World Applied Sciences Journal 3 (3): 504-513, 2008
ISSN 1818-4952

Analytic Evaluation on Petri Net by Using Markov Chain Theory to Achieve Optimized Models

[1]H. Motameni, [2]A. Movaghar, [3]M. Siasifar, [3]H. Montazeri and [4]A. Rezaei

[1]Department of Computer Engineering, Islamic Azad University, Sari Branch, Iran
[2]Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
[3]Department of Computer Engineering, Mazandaran University of Science & Tecnology, Babol, Iran
[4] Research office, Islamic Azad University, Sari Branch, Iran

Abstract: The quality of an architectural design of a software system has a great influence on achieving non-functional requirements of a system. A regular software development project is often influenced by non-functional factors such as the customers' expectations about the performance and reliability of the software as well as the reduction of the underlying risk. The evaluation of non-functional parameters of a software system at the early stages of design and its development process are often considered as major factors in dealing with these issues. Because these evaluations can help us to choose the most proper model which is the securest and the most reliable. In this paper, a method is presented to obtain performance parameters from Generalized Stochastic Petri Net (GSPN) to be able to analyze the stochastic behaviour of the system. The embedded Continuous Time Markov Chain (CTMC) is derived from the GSPN and the Markov chain theory is used to obtain the performance parameters. We have designed a case tool to obtain some performance parameters that we discuss about them in this paper in addition to a case study.

Key words: UML Generalized Stochastic Petri Net (GSPN) . Continuous Time Markov Chain (CTMC) . Non-Functional Parameters . Markov Reward Models

INTRODUCTION

A PN is an abstract, formal model of information flow. The properties, concepts and techniques of PNs are being developed for describing and analyzing the flow of information and control in systems, particularly systems that may exhibit asynchronous and concurrent activities. The major use of PNs has been the modeling of systems of events in which it is possible for some events to occur concurrently but there are constraints on the concurrence, precedence, or frequency of these occurrences[1].

There are three general characteristics of PNs that make them interesting in capturing concurrent object-oriented behavioural specifications. First, PNs allow the modeling of concurrency, synchronization and resource sharing behavior of a system. Secondly, there are many theoretical results associated with PNs for the analysis of such issues as deadlock detection and performance analysis. Finally, the integration of PNs with object oriented software design architecture could provide a means for automating behavioral analysis [1-4].

We present a method for obtaining non-functional parameters from GSPN. The reason for using GSPN it is that, there are some methods for transforming UML diagrams to GSPN for example one of them is introduced in [5].

Currently the Unified Model Language (UML) diagrams are widely used in the field of software design as it is easy to use comparing to the alternatives and is powerful in describing different aspects of the system. However, the semi-formal property of the UML diagram cannot satisfy the industry's need in predicting the non-functional parameters of the software in the early stages of the software life cycle.

Since it is not possible to use UML diagrams for performance evaluation, they were translated to Generalized Stochastic Petri Net (GSPN) [2, 3], a more formal model that enables the authors to do the performance evaluations.

The authors' previous work on AD includes the transformation of AD to Colored Petri Net [3, 4], where some performance measures could be obtained using simulation. The simulation-based measurements seam to be more straightforward compared to its alternatives, which are analytic methods.

We present at first a brief talk about GSPN, then we discuss about UML to introduce its fundamentals and history.

Corresponding Author: Dr. H. Motameni, Department of Computer Engineering, Islamic Azad University, Sari Branch, Iran

At the next step, we derive a CTMC from the GSPN. Finally, we do a performance evaluation on the derived CTMC. Then a case study is introduced which is explained by a case tool that we designed it.

In this research, analytic methods are used to obtain results that are more accurate. Although using these kinds of methods induces some computational complexities to the calculation of system performance, the gained results are more reliable comparing to the simulation techniques. Therefore, analytic methods remain the only choice for evaluating critical systems. However, we should consider that this method is more useful in small systems because it is possible to have more details.

RELATED WORK

Using Stochastic Petri Net (SPN) and its extensions have been discussed in several papers [2, 5-8]. Merseguer *et al*. used the derived SPN from the UML model to evaluate performance of internet based software retrieval systems [7]. Derivation of an executable GSPN model from a description of a system expressing a set of UML State Machines (SMs) was reported in [9].

A group of works is devoted to transforming the software model to Coloured Petri Net (CPN), which seems to be more related to software properties than the other UML extensions [10-15].

In the authors' previous works [4, 16], the UML model was transformed to CPN and then analyzed the CPN by means of simulation. Trowitzsch *et al.* have transformed the software UML diagrams to SPN models for performance evaluation of real-time systems [5].

Most of the previous works discuss about transforming the software model to analytical models or discuss about evaluating the performance model. In other words, none of them provides an integrated method, which can start from software model and terminate with some derived performance parameters.

The method of [9, 17] was used to transform the software model to a GSPN for evaluating software performance parameters. In our previous work [18, 19] by using [2] and transforming software model to GSPN some performance parameters are calculated. In this paper, we evaluate the performance model in the way that leads to gain meaningful parameters of the system like reliability, security, performance efficiency and availability.

REVIEWING THE GSPN AND UML

GSPN: The basic PN model includes two components: places and transitions connected together via arcs to model system behaviour; however, it may be extended by introducing the notion of time, leading to timed Petri nets for a performance analysis of Petri Nets quantitative analysis. A timed PN is called SPN, when random variables are used in specifying the time behaviour. Whereas, it has been shown that SPNs are, under certain conditions, isomorphic to homogeneous Markov chains, by analyzing metrics of the Markov chain (such as the steady state probability distribution) it is possible to investigate the behavior of the underlying system being modeled by the PN [20].

GSPN is defined as a PN $N=(P, T, W, M_0)$ with its transition set T divided into two sub-sets T_I and T_T, defining respectively the set of immediate and timed transitions. Immediate transitions are fired immediately once they are enabled, whereas, timed transitions are fired after a random, exponentially distributed, enabling time. Hence, in GSPN N, transitions $t \in T_T$ is associated with a (possibly marking-dependent) firing rate, $r(t)$ that constitutes the defining parameter of the corresponding exponential distribution.

The above characterization of immediate and timed transitions implies that in a net reachable marking, m, where, both, immediate and timed transitions are enabled, immediate transitions have precedence over the timed ones (since they are instantaneous). Furthermore, such a marking m has zero duration in the net dynamics and therefore, it is characterized as vanishing. On the other hand, a marking m in which all enabled transitions are timed transitions has duration; therefore, such a marking is characterized as tangible.

Given a marking m with a set of simultaneously enabled immediate transitions, $I(m)$, the modeler must provide a probability distribution regulating the firing of the transitions in $I(m)$. In the GSPN terminology, this probability distribution is characterized as a random switch $E= \{W_1, W2, W_{(m)}\}$. Furthermore, if the set of random switches regulating the net behavior are marking-dependent, they are characterized as dynamic; otherwise, they are static [21].

UML: UML consists in a set of graphs or charts with explanatory comments that can be expressed either in a formal way or in natural language. The designer can freely choose a subset of diagrams to present the system design. Activity diagram is the most important UML diagram that is used for presenting actions. An activity diagram is a dynamic diagram that shows the activity and the event that cause the object to be in the particular state. The activity is triggered by one or more events and it may result in one or more events that may trigger other activity or processes. The biggest disadvantage of activity diagrams is that they do not make explicit which objects execute which

activities and the way that the messaging works between them. However, labelling of each activity with the responsible object can be performed. Often it is useful to draw an activity diagram early in the modeling of a process, to help understand the overall process [22].

DERIVING THE EMBEDDED CTMC

The transformation algorithm is used to translate the activity diagram to the GSPN model is the one that is explained by Merseguer *et al*. [2]. The authors have constructed a formalism to transform the elements of UML behavioral models to GSPN and to integrate the outputs together.

The stochastic process associated with k-bounded GSPN systems with M_0, as their home state, can be classified as a finite state space, stationary (homogeneous), irreducible and continuous-time semi-Markov process [3]. In the case of GSPNs, the Embedded Markov Chain (EMC) can be recognized disregarding the concept of time and focusing the attention on the set of states of the semi-Markov process.

The specifications of a GSPN system are sufficient for the computation of the transition probabilities of such a chain. The CTMC associated with a given GSPN system is obtained by applying some simple rules:

- The CTMC state space $S = \{s_i\}$ corresponds to the reachability set $RS\ (M_0)$ of the PN associated with the GSPN ($M_i \leftrightarrow s_i$).
- The transition rate from state s_i (corresponding to marking M_i) to state s_j (M_j) is obtained as the sum of the firing rates (for timed transitions) or weights (for immediate transitions) of the transitions that are enabled in M_i and whose firings generate marking M_j.

Based on the simple rules listed above, it is possible to devise algorithms for the automatic construction of the infinitesimal generator (also called the state transition rate matrix) of the isomorphic CTMC, starting from the GSPN description. Denoting this matrix by U, with w_k the firing rate (or weight for immediate transitions) of transition T_k and with $E_j(M_i)$ the set of transitions whose firings bring the net from marking M_i to marking M_j, the components of the transition probability matrix would be:

$$u_{ij} = \frac{\sum_{T_k \in E_j(M_i)} W_k}{q_i} \qquad (1)$$

The components of the infinitesimal generator are:

$$q_{ij} = \begin{cases} \sum T_k \in E_j(M_i)\ w_k & i \neq j \\ q_i & i = j \end{cases} \qquad (2)$$

$$q_i = \sum_{T_k \in E(M_i)} w_k \qquad (3)$$

The sojourn time is the time spent by the PN system in a given marking M.

We can observe that the average sojourn time in marking M_i is given by the following expression:

$$SJ_i = \frac{1}{q_i}$$

Let *RS*, *TRS* and *VRS* indicate the reachability set, tangible reachability set and vanishing reachability set of the stochastic process the following relation is true among these sets:

$$RS = TRS \cup VRS \text{ and } VRS \cap TRS = \varphi \qquad (5)$$

By ordering the markings so that the vanishing ones correspond to the first entries of the matrix and the tangible ones to the last, the transition probability matrix U can be decomposed in the following manner:

$$U = A + B = \begin{bmatrix} C & D \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ E & F \end{bmatrix} \qquad (6)$$

ANALYZING THE DERIVED CTMC

The solution of the system of linear matrix equations

$$\begin{cases} \psi = \psi U \\ \psi 1^T = 1 \end{cases} \qquad (7)$$

In which ψ is a row vector representing the steady-state probability distribution of the EMC, can be interpreted in terms of numbers of state-transitions performed by the EMC. Indeed, $1/\psi_i$ is the mean recurrence time for state s_i (marking M_i) measured in number of transition firings.

Although this method is computationally acceptable when the number of vanishing states are small (compared with the number of tangible states) but it also computes the probability of vanishing markings that do not increase the information content of the final solution since the time spent in these markings is

known to be null. Moreover, vanishing markings, by enlarging the size of the transition probability matrix *U*, tend to make the computation of the solution more expensive and in some cases even impossible to obtain. So the model must be reduced by computing the total transition probabilities among tangible markings only, thus identifying a Reduced EMC (REMC). The transition probability matrix of the REMC can thus be expressed as:

$$U' = F + EH \tag{8}$$

Where

$$H = \begin{cases} \left(\sum_{k=0}^{n_0} C^k\right) D \\ \begin{bmatrix} I & C \end{bmatrix}^1 D \end{cases} \tag{9}$$

The solution of the problem

$$\begin{cases} \psi' = \psi' U' \\ \psi' 1^T = 1 \end{cases} \tag{10}$$

Gives ψ that is a row vector representing the steady-state probability distribution of the REMC. The infinitesimal generator Q' of the CTMC associated with a GSPN can be constructed from the transition probability rate matrix U' of the REMC by dividing each of its rows by the mean sojourn time ($1/q_i$) of the corresponding tangible marking. To conform to the standard definition of the infinitesimal generators, the diagonal elements of Q' are set equal to the negative sum of the off diagonal components:

$$q_{ij}' = \begin{cases} \frac{1}{SJ_i} u_{ij}' & i \neq j \\ \sum_{j=1} q_{ij}' & i = j \end{cases} \tag{11}$$

An alternative way of computing the steady-state probability distribution over the tangible markings is thus that of solving the following system of linear matrix equations:

$$\begin{cases} \eta Q' = 0 \\ \eta 1^T = 1 \end{cases} \tag{12}$$

The probability that a given transition $T_k \in E(M_i)$ fires first in marking M_i has the expression:

$$P\{T_k \mid M_i\} = W_k / q_i' \tag{13}$$

Using the same argument, it can be observed that the average sojourn time in marking M_i is given by the following expression:

$$SJ_i = 1/q_i' \tag{14}$$

The steady-state distribution η' is the basis for a quantitative evaluation of the behaviour of the SPN that is expressed in terms of performance indices. These results can be computed using a unifying approach in which proper index functions (also called reward functions) are defined over the markings of the SPN and an average reward is derived using the steady-state probability distribution of the SPN. Assuming that r (M) represents one of such reward functions, the average reward can be computed using the following weighted sum:

$$R = \sum_{M_i \in RS(M_0)} r(M_i)\eta_i \tag{15}$$

Different interpretations of the reward function can be used to compute different performance indices. In particular, the following quantities can be computed using this approach:

The probability of a particular condition of the GSPN: Assuming that condition *Y (M)* is true only in certain markings of the PN, the following reward function can be defined [3]:

$$r(M) = \begin{cases} 1 & Y(M) = \text{true} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

The desired probability *P{Y}* is then computed using equation. The same result can also be expressed as:

$$P\{Y\} = \sum_{M_i \in A} \eta_i' \tag{17}$$

Where, $A = \{M_i \in RS(M_0): Y(M_i) = true\}$.

The expected value of the number of tokens in a given place: In this case, the reward function *r (M)* is simply the value of the marking of that place (say place *j*):

$$r(M) = n \quad \text{if } M(p_j) = n \tag{18}$$

Again, this is equivalent to identifying the subset *A (j, n)* of *RS (M_0)* for which the number of

tokens in place p_j is n $(A(j, n) = \{M_i \mid RS(M_0) \in M_i(p_j)=n\})$ the expected value of the number of tokens in p_j is given by:

$$E\left[M\left(p_j\right)\right] = \sum n \quad 0\left[n \quad P\{A(j,n)\}\right] \qquad (19)$$

Where, the sum is obviously limited to n values of $n \leq k$, if the place is k bounded.

The mean number of firings per unit of time of a given transition: Assume that the firing frequency of transition T_j (the throughput of T_j) was wanted to compute; observing that a transition may fire only when it is enabled, the reward function is assumed the value w_j in every marking that enables T_j:

$$r(M) = \begin{cases} w_j & T_j \in E(M) \\ 0 & otherwise \end{cases} \qquad (20)$$

The same quantity can also be computed using the more traditional approach of identifying the subset A_j of $RS(M_0)$ in which a given transition T_j is enabled $(A_j = \{M_i \in RS(M_0): T_j \in E(M_i)\})$. The mean number of firings of T_j per unit of time is then given by:

$$f_i = \sum_{M_i \in A_j} w_j \eta_i \qquad (21)$$

These results show that indeed, Petri nets can be used not only as a formalism for describing the behavior of distributed/parallel systems and for assessing their qualitative properties, but also as a tool for computing performance indices that allow the efficiency of these systems to be evaluated. As these basic parameters are computed, some more meaningful information can be derived. For example, a metric for comparing the security of different architectures can be gained by using the equation:

$$Security_{Net} = \frac{\left(\frac{\left(\sum_{t \in T} S_t \quad f_t \right)}{\sum_{t \in T} f_t} + \frac{\left(\sum_{p \in P} S_p \quad T_p \right)}{\sum_{p \in P} SJ_p} \right)}{2} \qquad (22)$$

Where S_t is the data security factor associated to the transition t, f_t is the firing rate of t, S_p is the data security factor associated to the place p, T_p is the expected time in which there is a token in place p. This is similar to the authors' previous work using simulation. Identically the reliability can be computed, but because the reliability is usually related to the processes of system, the reliability factor is just usually associated to the transitions than the places:

$$\text{Re}liability_{Net} = \frac{\left(\sum_{t \in T} RL_t \quad f_t \right)}{\sum_{t \in T} f_t} \qquad (23)$$

Where RL_t, stands for the reliability of process t.

We can compute some other parameters like those computed above for example we could gain a metric for comparing the availability of different architectures but because availability is usually related to the places of the system. Thus, the availability factor is usually associated to the places than transitions. We can gain it by the equation:

$$Avalibility_{Net} = \frac{\left(\sum_{p \in P} A_p \quad T_p \right)}{\sum_{p \in P} SJ} \qquad (24)$$

Which A_p is the availability associated to the transition t, t_p is the expected time in which there is a token in place P.

This is similar to our previous work using simulation. Another parameter that we can gain a metric is performance efficiency because performance efficiency is usually related to the transitions of the system it is associated to the transitions than places and we can gain it by the equation:

$$Performance\ Efficiency_{Net} = \frac{\left(\sum_{t \in T} P_t \quad f_t \right)}{\sum_{t \in T} f_t} \qquad (25)$$

Where P_t is the performance efficiency that associated to the transition t and f_t is the firing rate of t.

CASE STUDY

We have designed and implemented a case tool "*NFPG*".NFPG has the ability of drawing any kind of GSPN and driving the CTMC from the reachability graph, which is obtained from drawing GSPN and finally calculation the parameters that are mentioned above.

We have evaluated a real guide robot *'Jinny'* by NFPG as a case study. This case study proposes a selection framework of multiple navigation primitives

Table 1: Description of two navigations

Type	AutoMove	Contour tracking
Algorithm	Shortest path planning with obstacle avoidance	A (left, right, center) wall following technique using only laser scan data
Merits	Optimality (shortest path to any points on the maps) Generality (applicable in any situations)	Reactive Rise localization reliability Less affected by localization accuracy
Desirable environment	Generally applicable, but the performance drops in a narrow or crowded region.	An area where there are many static feature like walls or exhibits

Table 2: Description of the places and the transition

Place	Description	
$P_0(P_5)$	Navigation available (Completion)	
$P_1(P_2)$	Running *AutoMove* (*Contour tracking*)	
$P_3(P_4)$	Localization *Success* (*Warning*)	
Transition	Description	Firing rate
t_0	Start *AutoMove* (prob. *p*)	-
t_1	Start *Contour tracking* (prob. 1-*p*)	-
$t_2(t_4)$	Convert to *Contour tracking* (*AutoMove*) due to performance estimation	$\lambda_1(\lambda_2)$
t_3	Convert to *Contour tracking* due to localization *Warning*	-
$t_5(t_6)$	Localization *Warning* (*Success*) event fired	$\lambda_3(\lambda_4)$
$t_7(t_8)$	*AutoMove* (*Contour tracking*) completed	$\lambda_5(\lambda_6)$
t_9	Initialization	Λ_7

for a service robot using *Generalized Stochastic Petri Nets* (GSPN's). Jinny was developed by using a Petri net (PN) based control architecture, which was designed for multifunctional service robots.

Through their experiences, they concluded it is important for the robot adaptively to select its navigation primitives according to the conditions of environments. For example, in general cases, it is advisable that the robot uses a map-based navigation [23].

In general, navigation task is accomplished by the cooperation of several components such as a localizer and a path planner. As the related components and navigation primitives increase, it becomes troublesome to manage the relationships between them. A major scope of this paper is to propose a selection framework of multiple navigation primitives for a service robot.

In this approach, modeling, analysis and performance evaluation are carried out based on the *Generalized Stochastic Petri Nets* (GSPN's).Owing to the formalism; the strategy has following three major advantages. First, the framework is developed on firm mathematical foundation. This advantage makes it possible to set up state equations and other mathematical models governing the behaviors of a system. Second, the method supports modular and incremental designs of navigation framework since GSPN's have powerful modeling ability. It can model concurrency, asynchronous events, logical priority relations and structural interactions. In addition, several free or commercialized tools can automate the transformation from GSPN model to the mathematical representation. Third, as a graphical tool, GSPN's can represent both static and dynamic aspects of a system.

In this case, study, it is considered two types of navigation primitives, *AutoMove* and *Contour tracking*. The detailed description of these motions is summarized in Table 1.

From this observation, one rule is made for the primitive selection. It is that if the localizer falls into the *Warning* state, *Contour tracking* is unconditionally selected. The criterion of this selection problem is "which primitive leads the robot to a goal faster than the other with guaranteeing localization safety."

The modeling method goes through following procedure. First, based on a given system description, navigation primitives and required components are identified. Primitives are designed as places and the changes between them are modeled as transitions. Each component is represented as an independent GSPN's model. You can see the resultant GSPN model, which is drown in NFPG environment in Fig. 1.

Table 2 describes the physical meaning of places and transitions of the model. The GSPN model has six

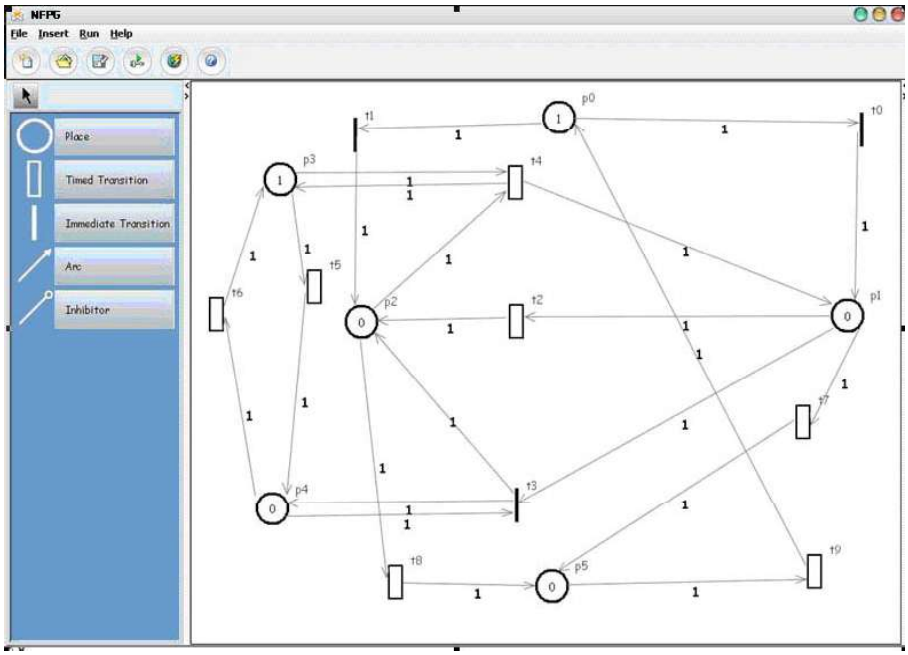

Fig. 1: Resultant GSPN model which is drown in NFPG environment

Table 3: Description of reachability set of the GSPN

Reachability set of the GSPN						
M_0	P_0 +			P_3		
M_1		P_1 +		P_3		
M_2			P_2 +	P_3		
M_3				P_3 +		P_5
M_4		P_1 +			P_4	
M_5					P_4 +	P_5
M_6			P_2 +		P_4	
M_7	P_0 +				P_4	

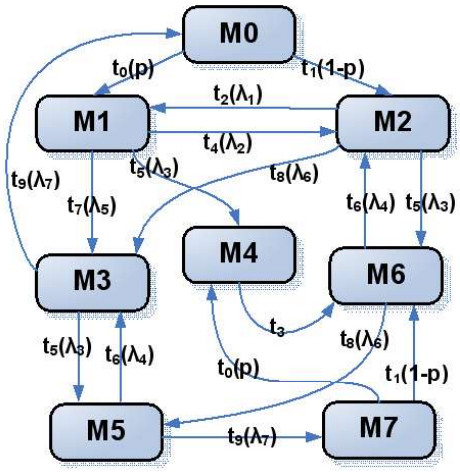

Fig. 2: The reachability graph of the GSPN system

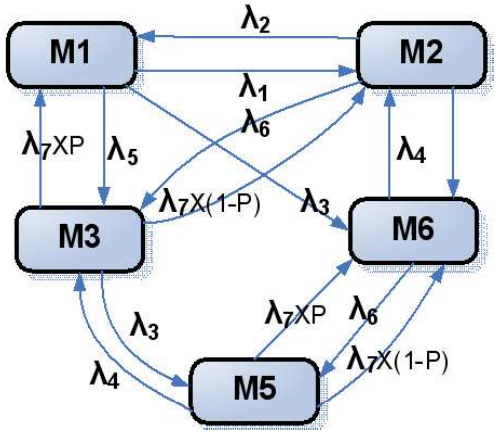

Fig. 3: The state transition rate diagram of the Markov chain associated with the GSPN

places, seven timed transitions (drawn as white bars) and three immediate transitions (drawn as black bars).

The initial marking is M_0= (1 0 0 1 0 0), which is denoted as P_0P_3 in the reachability graph in Fig. 2 by specifying the places having tokens.

Figure 3 shows the state transition rate diagram of the Markov Chain

The localizer has two internal states, *Success* and *Warning*. In the initial marking, a token is assigned to

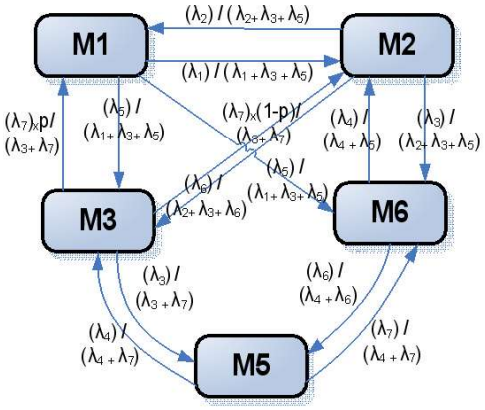

Fig. 4: Reduced embedded Markov chain

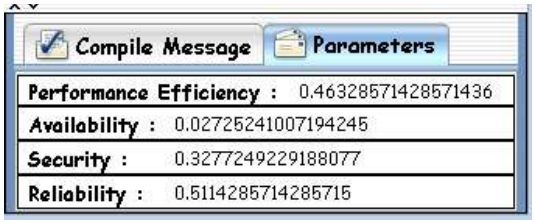

Fig. 5: Calculated availability, performance efficiency, security and reliability

P_3, i.e., it is assumed that the localizer initially knows its position. The *Warning* event t_5 fires when the localizer fails in estimating robot's accurate position for several steps. Two navigation primitives, *AutoMove* and *Contour tracking*, are modeled as P_1, P_2, respectively. Initially, the robot selects its motion by a random switch comprising the transitions t_0 and t_1 with corresponding probabilities p and 1-p, respectively. The transition between them takes place according to the change of localizer states. The immediate transition t_3 means that the robot takes *Contour tracking* as soon as the localizer *Warning* event fires does. The other transition between two primitives, t_2 and t_4, are modeled as timed transitions in order to express that the robot can change its current navigation primitive during the localizer *Success* state, if necessary. One of the most important modeling issues is how to set the firing rates $\Lambda = \{\lambda_1,\ldots,\lambda 7\}$. In order to perform the evaluation of GSPN designs, it is necessary to obtain an embedded Markov chain (EMC).

Figure 4 shows the EMC induced from the rechability graph of Fig. 2, which is derived from GSPN model of Fig. 1 and Ap, tp, f_t and P_t are as follow:

$$Q=\begin{bmatrix} -\lambda_1-\lambda_3-\lambda_5 & \lambda_1 & \lambda_5 & 0 & \lambda_3 \\ \lambda_2 & -\lambda_2-\lambda_3-\lambda_6 & \lambda_6 & 0 & \lambda_3 \\ \lambda_7.P & \lambda_7.(1-P) & -\lambda_7-\lambda_3 & \lambda_3 & 0 \\ 0 & 0 & \lambda_4 & -\lambda_4-\lambda_7 & \lambda_7 \\ 0 & \lambda_4 & 0 & \lambda_6 & -\lambda_4-\lambda_6 \end{bmatrix}$$

Fig. 6: The infinitesimal generator of the Markov chain

$$U'=\begin{bmatrix} 0 & \frac{\lambda_1}{(\lambda_1+\lambda_3+\lambda_5)} & \frac{\lambda_5}{(\lambda_1+\lambda_3+\lambda_5)} & 0 & \frac{\lambda_3}{(\lambda_1+\lambda_3+\lambda_5)} \\ \frac{\lambda_2}{(\lambda_2+\lambda_3+\lambda_6)} & 0 & \frac{\lambda_6}{(\lambda_2+\lambda_3+\lambda_6)} & 0 & \frac{\lambda_3}{(\lambda_2+\lambda_3+\lambda_6)} \\ \frac{\lambda_7.P}{(\lambda_3+\lambda_7)} & \frac{\lambda_7.(1-P)}{(\lambda_3+\lambda_7)} & 0 & \frac{\lambda_3}{(\lambda_3+\lambda_7)} & 0 \\ 0 & 0 & \frac{\lambda_4}{(\lambda_4+\lambda_7)} & 0 & \frac{\lambda_7}{(\lambda_4+\lambda_7)} \\ 0 & \frac{\lambda_4}{(\lambda_4+\lambda_6)} & 0 & \frac{\lambda_6}{(\lambda_4+\lambda_6)} & 0 \end{bmatrix}$$

Fig. 7: The reduced embedded Markov chain matrixin

$f_t = \{f_{t0},\ldots,f_{t9}\} = \{0, 0, 0.001, 0, 1000, 0.0117, 0.0166, 0.0142, 0.0071, 1000\}$

$P_t = \{P_{t0},\ldots,P_{t9}\} = \{0.008, 0.018, 0.458, 0.489, 0.599, 0.435, 0.685, 0.752, 0.833, 0.501\}$

$A_p = \{A_{p0}, Ap5\} = \{0.136, 0.454, 0.496, 0.355, 0.272, 0.145\}$

$t_p = \{t_{p0},\ldots,t_{p5}\} = \{0.004, 0.116, 0.108, 0.135, 0.127, 0.005\}$

$S_p = \{S_{p0},\ldots,S_{p5}\} = \{0.3, 0.7, 0.4, 0.5, 0.5, 0.2\}$

$S_t = \{S_{t0},\ldots,S_{t9}\} = \{0.53, 0.55, 0.5, 0.42, 0.56, 0.72, 0.68, 0.82, 0.83, 0.63\}$

$RL_t = \{RL_{t0},\ldots,RL_{t9}\} = \{0.35, 0.33, 0.31, 0.52, 0.27, 0.7, 0.63, 0.71, 0.73, 0.52\}$

Using Equations (22), (23), (24) and (25), it is obtained: Security = 0.327, Reliability = 0.511, Availability = 0.027 and Performance Efficiency = 0.463, which is shown in Fig. 5.

The Infinitesimal Generator of the Markov Chain is shown in Fig. 6. The Reduced Embedded Markov Chain Matrix is shown in Fig. 7.

CONCLUSION AND FUTURE WORKS

In this paper, we presented a method to derive non-functional parameters from Generalized Stochastic Petri Net. These parameters can be a good guidance for selecting sufficient software model between recommended software models, to achieve a model with a high security, reliability, efficiency and availability. We use GSPN because it is a formal model and there are many methods for transforming UML

(which is using for modeling the system widely). There are some key activities to achieve this goal: Driving the CTMC from GSPN; which is described in this paper extensively, analyzing the CTMC then obtaining the non-functional parameters. Finally, by designing and implementing NFPG we could calculate some of these parameters we used NFPG to calculate the parameters of a guide robot. One area for future research is to obtain other parameters [22-24].

REFERENCES

1. Robert, G., I.V. Pettit and Hassan Gomaa, 2000. Validation of Dynamic Behavior in UML Using Colored Petri Nets. UML 2000 Dynamic Behavior Workshop, York, England.
2. Merseguer, J., J.P. L´opezGrao and J. Campos, 2004. From UML Activity Diagrams to Stochastic Petri Nets: Application to Software Performance Engineering. ACM, WOSP 04.
3. Ajmone Marsan, M., 1995. Modeling with Generalized Stochastic Petri Nets. John Wiley Series in Parallel Computing-Chichester.
4. Motameni, H., A. Movaghar and M. Mozafari, 2005. Evaluating UML State Diagrams Using Colored Petri Net. Proc. of SYNASC'05, Romania.
5. Trowittzsch, A., Zimmermann and G. Hommel, 2005. Toward Quantitative Analysis of Real-Time UML using Stochastic Petri Nets. IPDPS.
6. Bernardi, S., S. Donatelli and J. Merseguer, 2002. From UML Sequence Diagrams and State Charts to Analysable Petri Net Models. ACM Proc. Int'l Workshop Software and Performance, pp: 35-45.
7. Merseguer, J., J. Campos and E. Mena, 2001. Performance Analysis of Internet Based Software Retrieval Systems using Petri Nets. ACM.
8. King, P. and R. Pooley, 1999. Using UML to derive Stochastic Petri Net Models. UKPEW.
9. Merseguer, J., S. Bernardi, J. Campos and S. Donatelli, 2002. A Compositional Semantics for UML State Machines Aimed at Performance Evaluation. Silva, M., A. Giua and J.M. Colom (Eds.). Proc. of the 6th Int. Workshop on Discrete Event Systems (WODES'02), Zaragoza, Spain, pp: 295-302.
10. Elkoutbi, M. and R.K. Keller, 1998. Modeling Interactive Systems with Hierarchical Colored Petri Nets. Advanced Simulation Technologies Conf., Boston, MA, pp: 432-437.
11. Eshuis, R., 2002. Semantics and Verification of UML Activity Diagrams for Workflow Modeling. Ph.D. Thesis, University of Twente.
12. Fukuzawa, K. and Saeki, 2002. Evaluating Software Architecture by Colored Petri Net. Dept. of Computer Science, Tokyo Institute of Technology, Okayama 2-12-1, Meguro-uk, Tokyo, Japan, pp: 152-8552.
13. Pettit, R.G. and H. Gomaa, 2000 Validation of Dynamic Behavior in UML Using Colored Petri Nets. UML'00.
14. Shin, M., A. Levis and L. Wagenhals, 2003. Transformation of UML-Based System Model into CPN Model for Validating System Behavior. Proc. of Compositional Verification of UML Models, Workshop of the UML'03 Conference, California, USA.
15. Faul, M.B., Verifiable Modeling Techniques Using a Colored Petri Net Graphical Language. Technology Review Journal, Spring/Summer.
16. Motameni, H., A. Movaghar and B. Kardel, 2005. Verifying and Evaluating UML Activity Diagram by Converting to CPN. Proc. of SYNASC'05.
17. Motameni, H., M. Zandakbari and A. Movaghar, 2006. Deriving Performance Parameters from the Activity Diagram Using GSPN and Markov. ICCSA 2006 Proceedings of 4th International Conference on Computer Science and Its Applications, San Ddiego, California.
18. Motameni, H., H. Montazeri, M. Siasifar, A. Movaghar and M. Zandakbari, 2006. Mapping State Diagram To Petri Net: An Approach To Use Markov Theory For Analyzing Non-Functional Parameters. CISSE'06 Proceedings of 2nd IEEE International Conferences on Computer, Information and Systems Sciences and Engineering, Bridgeport, USA.
19. Motameni, H., H. Montazeri, M. Siasifar, A. Movaghar and M. Zandakbari, 2006. Using Markov Theory for Deriving Non-Functional Parameters on Transformed Petri Net from State Diagram. SEC(R) 2006 Proceedings of International Conference on software engineering conference (Russia), Moscow, Russia.
20. Rana, O.F. and M.S. Shields, 2000. Performance Analysis of Java Using Petri Nets. LNCS 1823: High Performance Computing and Networking, pp: 657-667. 8th International Conference, HPCN Europe, 2000, Amsterdam, The Netherlands, May 2000, Proceedings/M. Bubak, H. Afsarmanesh, R. Williams, B. Hertzberger (Eds.), Springer Verlag.
21. Object Management Group, UML™ Profile for Schedulability, Performance and Time Specification, OMG document, Version 1.1, January 2005.

22. Kim, G., W. Chung and M. Kim, 2005. A Selection Framework of Multiple Navigation Primitives Using Generalized Stochastic Petri Nets. Poceedings of the IEEE International Conference on Robotics and Automation (ICRA), Barcelona, Spain.
23. Nihal, Y. Ö., 2007. On the Numbers of the Form n = x2 + Ny2, World Applied Sciences Journal 2 (1): 45-48.
24. Erçetin, S. S., Çetin, B. and N. Potas, 2007 , Multi-Dimensional Organizational Intelligence Scale (Muldimorins) , World Applied Sciences Journal 2 (3): 151-157.