Contents lists available at GrowingScience

Management Science Letters

homepage: www.GrowingScience.com/msl

Modeling and evaluation of information systems using coloured petri network

Ehsan Zamirpour[a*], Ali Harounabadi[b] and Seyed Javad Mirabedini[c]

[a]Department of Computer Engineering, Dezful Branch, Islamic Azad University, Dezful, Iran
[b]M.Sc. Student, Department of Public Management, Science and Research branch, Islamic Azad University (IAU), Tehran, Iran
[c]Department of Computer Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran

CHRONICLE

Article history:
Received January 4, 2014
Accepted 1 June 2014
Available online
June 9 2014

Keywords:
Performance evaluation
Unified Modeling Language
Coloured Petri Net

ABSTRACT

Nowadays with the growth of organizations and their affiliates, the importance of information systems has increased. Functional and non-functional requirements of information systems in an organization are supported. There are literally several techniques to support the functional requirements in terms of software methodologies, but support for the second set of requirements has received little attention. Software Performance Engineering (SPE) forum tries to address this issue by presenting software methodologies to support both types of requirements. In this paper, we present a formal model for the evaluation of system performance based on a pragmatic model. Because of supporting the concurrency concepts, petri net has a higher priority than queuing system. For mapping UML to colored Petri net diagram, we use an intermediate graph. The preliminary results indicate that the proposed model may save significant amount of computations.

1. Introduction

Nowadays, the extent of information systems in different applications is undeniable. Many design problems are categorized into two groups of structure as well as object oriented techniques. Object-oriented technique plays essential role on software development. Unified modeling language (UML) as a modeling language supports object oriented concepts and it was introduced in 1999. Integration of symbols, techniques and guidelines outlined in UML has introduced it as a common language for different applications in object oriented form. However, during the past few years, there have been criticisms on UML techniques such as lack of performance measurement techniques. However, Object Management Group (OMG) carries the responsibility for developing UML introduced profiles to support the concepts performance (Object Management Group, 2002). However, for performance evaluation, a real model is needed for formal model.

[*]Corresponding author.
E-mail addresses: zamirpour.ehsan@gmail.com (E. Zamirpour)

doi: 10.5267/j.msl.2014.6.015

Petri net networks are used as a tool for systems modeling specifically systems by separated events (Haghani et al., 2014). In formal model, Petri nets are powerful tools for modeling concurrency and the implementation of Petri nets is based on graph theory. Coloured Petri net is proposed by Kurth Jenson in his PhD thesis in 1980. Different colours in coloured Petri net Provides transfer for Beads, which carry data values and they are distinguishable from each other. In these nets, for a transition to fire, Coloured Petri needs to be consistent with present beads in input places as well as the expressions on the arcs. Coloured petri net is introduced as developing model of petri nets. In addition to places, investment and beads in these nets, the concepts of expression, protective and colour are introduced. Coloured petri nets use the ability of Petri Nets and simple programming languages. The values given in these networks are carried by the beads. Coloured Petri Nets provide more accurate models of complex systems, which are non-phase processing. In these nets unlike petri net, beads are distinguishable from each other because each bead has traits such as colour.

Nowadays, there are many studies accomplished to convert the actual model to the formal models. Among the formal model of Petri nets (Merseguer & Campos, 2004; Kumar & Jasperneite, 2010), queuing networks (Balsamo & Marzolla, 2005) and in the diagrams of UML (Eshuis & Wieringa, 2000; Behbahaninejad et al., 2012), the most activity is allocated to the state diagram. Among the activities carried out, approach of Soger (Merseguer & Campos, 2004) and Saldhana and Shatz (2000) have provided high capabilities. There are two approaches concentrated on labeled generalized stochastic Petri net and object Petri net. Merseguer and Campos (2004) reviewed the performance standard index developed by the Object Management forum and considered the role of UML diagram in software performance. Shatz and Saldhana (2000) turned the state diagram to object petri net and a presented a colored Petri net model for the whole system. To the best of our knowledge, there is no study on the use of performance indexes in Petri net analysis for evaluating the system performance and in this paper, we present an empirical investigation to measure the performance of a system.

2. The proposed method

Earlier, shortcomings of UML were introduced to address different issues. In this section, we present details of the proposed model of this paper. Fig. 1 presents the proposed model of this paper.

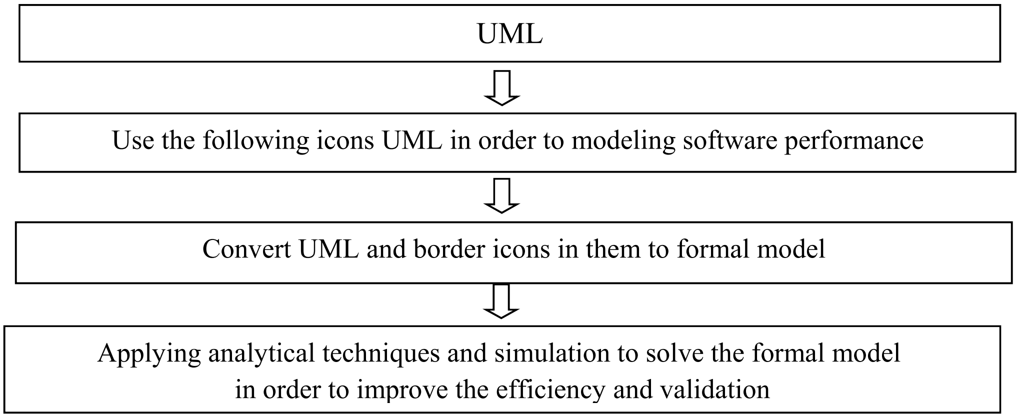
UML
Use the following icons UML in order to modeling software performance
Convert UML and border icons in them to formal model
Applying analytical techniques and simulation to solve the formal model in order to improve the efficiency and validation

Fig. 1. Proposed idea in paper

The emphasis of this paper is to propose a formal model of UML diagrams and analysis the method before implementation.

2.1. The use of colored Petri net model for making performance model

OMG presented some icons to expand UML and Kant and Srinivasan (1992) presented SPT profile for timing ability, performance & time specification. Kumar and Jasperneite (2010) investigated the performance index like other indexes to support the expansion process using stereotypes and labeled values. Each profile includes some stereotypes, where tagged values are traits of stereotypes in profile and they are matched to model stereotype element as comment.

For creating Coloured Petri Nets (CPN) model, diagrams of UML include Annotation performance. Use case diagram shows the work-load. Deployment diagram provides system physical resources as a plan of and Activity diagram expresses the demand of service from resources for various activities.

Performed annotations on Fig. 1 include stereotypes and tagged values and any agent in use case diagram provides a sequence of requests. If this sequence is unbounded, the agent is named with stereotype <<*PAopenLoad* >>, if this sequence is bounded, the agent is named with stereotype <<*PAclosedLoad*>> that has a tag is called *PApopulation,* which indicates the total number of requests in the system. The period between a complete request and the next interaction with the system can be expressed by Tag <<*PAextDelay*>>.

In the agent with stereotype <<*PAopendLoad*>>*,* there is a tag named *PAocurrence*, which represents the time between two consecutive requests. Each agent (minimum) has a use case where details of this service (use case) can be represented by the sequence diagram or activity diagram. Table 1 shows the used stereotype of the algorithm. For more detailed explanation on Stereotypes see (Object Management Group, 2002).

Table 1
Stereotypes and related tags

UML Diagram	Eleman in UML	Stereotype	Tag
Use Case Diagram	Actor	<<PAopenload>>	PAoccurrence
		<<PAclosedload>>	PApopulation
			PAextDelay
Activity Diagram	Action State	<<PAstep>>	PAhost
			PAdemand
	Transition	----------------	PAprob
Deployment Diagram	Node	<<PAhost>>	PAschdpolicy
			PArate

In modeling of Computer Systems Performance, two aspects should be noted: synchronization between different activities and contention for resources must happen, simultaneously (Kant & Srinivasan, 1992). The reason of using Petri net as a preferred net than queues is to protect these two properties. However, queue protects just the second property (Kant & Srinivasan, 1992). In other words, in such sequences simultaneously, in Activity diagram, queuing network becomes a problem.

In a system, some action can use a source and the service rate fire for actions is different and this is shown by chains. Therefore, in coloured petri net, different colors of beads that are assigned to the beads indicate this topic.

2.2 Proposed algorithm

In any CPN model, *T* represents transfer and *c* represents different colours and the number of customer classes. Transfers in CPN indicate UML model resources. For each actor with stereotype, <<*PAclosedLoad*>> defines one addition transfer in CPN and the time between the end of an interaction and the beginning of the next iteration. A CPN model can be decomposed into several independent sub-models, which are independent. Each sub-model may carry its workload. Requests of belonging to the same sub-model can have multiple classes. In CPN model, Classes in each sub-model in the model are shown with different colors. The color of each sub-model is unique and it cannot be repeated in other sub-models. An open petri net is the one that has input/output to the external environment. A closed petri net does not have any relationship with external environment. Open and closed petri nets support stereotypes <<*PAopenLoad*>> and <<*PAclosedLoad*>> in Use case diagram. By using activity diagram, we obtain Routing matrix of Petri net (Probability matrix). Note that a source can be used by several actions in systems. For supporting this case in CPN model, the following traits are defined:

The system resources are shown with RES= $\{res_1, res_2, \ldots, res_t\}$. For each resource reserves, one trait by the name Counter[res] is defined. *Counter [res]* represents the sum of the actions that Service will apply form *res*. To identify the resources, a unique index is defined with the attribute *identity* as follows,

Identity $[res_i]$=I for each res_i € RES

The total actions that use the resource *res*, are presented by

$ACT= \{act_1, act_2, \ldots, act_m\}$.

It is clear that we have Counter *[res] = m*.

For each resource, *res Є RES,* the whole actions *in { act Є ACT* │ *resource(act)=res}* are tagged with a unique number in the interval [1, 2,…, counter(res)]

This unique number is determined by trait *indicator [act]*. We define the trait *indicator [act],* for each action *act € ACT*).

An input in open petri net entered a transfer that the action of initial interaction diagram uses that resource. The input rate to Petri net is equal to tag *PAoccurrence* and actors use the activity diagram for reaching its use case. The transition probability in Petri net from tag <<*PAprob*>> Will result in a corresponding activity diagram. Suppose actor x is annotated by stereotype <<*PAopenload*>>), then the algorithm of converting model UML to model CPN to assess the performance of the software is as follows,

1. The following attribute values are specified:
counter [res] √ res € RES,
indicator [act] √ act € ACT,
C=maximum $_{\text{res € RES}}$ {counter [res]} and T
2. Calculate the probability matrix
3. Calculate the service rate for customers with class *r* and transfer *i*

$$SR[i,r] = rate[r] / demand\ [act]$$

$$where \quad i = identity\ [resource\ [m]],$$

$$r = indicator\ [act]$$

4. vector λ[r], which is an indicator to the entry for the customer with class *r* is as follows,

$$\lambda[r] = arrivalrate[X]$$

In other words, entry rate with respect to the use case (tag PAoccurence) that lead to activity diagram is calculated.

If, agent *x* with stereotype <<*PAclosedLoad*>> is fretwork, the following changes are applied to the above algorithm:
T=│RES│+1
- Total transfers deploy a greater number of resources in the deployment diagram.

- Added transmission is determined by zero and as mentioned it comes to address the delay (for example the time of thought). Let s be a class for client, then the transfer rate of transfer fire for all customers classes is as follows,

$$SR[0,\ s] = 1/\ \ extdelay\ \ [X]$$

- The amount of N is equal to the number of requests in the system and the actor is calculated according to tag PApopulation. The above tag in petri net is indicated by trait *population,* therefore we have,

$$\mathrm{N} = \mathrm{population}[\mathrm{X}]$$

By combining the above two algorithms, a comprehensive algorithm can be achieved, and the number of sub-model CPN is equal to A+B where A and B are the number of actors with stereotypes <<PAopenLoad>> and <<PAclosedLoad>>, respectively.

3. Case study

The case study is devoted to an investigation on a hypothetical bank. Fig. 2 shows case study diagram with just one case study.

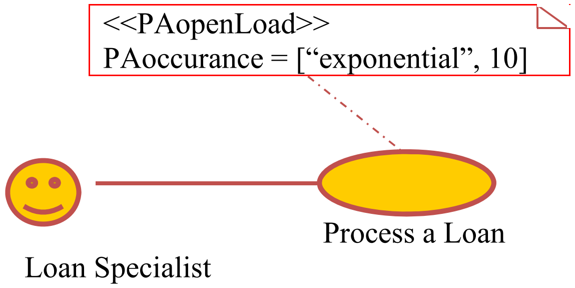

Fig. 2. Case study diagram with performance annotation

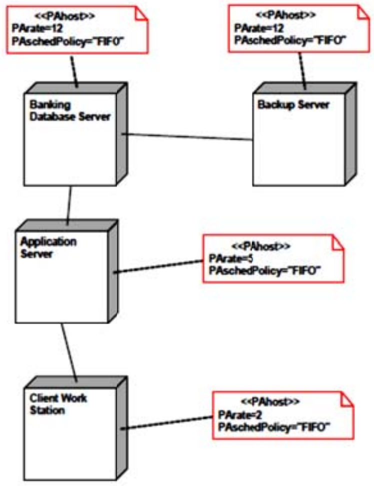

Fig. 3. Deployment diagram with performance Annotation

Deployment diagram of the system is shown in Fig. 3. Fig. 4 shows the activity diagram that leads to case study. The middle graph, which displays resources and the number of using them is considered in Fig 5. In this graph, for each action of activity diagram, one node is considered by numbering the resources in the deployment diagram and we consider when a resource is used more than once, one unit is added to indicator. Transition probability on the edges of the graph can be seen.

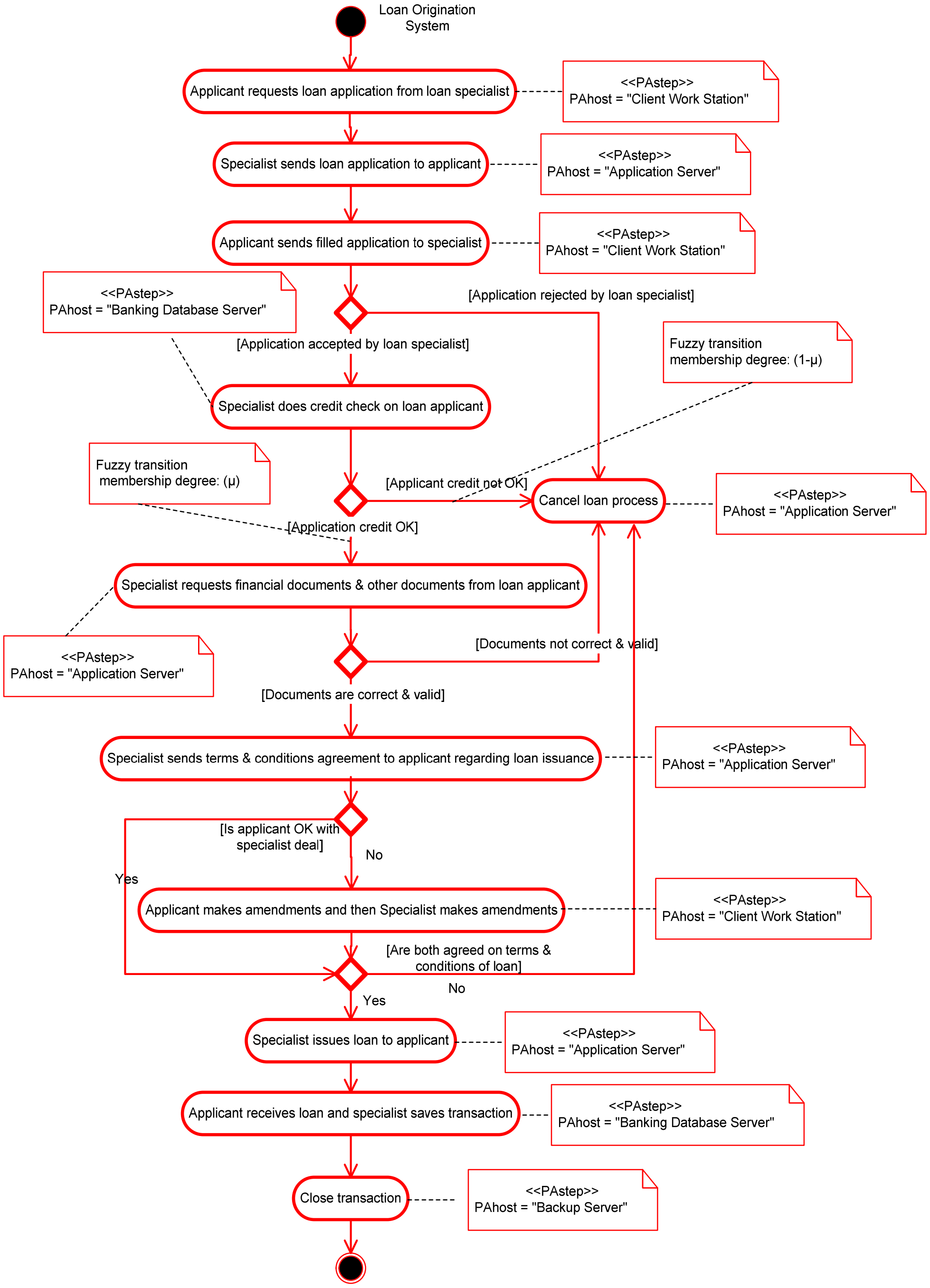
Loan Origination System
Applicant requests loan application from loan specialist
<<PAstep>> PAhost = "Client Work Station"
Specialist sends loan application to applicant
<<PAstep>> PAhost = "Application Server"
Applicant sends filled application to specialist
<<PAstep>> PAhost = "Client Work Station"
<<PAstep>> PAhost = "Banking Database Server"
[Application rejected by loan specialist]
[Application accepted by loan specialist]
Fuzzy transition membership degree: (1-μ)
Specialist does credit check on loan applicant
Fuzzy transition membership degree: (μ)
[Applicant credit not OK]
Cancel loan process
<<PAstep>> PAhost = "Application Server"
[Application credit OK]
Specialist requests financial documents & other documents from loan applicant
<<PAstep>> PAhost = "Application Server"
[Documents not correct & valid]
[Documents are correct & valid]
Specialist sends terms & conditions agreement to applicant regarding loan issuance
<<PAstep>> PAhost = "Application Server"
[Is applicant OK with specialist deal]
No
Yes
Applicant makes amendments and then Specialist makes amendments
<<PAstep>> PAhost = "Client Work Station"
[Are both agreed on terms & conditions of loan]
No
Yes
Specialist issues loan to applicant
<<PAstep>> PAhost = "Application Server"
Applicant receives loan and specialist saves transaction
<<PAstep>> PAhost = "Banking Database Server"
Close transaction
<<PAstep>> PAhost = "Backup Server"

Fig. 4. Activity diagram with performance annotation

Fig. 6 shows coloured petri net, which is associated with the middle graph by assuming that the use case associated with to the activity diagram, has tag, with $\lambda = 1/10$.

PAoccurrence=[“exponential”,10]

“PAdemand” for action that use the resource application server, takes the following values fot coloured petri net with 5 classes:

$$\{0.2, 0.1, 0.5, 0.3, 0.2\}$$

Fire rate on Petri net transfer for resource is application server is:

SR=rate [res]/demand [act] = (25,50,10,16.666,25)

By specifying entry rate (λ) and fire rate, we may perform the analysis of Petri nets. Implementation tools for drawing diagrams UML, tools Rational Rose (ver7) and to simulate coloured petri net is based on CPN tools.

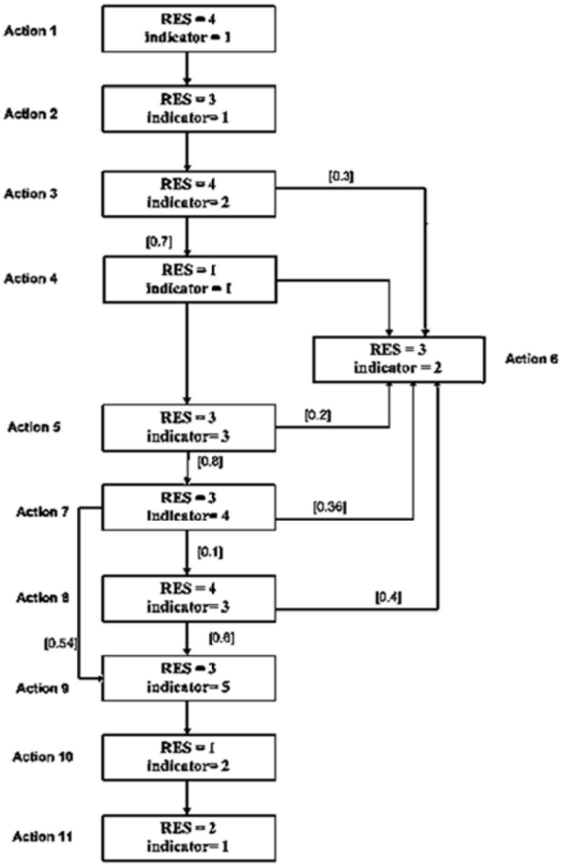

Fig. 5. Intermediate graph between pragmatic model and formal model

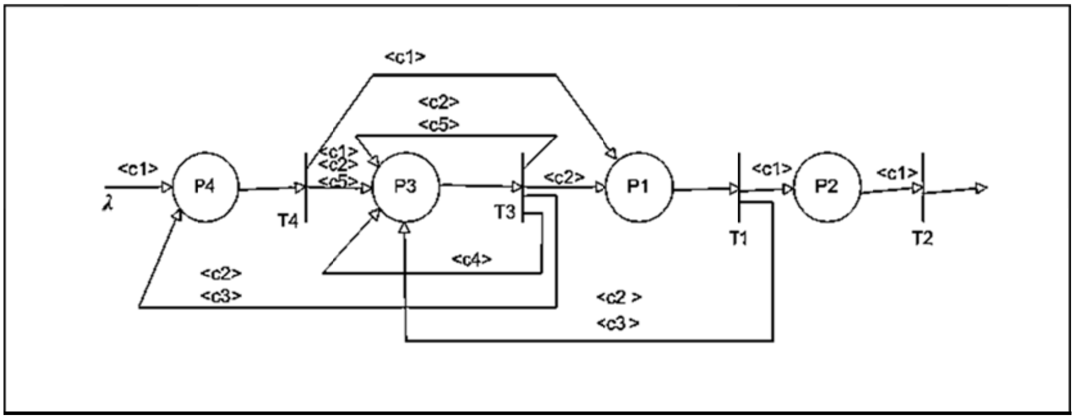

Fig. 6. CPN model for activity diagram

The CPN tools after applying simulation provides the review probability of meters associated with performance. In this case study, productivity throughput and queue length are emphasized. It is clear that the number of present beads in each place indicates the queue length. Fig. 7 shows the queue length and determines the necessary changes on the model to reduce the queue length. The confidence of the system hardware configuration in the early stages of system development before the implementation causes the cost reduction of implementation. By checking the feedback and analysis of this model, we may study the changes on the model and after reaching the optimal status, the system can be implemented.

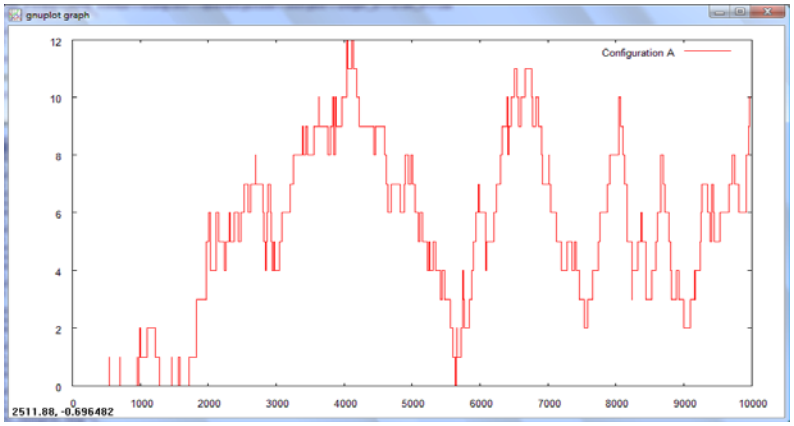

Fig. 7. Queue length in design phase

4. Conclusion

In this paper, a new mechanism for evaluating the software systems has been explained. Providing a formal model based on the real model (UML) allows us to develop the systems along with software process models. Ii also helps us investigate software non-functional requirements. The results of simulation give the possibility of performance evaluation before implementation.

References

Balsamo, S., & Marzolla, M. (2005, July). Performance evaluation of UML software architectures with multiclass Queueing Network models. In *Proceedings of the 5th international workshop on Software and performance*(pp. 37-42). ACM.

Behbahaninejad, P., Harounabadi, A., & Mirabedini, S. J. (2012). Evaluating software architecture using fuzzy formal models. *Management Science Letters*,*2*(2), 469–476.

Eshuis, R., & Wieringa, R. (2000). *Requirements-level semantics for UML statecharts* (pp. 121-140). Springer US.

Kant, K., & Srinivasan, M. M. (1992). *Introduction to computer system performance evaluation*. McGraw-Hill College.

Kumar, B., & Jasperneite, J. (2010). UML profiles for modeling real-time communication protocols. *Journal of Object Technology*, *9*(2), 178-198.

Merseguer, J., & Campos, J. (2004). Software performance modeling using UML and petri nets. In *Performance Tools and Applications to Networked Systems*(pp. 265-289). Springer Berlin Heidelberg.

Haghani, S., Abbasnejad, Y., & Harounabadi, A. (2014). An evaluation of the software architecture efficiency using the Clichés and behavioral diagrams pertaining to the unified modeling language. *Decision Science Letters*, *3*(3), 411-430.

Object Management Group (2002). *UML Profile for Schedulability, Performance and Time Specification*, March 2002, http:/www.omg.org

Saldhana, J., & Shatz, S. M. (2000, July). Uml diagrams to object petri net models: An approach for modeling and analysis. In *International Conference on Software Engineering and Knowledge Engineering* (pp. 103-110).