MAPPING ACTIVITY DIAGRAM TO PETRI NET: APPLICATION OF MARKOV THEORY FOR ANALYZING NON-FUNCTIONAL PARAMETERS

H. Motameni*

Department of Computer Engineering, Islamic Azad University
Science and Research Branch, Tehran, Iran
motameni@iausari.ac.ir

A. Movaghar

Department of Computer Engineering, Sharif University of Technology
Tehran, Iran
movaghar@sharif.edu

M. Fadavi Amiri

Department of Computer Engineering, Iran University of Science and Technology
Tehran, Iran
fadavi@comp.iust.ac.ir

*Corresponding Author

(**Received: January 17, 2006 – Accepted in Revised Form: March 18, 2007)**

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 underlying risks. 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 behavior 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.

Keywords UML, Activity Diagram (AD), Generalized Stochastic Petri Net (GSPN), Continuous Time Markov Chain (CTMC), Non-Functional Parameters, Markov Reward Models

چکیده کیفیت طراحی معماری یک سیستم نرم افزاری تاثیر زیادی در بدست آوردن نیازهای غیرعملیاتی دارد. یک پروژه نرم افزاری در حال توسعه اغلب به وسیله فاکتورهای غیر عملیاتی مانند انتظارات مشتری درباره کارایی و قابلیت اعتماد تحت تاثیر قرار گرفته، ریسک های اساسی آن کاهش می‌یابد. ارزیابی پارامترهای غیر عملیاتی یک سیستم نرم افزاری در مراحل اولیه طراحی و فرایند توسعه آن، اغلب به صورت فاکتورهای اساسی این بحث در نظر گرفته می‌شوند و این ارزیابی ها می‌توانند در انتخاب امن ترین و مطمئن ترین مدل کمک کنند. در این مقاله، روشی برای به دست آوردن این پارامتر ها از "شبکه پتری تصادفی عمومی" (GSPN) معرفی می‌شود تا بتوان رفتار تصادفی سیستم را تحلیل نمود. سپس زنجیره مارکوف زمان پیوسته از GSPN مشتق شده و تئوری مارکوف برای به دست آوردن این پارامترها مورد استفاده قرار می‌گیرد.

1. 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.

There are three general characteristics of PNs that make them interesting in capturing concurrent, object-oriented behavioral 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].

We present a method for obtaining non-functional parameters from GSPN. The reason for using GSPN is that, there are some methods for transforming UML diagrams to GSPN as an example one of them is introduced in [2,3].

Currently the Unified Model Language (UML) diagrams are widely used in the field of software design as it is easy to use in comparison to other alternatives, and is powerful in describing different aspects of a system. However, the semi-formal properties 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 seem to be more straightforward compared to its alternatives, which are analytic methods.

First a brief discussion on GSPN is presented, then UML will be disscussed to introduce its fundamentals and history.

For the next step transforming AD to LGSPN(GSPN) is discussed and then how a CTMC is derived from GSPN. Finally, performance evaluation on the derived CTMC is conducted and a case study is explained.

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 compared to simulation techniques. Therefore, analytic methods remain as the only choice for evaluating critical systems, but we should consider that this method is more useful in small systems because it is possible to have more details

2. RELATED WORK

The Use of Stochastic Petri Net (SPN) and its extensions have been discussed in several papers [2,5,6,7 and 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 Colored 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 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 discussed transforming the software model to analytical models or evaluating the performance model. In other words, none of them provides an integrated method, which can start from software models 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, the performance model is evaluted in a way that leads to gain meaningful parameters of the system like reliability and security.

3. REVIEWING THE GSPN AND UML

3.1. GSPN The basic PN model includes two components: places and transitions connected together via arcs to model system behavior; however, it may be extended by introducing the notion of time, leading to timed Petri nets (TPN) for a performance analysis of Petri Nets quantitative analysis. In TPN an exact time is associated to each transition. A timed PN is called a SPN, when random variables are used in specifying the time behavior. 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 $t \in T_T$ are 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, in 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 and has zero 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 $E = \{W_1, W_2, ..., W_{(m)}\}$. Furthermore, if a set of random switches regulating the net behavior are marking-dependent, they are characterized as dynamic; otherwise, they are static [21].

3.2. UML UML consists of a set of graphs or charts with explanatory comments that can be expressed either in a formal way or in natural language. Each diagram has a specific and precise position in the design process. An activity diagram is a dynamic diagram that shows the activity and the event that causes 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 activities or processes. The biggest disadvantage of activity diagrams is that they do not clearly explain which objects execute which activities, and the way that the contection works between them. However, labeling 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].

3.2.1. Annotating AD Additional information is needed to transform the AD to GSPN, which includes time information and priorities of conflict sets. This information is provided by the notation. The method used in this paper is identical to the method introduced by Merseguer et al recommending two different aspects in the annotations: time and probability [2]. The method uses tagged values as an extensibility mechanism to integrate them in the UML models. Annotations will be attached to both transitions and states.

In this paper the suggested format is {n sec.; P(k)} or {n-m sec.; P(k)} for timed transitions and {P(k)} for untimed transitions. If no probability (P(k)) is provided, it is assumed identical probability for each transition in the same Enabling Conflict Set (ECS). The other parameters needed for evaluation could be attached to the above notation with separated tagged values or in the form of constraints.

4. TRANSFORMING AD TO LGSPN

The transformation algorithm used to translate the activity diagram to the GSPN model is the one that is explained by Merseguer et al [2]. As long as the provided formalism seems to be well formed and well described, it was preferred to be used in this research to define an alternative. The only change made to the use of the algorithm is to relate the ratios (like security, dependability etc) assigned to the UML AD to GSPN elements. These ratios are then included in the LGSPN together with firing rates of transitions and the weights of immediate transitions.

The parameters like security ratio and reliability ratio of the action are just attached to those AD transitions that represent an action. These parameters are then simply related to time transitions of the LGSPN, which the tags are attached to. The parameters attached to the action states are attached to all of the places existing in the transformation of that element to LGSPN. Once the GSPN system is defined, some structural properties may be computed to perform a validation of the model. First, P and T semi-flows can be computed to check whether the net is structurally bounded and whether it may have home-states. Other structural results that may be computed are the Effective Conflict Sets (ECSs) of the model. These results ensure that the net is suitable for a numerical evaluation yielding the steady-state probabilities of all its markings [3].

5. DERIVING THE EMBEDDED CTMC

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 as disregarding the concept of time and focusing 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 (the term GSPN is used instead of LGSPN, as the labels provided by the LGSPN do not have any effect on analyzing of the net) 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 T_k and with $E_j(M_i) = \{h : T_h \in E(M_i) \wedge M_i \mid T_h > M_j\}$ 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:

$$U_{i,j} = \frac{\sum_{T_k \in E_j(M_i)} W_k}{q_i} \qquad (1)$$

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:

$$RS = TRS \cup VRS \text{ and } VRS \cap TRS = \phi \qquad (2)$$

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:[3]

$$U = A + B = \begin{bmatrix} C & D \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ E & F \end{bmatrix} \qquad (3)$$

6. ANALYZING THE DERIVED CTMC

The solution of the system of linear matrix equations

$$\begin{cases} \psi = \psi U \\ \psi 1^T = 1 \end{cases} \quad (4)$$

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 does not increase the information content of the final solution since the time spent in these markings is known to be null. Moreover, vanishing markings, created 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:[3]

$$U' = F + E H \quad (5)$$

Where

$$H = \begin{cases} (\Sigma_{k0=0}^{n} C^k) D \\ [I - C]^{-1} D \end{cases} \quad (6)$$

The solution of the problem

$$\begin{cases} \psi' = \psi' U' \\ \psi' 1^T = 1 \end{cases} \quad (7)$$

gives ψ 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 ($1/u_i$) of the corresponding tangible marking (The sojourn time is the time spent by the PN system in a given marking M). 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:

$$q'_{ij} = \begin{cases} \frac{1}{SJ_i} u'_{ij} & i \neq j \\ -\sum_{j \neq i} q'_{ij} & i = j \end{cases} \quad (8)$$

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:

$$\begin{cases} \eta Q' = 0 \\ \eta 1^T = 1 \end{cases} \quad (9)$$

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

$$P\{T_k | M_i\} = W_k / q'_i \quad (10)$$

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

$$SJ_i = 1/q'_i \quad (11)$$

The steady-state distribution η' is the basis for a quantitative evaluation of the behavior 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:

$$R = \sum_{M_i \in RS(M_0)} r(M_i) \eta_i \quad (12)$$

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. Reward Function can be defined as follows [3]:

$$r(M)=\begin{cases}1 & Y(M)=\text{true}\\ 0 & \text{otherwise}\end{cases} \quad (13)$$

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

$$P\{Y\}=\sum_{M_i\in A}\eta'_i \quad (14)$$

where

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

- 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):

$$r(M)=n \text{ if } M(P_i)=n \quad (15)$$

Again, this is an equivalent to identify 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:

$$E[M(p_j)]=\sum_{n>0}[n\ P\{A(j,n)\}] \quad (16)$$

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

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

$$r(M)=\begin{cases}W_j & T_j\in E(M)\\ 0 & \text{otherwise}\end{cases} \quad (17)$$

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:[3]

$$f_i=\sum_{M_i\in A_j}W_j\ \eta_i \quad (18)$$

As we know, Petri nets are not only used 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 formula comparing the security of different architectures can be gained by using the equation:

$$\text{Security}_{\text{Net.}} = ((\Sigma_{t\in T}S_t * f_t)/\Sigma_{t\in T}f_t + (\Sigma_{p\in P}S_p * T_p)/\Sigma_{p\in P}SJ_p))/2 \quad (19)$$

Where S_t is the data security factor associated to the transition t, f_t is the firing rate of t, S_p 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 the system, the reliability factor is just usually associated to the transitions than the places:

$$\text{Reliability}_{\text{Net.}} = \left(\Sigma_{t\in T}RL_t * f_t)/\Sigma_{t\in T}f_t\right) \quad (20)$$

where RL_t, stands for the reliability of process t.

7. CASE STUDY

Figure 1 shows the activity diagram of a parallel system. The system operations are modelled as follows. A set of new data is read (firing of transition t_{new}), and two processes are started in parallel with the same set of data (the fork operation-firing of t_{start}). When both processes are complete (firing of t_{par}, and t_{par1}, respectively), a synchronization takes place (the join operation-firing of transition t_{syn}). The consistency of the two results is then controlled, and one of the two transitions t_{OK} or t_{KO} fires, indicating whether the results are acceptable, or not. If the results are not consistent, the whole computation is repeated on the same data, after a further control (firing of

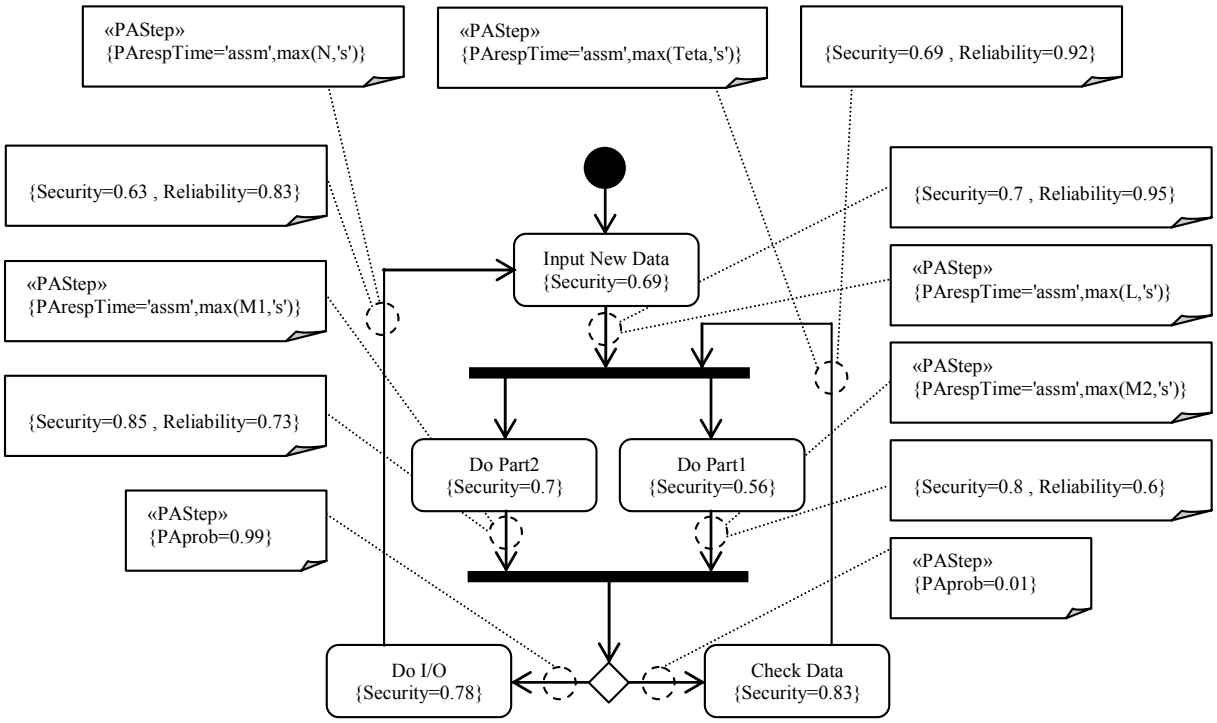

Figure 1. Activity diagram that specifies a parallel system.

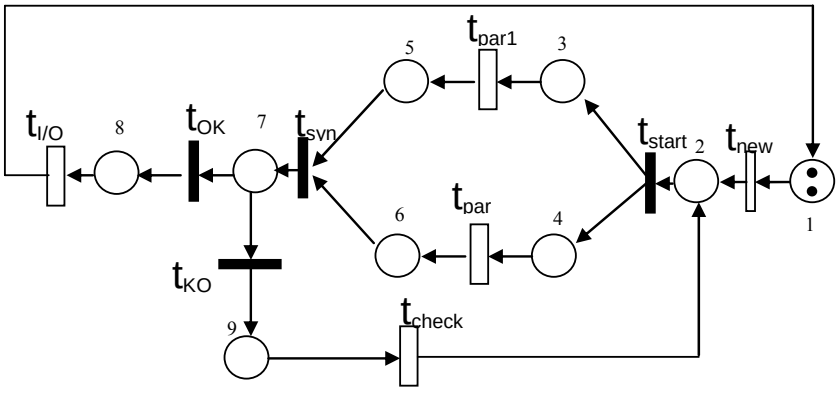

Figure 2. Conversion result in GSPN.

t_{check}); otherwise, the results are output (firing of transition $t_{I/O}$), and a new set of data is considered. The model is then converted to a GSPN model by the methodology [2]. The conversion result is shown in Figure 2. The model specifications are shown in Tables 1 and 2. The net has 38 different markings that could be separated into two subsets of 18 vanishing markings and 20 tangibles. All these markings are listed in Table 3. The four sub matrices C, D, E, and F of Equation 3 have dimensions (18×18), (18×20), (20×18), and (20×20), respectively. Because of the relatively

TABLE 1. Timed Transitions of Figure 2 and Their Specifications.

Transition	Rate	Semantics
$T_{newdata}$	λ	Infinite-server
T_{par1}	μ_1	Single-server
T_{par2}	μ_2	Single-server
$T_{I/O}$	ν	Single-server
T_{check}	θ	Single-server

TABLE 2. Immediate Transitions of Figure 2 and Their Specifications.

Transition	Weight	Priority	ECS
t_{start}	1	1	1
t_{syn}	1	1	2
t_{OK}	α	1	3
t_{KO}	β	1	3

TABLE 3. The Markings of the GSPN Presented in Figure 2.

M0 = 2p1	M1 = p1 + p2	M2 = p1 + p3 + p4	M3 = p2 + p3 + p4
M4 = 2p3 + 2p4	M5 = p1 + p4 + p5	M6 = p1 + p3 + p6	M7 = p3 + 2p4 + p5
M8 = 2p3 + p4 + p6	M9 = p2 + p4 + p5	M10 = P1+p5+p6	M11 = p1 + p7
M12 = p1 + p9	M13 = p1 + p8	M14 = p2 + p3 + p6	M15 = 2p4 + 2p5
M16 = p3 + p4 + p5 + p6	M17 = p3 + p4 + p7	M18 = p3 + p4 + p9	M19 = p3 + p4 + p8
M20 = 2p3 + 2p6	M21 = p2 + p9	M22 = p2 + p8	M23 = p4 + 2p5 + p6
M24 = p4 + p5 + p7	M25 = p3 + p5 + 2p6	M26 = p4 + p5 + p8	M27 = p3 + p6 + p9
M28 = p3 + p6 + p8	M29 = p3 + p5 + 2p6	M30 = p3 + p6 + p7	M31 = p5 + p6 + p7
M32 = p7 + p9	M33 = 2p9	M34 = p8 + p9	M35 = p5 + p6 + p8
M36 = p7 + p8	M37 = 2p8		

TABLE 4. Non-Zero Components of Matrixes C, D, E and F.

Component	Probability	Component	Probability
$c_{4,5}(10,11)$	1.0	$d_{1,2}(1,2)$	1.0
$c_{7,8}(16,17)$	1.0	$d_{2,3}(3,4)$	1.0
$c_{11,12}(23,24)$	1.0	$d_{3,6}(9,7)$	1.0
$c_{13,14}(29,30)$	1.0	$d_{5,8}(11,12)$	*a*
$c_{15,16}(31,32)$	1.0	$d_{5,9}(11,13)$	*b*
$c_{17,18}(35,36)$	1.0	$d_{6,7}(14,8)$	1.0
-	-	$d_{8,11}(17,18)$	*a*
$e_{1,1}(0,1)$	1.0	$d_{8,12}(17,19)$	*b*
$e_{2,2}(2,3)$	$\lambda/(\lambda+\mu_1+\mu_2)$	$d_{9,11}(21,18)$	1.0
$e_{4,3}(5,9)$	$\lambda/(\lambda+\mu_2)$	$d_{10,12}(22,19)$	1.0
$e_{4,4}(5,10)$	$\mu_2/(\lambda+\mu_2)$	$d_{12,14}(24,25)$	*a*
$e_{5,4}(6,10)$	$\mu_1/(\lambda+\mu_1)$	$d_{12,15}(24,26)$	*b*
$e_{5,6}(6,14)$	$\lambda/(\lambda+\mu_1)$	$d_{14,16}(30,27)$	*a*
$e_{6,7}(7,16)$	$\mu_2/(\mu_1+\mu_2)$	$d_{14,17}(30,28)$	*b*
$e_{7,7}(8,16)$	$\mu_1/(\mu_1+\mu_2)$	$d_{16,18}(32,33)$	*a*
$e_{8,9}(12,21)$	$\lambda/(\lambda+\nu)$	$d_{16,19}(32,34)$	*b*
$e_{9,1}(13,1)$	$\theta/(\lambda+\theta)$	$d_{18,19}(36,34)$	*a*
$e_{9,10}(13,22)$	$\lambda/(\lambda+\theta)$	$d_{18,20}(36,37)$	*b*
$e_{10,11}(15,23)$	1.0	-	-
$e_{12,2}(19,3)$	$\theta/(\mu_1+\mu_2+\theta)$	$f_{2,4}(2,5)$	$\mu_1/(\lambda+\mu_1+\mu_2)$
$e_{13,13}(20,29)$	1.0	$f_{2,5}(2,6)$	$\mu_2/(\lambda+\mu_1+\mu_2)$
$e_{14,15}(25,31)$	$\mu_2/(\mu_2+\nu)$	$f_{3,6}(4,7)$	$\mu_1/(\mu_1+\mu_2)$
$e_{15,3}(26,9)$	$\theta/(\mu_2+\theta)$	$f_{3,7}(4,8)$	$\mu_2/(\mu_2+\mu_2)$
$e_{15,17}(26,35)$	$\mu_2/(\mu_2+\theta)$	$f_{6,10}(7,15)$	$\mu_1/(\mu_1+\mu_2)$
$e_{16,15}(27,31)$	$\mu_1/(\mu_1+\nu)$	$f_{7,13}(8,20)$	$\mu_2/(\mu_1+\mu_2)$
-	-	$f_{8,1}(12,0)$	$\nu/(\lambda+\nu)$
-	-	$f_{11,2}(18,2)$	$\nu/(\mu_1+\mu_2+\nu)$
-	-	$f_{11,14}(18,25)$	$\mu_1/(\mu_1+\mu_2+\nu)$
-	-	$f_{11,16}(18,27)$	$\mu_2/(\mu_1+\mu_2+\nu)$
-	-	$f_{12,15}(19,26)$	$\mu_1/(\mu_1+\mu_2+\theta)$
-	-	$f_{12,17}(19,28)$	$\mu_2/(\mu_1+\mu_2+\theta)$
-	-	$f_{14,4}(25,5)$	$\nu/(\mu_2+\nu)$
-	-	$f_{16,5}(27,6)$	$\nu/(\mu_1+\nu)$
-	-	$f_{18,8}(33,12)$	1.0
-	-	$f_{19,9}(34,13)$	$\nu/(\theta+\nu)$

TABLE 5. Non-zero Components of Matrix U'.

Component	Probability	Component	Probability
$u'_{1,2}(0,2)$	1.0	$u'_{11,2}(18,2)$	$\nu/(\mu_1+\mu_2+\nu)$
$u'_{2,3}(2,4)$	$\lambda/(\lambda+\mu_1+\mu_2)$	$u'_{11,14}(18,25)$	$\mu_1/(\mu_1+\mu_2+\nu)$
$u'_{2,4}(2,5)$	$\mu_1/(\lambda+\mu_1+\mu_2)$	$u'_{11,16}(18,27)$	$\mu_2/(\mu_1+\mu_2+\nu)$
$u'_{2,5}(2,6)$	$\mu_2/(\lambda+\mu_1+\mu_2)$	$u'_{12,3}(19,4)$	$\theta/(\mu_1+\mu_2+\theta)$
$u'_{3,6}(4,7)$	$\mu_1/(\mu_1+\mu_2)$	$u'_{12,15}(19,26)$	$\mu_1/(\mu_1+\mu_2+\theta)$
$u'_{3,7}(4,8)$	$\mu_2/(\mu_2+\mu_2)$	$u'_{12,17}(19,28)$	$\mu_2/(\mu_1+\mu_2+\theta)$
$u'_{4,6}(5,7)$	$\lambda/(\lambda+\mu_2)$	$u'_{13,16}(20,27)$	a
$u'_{4,8}(5,12)$	$\alpha\mu_2/(\lambda+\mu_2)$	$u'_{13,17}(20,28)$	b
$u'_{4,9}(5,13)$	$\beta\mu_2/(\lambda+\mu_2)$	$u'_{14,4}(25,5)$	$\nu/(\mu_2+\nu)$
$u'_{5,7}(6,8)$	$\lambda/(\lambda+\mu_1)$	$u'_{14,18}(25,33)$	$\alpha\mu_2/(\mu_2+\nu)$
$u'_{5,8}(6,12)$	$\alpha\mu_1/(\lambda+\mu_1)$	$u'_{14,19}(25,34)$	$\beta\mu_2/(\mu_2+\nu)$
$u'_{5,9}(6,13)$	$\beta\mu_1/(\lambda+\mu_1)$	$u'_{15,6}(26,7)$	$\theta/(\mu_2+\theta)$
$u'_{6,10}(7,15)$	$\mu_1/(\mu_1+\mu_2)$	$u'_{15,19}(26,34)$	$\alpha\mu_2/(\mu_2+\theta)$
$u'_{6,11}(7,18)$	$\alpha\mu_2/(\mu_1+\mu_2)$	$u'_{15,20}(26,37)$	$\beta\mu_2/(\mu_2+\theta)$
$u'_{6,12}(7,19)$	$\beta\mu_2/(\mu_1+\mu_2)$	$u'_{16,5}(27,6)$	$\nu/(\mu_1+\nu)$
$u'_{7,11}(8,18)$	$\alpha\mu_1/(\mu_1+\mu_2)$	$u'_{16,18}(27,33)$	$\alpha\mu_1/(\mu_1+\nu)$
$u'_{7,12}(8,19)$	$\beta\mu_1/(\mu_1+\mu_2)$	$u'_{16,19}(27,34)$	$\beta\mu_1/(\mu_1+\nu)$
$u'_{7,13}(8,20)$	$\mu_2/(\mu_1+\mu_2)$	$u'_{17,7}(28,8)$	$\theta/(\mu_1+\theta)$
$u'_{8,1}(12,0)$	$\nu/(\lambda+\nu)$	$u'_{17,19}(28,34)$	$\alpha\mu_1/(\mu_1+\theta)$
$u'_{8,11}(12,18)$	$\lambda/(\lambda+\nu)$	$u'_{17,20}(28,37)$	$\beta\mu_1/(\mu_1+\theta)$
$u'_{9,2}(13,2)$	$\theta/(\lambda+\theta)$	$u'_{18,8}(33,12)$	1.0
$u'_{9,12}(13,19)$	$\lambda/(\lambda+\theta)$	$u'_{19,9}(34,13)$	$\nu/(\theta+\nu)$
$u'_{10,14}(15,25)$	a	$u'_{19,11}(34,18)$	$\theta/(\theta+\nu)$
$u'_{10,15}(15,26)$	b	$u'_{20,12}(37,19)$	1.0

simple structure of the model, all four blocks are quite sparse (few of the entries are non-zero). Table 4 reports the non-zero values of all these matrices, with the understanding that $x_{i,j}(r,s)$ represents the non-zero component of the X matrix located in row i and column j, and corresponding

to the transition probability from state r to state s.

Using the method outlined by Equation 5, the transition probability matrix of the REMC reported in Table 5 is obtained. Solving the system of linearEquation 4, the steady state probabilities are obtained for all the states of the REMC. Supposing that the transition weights assume the following values: $\lambda = 0.2$, $\mu_1 = 2.0$, $\mu_2 = 1.0$, $\theta = 0.1$, $\nu = 5.0$, $\alpha = 0.99$, and $\beta = 0.01$. Using Equations 19 and 20, it is obtained: Security = 0.7078 and Reliability = 0.678.

8. CONCLUSION AND FUTURE WORKS

In this paper, we presented a method to derive non-functional parameters from the Generalized Stochastic Petri Net. These parameters can be a good guidance for selecting sufficient software models between recommended software models, to achieve a model with a high security, reliability. We use GSPN because it's a formal model and there are many methods for transforming UML (which is widely used for modeling the system). There are some key activities to achieve this goal: Driving the CTMC from GSPN; which is extensively described in this paper by analyzing the CTMC then by obtaining the non-functional parameters. The work can be improved by integrating these steps in a CASE tool. It can also be expanded by mapping other UML diagrams to GSPN models.

9. REFERENCES

1. Robert, G,. Pettit, IV. and Gomaa, H., "Validation of Dynamic Behavior in UML Using Colored Petri Nets", UML, Dynamic Behavior Workshop, York, England, (October, 2000).
2. Merseguer, J., L´opezGrao, J. P. and Campos, J., "From UML Activity Diagrams to Stochastic Petri Nets: Application to Software Performance Engineering", ***ACM, WOSP 04***, California, (January, 2004).
3. Ajmone Marsan, M., "Modeling with Generalized Stochastic Petri Nets", John Wiley Series in Parallel Computing-Chichester, (1995).
4. Motameni, H., Movaghar, A. and Mozafari, M., "Evaluating UML State Diagrams Using Colored Petri Net", ***Proc. of SYNASC'05***, Romania, (2005).
5. Trowittzsch Zimmermann, A. and Hommel, G., "Toward Quantitative Analysis of Real-Time UML using Stochastic Petri Nets", ***IPDPS***, Colaorado, (2005).
6. S. Bernardi, S. Donatelli and J. Merseguer, "From UML Sequence Diagrams and State Charts to Analysable Petri Net Models", ***ACM Proc. Int'l Workshop Software and Performance***, (2002), 35-45.
7. Merseguer, J., Campos, J. and Mena, E., "Performance Analysis of Internet Based Software Retrieval Systems using Petri Nets", ***ACM***, Colaorado, (2001).
8. King, P. and Pooley, R., "Using UML to derive Stochastic Petri Net Models", ***UKPEW***, Bristol, (1999).
9. Merseguer, J., Bernardi, S., Campos, J. and Donatelli, S., "A Compositional Semantics for UML State Machines Aimed at Performance Evaluation", M. Silva, A. Giua and J. M. Colom (Eds.), ***Proc. of the 6th Int. Workshop on Discrete Event Systems (WODES'02)***, Zaragoza, Spain, (2002), 295-302.
10. Elkoutbi, M. and Keller, R. K., "Modeling Interactive Systems with Hierarchical Colored Petri Nets", ***Advanced Simulation Technologies Conf.***, Boston, MA, (1998), 432- 437.
11. Eshuis, R., "Semantics and Verification of UML Activity Diagrams for Workflow Modeling", Ph.D. Thesis, University of Twente, (2002).
12. Fukuzawa, K. and Saeki, "Evaluating Software Architecture by Colored Petri Net", Dept. of Computer Science, Tokyo Institute of Technology, Okayama 2-12-1, Meguro-uk, Tokyo, 152-8552, ***SDkE***, Japan, (2002).
13. Pettit, R. G. and Gomaa, H., "Validation of Dynamic Behavior in UML Using Colored Petri Nets", ***UML***, York, England, (2000).
14. Shin, M. E., Levis, A. H. and Wagenhals, L. W., "Transformation of UML-Based System Model into CPN Model for Validating System Behavior", ***Proc. Compositional Verification of UML Models, Workshop, Sixth International Conference on the UML***, San Francisco, CA, (October, 2003), 397-404.
15. Faul, M. B., "Verifiable Modeling Techniques Using a Colored Petri Net Graphical Language", ***Technology Review Journal***, Spring/Summer, (2004).
16. Motameni, H., Movaghar, A. and Kardel, B., "Verifying and Evaluating UML Activity Diagram by Converting to CPN", ***Proc. of SYNASC'05***, Romania, (2005).
17. Motameni, H., Zandakbari, M. and Movaghar, A., "Deriving Performance Parameters from the Activity Diagram Using GSPN and Markov Chains", ***ICCSA 2006 Proceedings of 4th International Conference on Computer Science and Its Applications***, San Ddiego, California, USA, (2006).
18. Motameni, H., Montazeri, H., Siasifar, M., Movaghar, A. and Zandakbari, M., "Mapping State Diagram To Petri Net : An Approach To Use Markov Theory For Analyzing Non-Functional Parameters", ***CISSE'06 Proceedings of 2th IEEE International Conferences on Computer, Information, and Systems Sciences, and Engineering***, Bridgeport, USA, (2006).
19. Motameni, H., Montazeri, H., Siasifar, M., Movaghar, A. and Zandakbari, M., "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, (2006).

20. Rana, O. F. and Shields, M. S., "Performance Analysis of Java Using Petri Nets", ***Proceedings of the 8th International Conference on High-Perfromance Computing and Networking***, (May, 2000), 657-667.
21. Choi, J. Y. and Reveliotis, S. A., "A Generalized Stochastic Petri Net Model for Performance Analysis and Control of Capacitated Re-entrant Lines", ***IEEE Transactions on R and A***, Vol. XX, No. Y, (2003).
22. Object Management Group, UML™ Profile for Schedulability, Performance, and Time Specification, OMG document, Version 1.1, (January, 2005).
23. Kim, G., Chung, W. and Kim, M., "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, (2005).