert interfaces.
6. Sleep until next monitoring period.
7. Go back to step 1

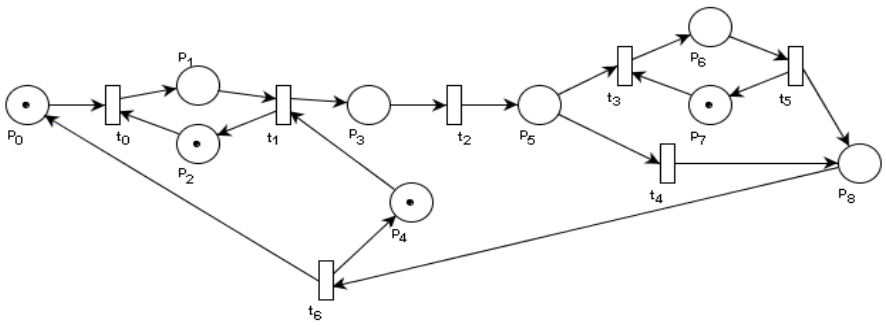

Fig.(5) Petri net model of monitoring CPU usage levels in a computer.

Resources: Manager, agents, alert mechanism
Tasks: Reading Usage values and Thresh hold values, comparing, raising alert, initiate next monitoring.

We now define the transitions and places in the Petri net model as follows:

Places

p_0, p_8, - represents start monitoring and complete monitoring and sleep for certain period.
p_1,p_3,p_5 - represents ready with the values.
p_6 - alert interface triggering fact
p_2 - agents

p_4 - manager
p_7 - alert mechanism
p_2,p_4, p_7 - denotes the availability of resources

Transitions

t_0 - reads CPU value
t_1 - send the CPU value to manager.
t_2 - manager reads threshold and calculating
$t_3$and t_4 - compares the values and conditions
t_5 - raise alert
t_6 - initiate the next monitoring

We assume that the time delay between each transition firing is 5 ms and the sleep time between each subsequent measure is 1 minute. Firing of t_3 and t_4 gives 2 different cases. The first case use t_3 to continue and raise alarm. The second case use t_4 to show the completion of monitoring and shows the system is in the safe zone.

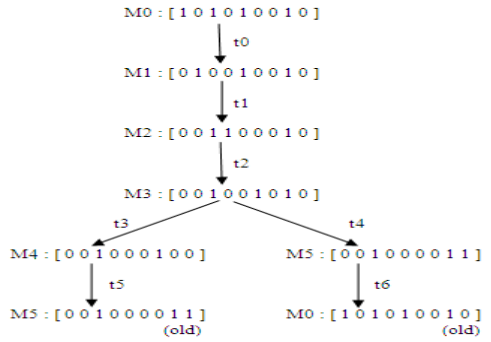

Fig.(6) Reachability Tree

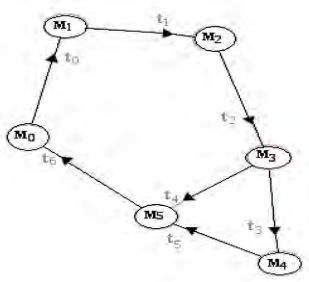

Fig.(7) Reachability Graph

Analysis of the reachability graph

At this point we are concerned with the net being bounded and live. An assessment of these properties based on the initial marking of the net can be performed by analysing the reachability tree of the net using M_0 as the root node as shown in Fig.(6). A reachability tree is a graph representation of the markings of a net. Each node in the tree represents a marking and the edges represent a transition firing. The resulting tree is small enough to be analysed by inspection and it can be concluded that

(a) The reachability set $R(M_0)$ is finite
(b) The number of tokens of every place in all marking is bounded (1-bounded)
(c) There are no dead transitions (all transitions can fire)

As a result it can be concluded that the net is bounded.

By looking at the reachability graph of the net with initial marking M_0 presented in Fig.(7) it can be seen that there is always an active transition regardless of the state of the net and all the transitions of T are included in the graph. From this it can be concluded that the Petri Net model of performance monitoring of CPU usage levels in a computer system is bounded and live. For models with a large number of places and transitions, the reachability graph construction is not practical without the use of computer tools to automate the reachability graph generation and the assessment of properties.

Structural analysis:

Place and transition invariants are powerful tools for studying structural properties of Petri nets. The structural behaviour of the net can be assessed using the algebraic analysis of the incidence matrix (invariant analysis).

The incidence matrix is defined as A= [a_{ij}] where $a_{ij} = a_{ij}^{+} - a_{ij}^{-}$

a_{ij}^{+} = w(i,j) is the weight of the arc from t_i to p_j and

a_{ij}^{-} = w(j,i) is the weight of the arc from p_j to t_i .

The incidence matrix of the net fig(5) is given below;

$$A = \begin{pmatrix} -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix}$$

Fig.(8) Incidence matrix

The order of the places and transitions in the matrix are

$P = \{p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$ (columns)
$T = \{t_0, t_1, t_2, t_3, t_4, t_5, t_6\}$ (rows)

A P- invariant is a vector that satisfies the equation Ax = 0
A T-invariant is a vector that satisfies the equation $A^T y=0$

The following invariants are obtained from the incidence matrix A.

P-invariants	**T-invariants**
$x_1= [1,1,0,1,0,1,1,0,1]^T$	$y_1= [1,1,1,0,1,0,1]^T$
$x_2= [0,1,1,0,0,0,0,0,0]^T$	
$x_3= [0,0,0,1,1,1,1,0,1]^T$	$y_2= [1,1,1,1,0,1,1]^T$
$x_4= [0,0,0,0,0,0,1,1,0]^T$	

A net is said to be covered by P- invariants if and only if, for each place p in the net there exist a positive P-invariant x such that x(p)>0. The net is covered by P-invariants since there is a positive element on x_1 or x_2 or x_3 or x_4 for every place, e.g., $x_1(p_2) = 0$ but $x_2(p_2) = 1$.In addition a net is covered by T-invariants if and only if for each transition t in the net, there exist a positive T- invariant y such that y(t)>0 .The net is also covered by T-invariant.

A Petri net is structurally bounded if it is covered by P-invariants and the initial marking M_0 is finite. Furthermore, a net is live and bounded if it is covered by T-invariants which is only a necessary condition. The Petri net Fig.(5) is covered by T-invariants and P-invariants. Since the initial marking is finite, we can conclude that the Petri net is bounded and that the necessary condition for liveness is met.

5. Conclusion

In this paper, we have emphasized the importance of performance monitoring in IT infrastructures and how modelling and simulation tools can help IT infrastructure managers to make informed fine tuning decisions. We have also showed how performance monitoring problems can be modeled in terms of a Timed Petri net formalism. We have provided a mapping from monitoring to a timed Petri net. A simple example of monitoring CPU usage is taken as case study and proposed methodology was applied to it. Structural analysis is done. Further work needs to be done in the future to develop and analyze dynamic behaviors of the model.

References

[1] Tadao Murata, Petri nets: properties, analysis and applications. Proceedings of the IEEE, Vol. 77, No. 4, Pages: 541-580, Apr 1989.
[2] James L. Peterson, "Petri Nets" ACM Computing Surveys Vol. 9(3), Pages: 223-252, 1977.
[3] James L.Peterson:Petri Net Theory And The Modelling of Systems. Prentice Hall, 1981.
[4] F.D.J.Bowdens, A Brief survey and synthesis of the roles of time in Petri Nets. Modelling 32(2000)55-68
[5] W.MZuberek,Timed Petri nets,Definition,Properties and ApplicationsMicroelectron Reliab.,Vol.31,No.4,pp.627-644, 1991.
[6] W.M.P.Vander Aalst Petri Net based scheduling Computing Science Reports 95/23,Bindhoven,july 1995.
[7] A.Ammu Qudsiya and K.Rangarajan.Petri Net based simulation of performance monitoring of IT infrastructure components, International Conference on Researching for transforming the society, Mother Teresa Women's University Kodaikanal,Tamilnadu. 2008.
[8] A.Ammu Qudsiya Generalised Stochastic Petri Net(GSPN)Based Simulation and Modelling, Nnational conference on Emerging Trends in Communicative English And Applied Sciences, Bharat University,Chennai. 2008.
[9] Richard Zurawski and Mengchu Zhou.,Petri nets and industrial Applications: A Tutorial proceedings of the IEEE Transactions on industrial Electronics,vol.41,No.6,December 1994.