International Journal of Computer Applications Technology and Research
Volume 4– Issue 2, 142 - 148, 2015, ISSN:- 2319–8656

Performance Evaluation using Blackboard Technique in Software Architecture

Fatemeh majidi
Department of Computer Engineering, Ardabil Science and Research branch, Islamic Azad University,
Ardabil, Iran

Department of Computer Engineering, Ardabil Branch, Islamic Azad University,
Ardabil, Iran

Ali Harounabadi
Department of Computer Engineering, Central Branch, Islamic Azad University,
Tehran, Iran

Abstract: Validation of software systems is very useful at the primary stages of their development cycle. Evaluation of functional requirements is supported by clear and appropriate approaches, but there is no similar strategy for evaluation of non-functional requirements (such as performance). Whereas establishing the non-functional requirements have significant effect on success of software systems, therefore considerable necessities are needed for evaluation of non-functional requirements. Also, if the software performance has been specified based on performance models, may be evaluated at the primary stages of software development cycle. Therefore, modeling and evaluation of non-functional requirements in software architecture level, that are designed at the primary stages of software systems development cycle and prior to implementation, will be very effective.
We propose an approach for evaluate the performance of software systems, based on black board technique in software architecture level. In this approach, at first, software architecture using blackboard technique is described by UML use case, activity and component diagrams. then UML model is transformed to an executable model based on timed colored petri nets(TCPN) Consequently, upon execution of an executive model and analysis of its results, non-functional requirements including performance (such as response time) may be evaluated in software architecture level.

Keywords: Software Architecture, Blackboard Technique, Performance Evaluation and time colored Petri net.

1. INTRODUCTION

Within the recent decades, the software complexities have been increased day to day and demands for more powerful and high quality software have been increased. Therefore, software development based on principles and methodologies that in addition to reduction of costs, meet all expected features of shareholders (functional and non-functional requirements) seems to be necessary. Establishing non-functional requirements in software engineering was raised recently whilst they have considerable effect on success of software systems. Software Architecture (SA) is established at the first stages of design and has a significant effect on access to nonfunctional requirements of software system. Therefore, establishment of an executive model of SA and evaluation of nonfunctional requirements thereby is a cheap solution for prevention of time and cost waste for achieving the qualitative goals for development of software systems. using the patterns and styles of software architecture is a procedure to exploit the possibilities of a design which is based on architecture and architectural styles promote the characteristics like having the possibility of reusability, providing with supporting documents, finding risks at early stages, and upgrating.

One of important goals that are followed during analysis of architecture quality is verifying the architecture's access to qualitative features such as performance [1]. In the most software systems, special methods are used for evaluation of qualitative features. Special methods are applicable commonly after architectural implementation means when an executable specimen of system is available. If after applying the special methods, it is revealed that the architecture selected for system may not respond the nonfunctional needs, more time and cost is needed for system architecture changing. In consideration of this subject, we need alternative methods for evaluation of qualitative features which are applicable in initial stages of production process. Establishment of executable models of system architecture is one of solution that may respond the raised problems. An executable model of architecture is assumed as a formal description of architecture through which may analyze the behavior of final system before architecture implementation and get aware of problems and their in performance and take measure for architecture implementation more confidently and so avoid extra costs and even its failure. In continue, different parts of paper are explained: in second part, a general description of blackboard technique, performance model in unified modeling language (UML) and time color Petri net (TCPN) is presented. In third part, some works related to the subject of this paper are reviewed. In fourth part, the offered model is described. In fifth part, a case study is analyzed for evaluation of offered method and in sixth part, a general conclusion of suggested method is explained.

2. BACKGROUND

In this section, a general description of performance modeling in UML, blackboard technique and timed colored petri net models is presented.

2.1 Blackboard technique

In the codified classification of techniques, blackboard is placed in the centralized group. One user is executed on a distinct control set and includes common data which is accessible by these users.
Blackboard is a technique therein independent processing components are referred to as knowledge resource that is operated on the common storage in the name of blackboard. Knowledge resources have no direct interaction with each

other, when blackboard is executable, knowledge resources run on the blackboard as an opportunity seeking [2].
This architectural style is always promoting and extending and a structural solution for reaching to the integrity. In plenty of systems particularly systems consisted of prefabricated components, data integrity is provided by blackboard mechanism. Major advantage of this method is that the users are available separate from each other. In addition, common data is an independent part of users. Therefore, this style is scalable and new users may be added easily.

2.2 Performance Modeling in UML

Software architecture describes the system in high abstraction levels through specifying the structural and behavioral aspects. But, unified molding language diagrams may not be used to evaluate the software architecture, because some architectural features are not executable using them. In consideration of this subject, a strategy was offered by OMG including performance sub index, similar to other indices for supporting the extension process, stereotypes and labeled values that improves the applicability of these features [3].

2.3 Time colored Petri net

Colored petri nets are used for formal description of activities flow in the complex systems and provide the requirements of concurrency and parallelism exhibition. Classic petri nets are not suitable for modeling the systems with large space or a complex temporary behavior. In these cases, we must use a developed petri net model having color and time. This model is the base of a framework that is used for solving the problems related to design and control in complex systems. In these networks, the concept of time is introduced by global element called global time. The values selected by this time explains the model time. This model may be an integral number that indicates the discrete time or maybe a true number explaining the continuous time. This value of time that is pertained to each token is referred to as stamp time that indicates the first time of model therein token may be used. As a result, these nets will be appropriate for evaluation of qualitative requirements (response time etc.) in SA[4].

3. RELATED WORKS

Model-based methods development for evaluation of systems and computer nets is referred to a long time ago. Correct application of these models may provide appropriate attitudes for evaluation of nonfunctional needs. Due to low knowledge level of software architect, evaluation of these features is not applicable for software architecture, because software architecture for describing the software architecture uses specific marks and signs which are not usable for experts evaluating these features. Therefore, a solution must be found to fill the gap between software designers and nonfunctional features evaluation experts. One of solutions is using the tools and markings of software modeling together with options added thereto that may considerably remove this gap.
Fukuzawa and Saeki [5] presented a method therein software architecture is described by UML Component diagram. Then, the above algorithm has been transformed to colored petri net by an algorithm and ultimately the performance is evaluated, so that the own component and its connector are transformed to a colored petri net but its interface is transformed to a place of colored petri net.
Balsamo and Marzolla [6] presented a method therein software architecture is described by UML Use Case, Activity and Deployment diagrams, then operational profiles related to performance are annotated therein. Ultimately, to evaluate the performance, UML diagrams are transformed to an executive model based on Queuing Networks.
Petit and Gamma [7] described the software architecture by collaboration diagram and then converted to Petri net. This method is used for evaluation of performance and reliability. In this method, a collection of predetermined molds in colored Petri nets formed based on objects' behavioral roles are used. These behavioral roles are formed based on available objects structuring in COMMET method, but are not dependent to a specific method and used within different application ranges. Later, results obtained for colored Petri net are reflected in unified modeling language diagrams and the designer may improve the design quality and consequently improve the performance and reliability of system.
Gyarmati et al [8] offered a model therein software performance engineering (SPE) is used for evaluation of performance specifications of software architecture and fabrication and analysis of software executive model resulted from ordinal diagram of unified modeling language. In this method, class diagram and unified modeling language placing is used for describing the software architecture completely, but is not used in the conversion process. Architectural descriptions are converted to the developed queue net to evaluate the performance specifications.
In this paper, three major objectives are under consideration as follows:
-Evaluation of information system performance based on blackboard technique;
-An algorithm for converting blackboard technique to component diagram;
-Converting UML diagrams to the formal models based on features available in blackboard technique for evaluation of its performance.

4. THE PROPOSED METHOD

The main method in this paper is performance evaluation using blackboard technique in software architecture. For this purpose, firstly software architecture based on blackboard technique is described by UML, later operational profiles related to performance feature is annotated therein. In continue, an algorithm is offered for transformation of UML model to TCPN model and ultimately the said nonfunctional requirements are evaluated by suggested techniques at the SA level.

4.1 Description of Software Architecture by UML Diagrams

In this article, to describe the software architectural structure and behavior, use case, component and activity diagrams are used. In continue these diagrams and notations related to performance are explained.

4.1.1 The Role of Use Case Diagram and Annotation of Performance Specification Therein

Use case diagram describes the functional requirements of system and interaction between system and environment [9]. In this paper, this diagram is used for exhibition of functional requirements and working load applied to the system in SA description. Annotations related to performance in this diagram are related to actors that requesting service from system.
The actors indicating a sequence of unlimited requests out of system are annotated by "PAopenLoad" stereotype and actors indicating a fixed population of requests from system are annotated by "PAclosedLoad" stereotype. "PAclosedLoad" stereotype has a tag called PAoccurrence that indicates the interarrival time between two subsequent requests. "PAclosedLoad" stereotype has two tags named PApopulation and PAextDelay that respectively indicates "the number of

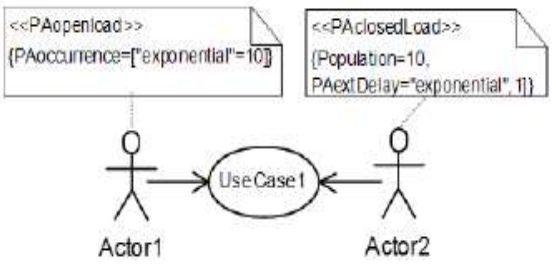
requests" and "the time spent by each completed request before the next interaction with the system". An annotated use case diagram is exhibited in figure 1.

4.1.2 The Role of Component Diagram and annotation of Performance Specifications Therein

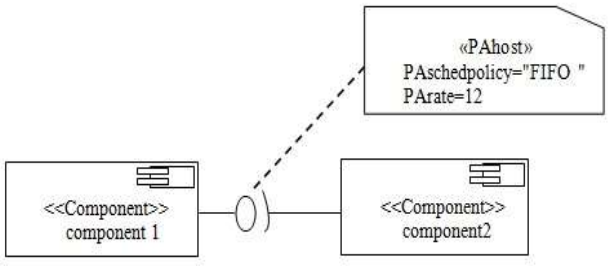
Component diagram describes the software system. In this paper, this diagram is used for describing the architectural structure based on blackboard technique. Moreover, a component includes interfaces that each interface defines a collection of component performed operation. Notations related to performance of this diagram are related to the interfaces and components. In this diagram, each component is noted with <<PAhost>> stereotype that specifies the software resource used in this project. This stereotype includes PAschedpolicy label that specifies the system schedule policy. Each interface is noted by <<PAstep>> stereotype that specifies the tasks performance time by the component together with PAdelay and PAdemand labels [10], [11]. Figure (1) shows an annotated component diagram. In addition, PArate label indicates the processing rate of processing source related to respective component.

4.1.3 The Role of Activity Diagram and Annotation of Performance Specifications Therein

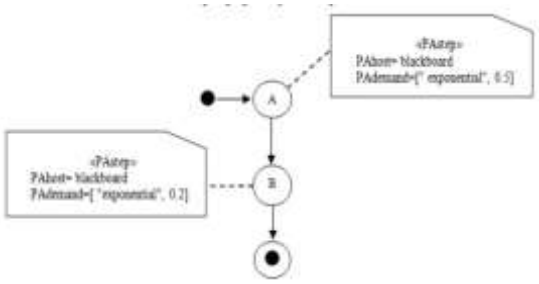
Activity diagram describes the software system behavior. This diagram is a graphic exhibition that shows the control flow from one activity to another. Notations related to performance in this diagram are related to transfers [9]. Each transfer is noted with <<PAstep>> stereotype which demonstrates the service provided in a component and according to its location and includes PAhost and PAdemand labels that each one denotes component location and service request, respectively. An annotated ctivity diagram is exhibited in figure 1.



(a) Annotated UML Use Case Diagram



(b) Annotated of integrated modeling language component diagram



(c) Annotated of integrated modeling language activity diagram

Figure 1. Annotated of integrated modeling language diagrams

4.2 The offered algorithm for converting modeling language diagrams to time colored Petri net

The offered algorithm in this paper for converting unified modeling language to time colored Petri net is raised for incorporating an executable model for evaluation of software architecture that includes following stages:

First stage: Description of architectural structure based on blackboard technique component diagram and determination of performance specifications in this diagram, blackboard-based architectural structure is described.

Second stage: Description of blackboard-based architectural behavior In this paper, to describe the software architecture behavior, case use and unified modeling language activity diagrams are used. In the suggested course of action, case use diagram is used to exhibit the functional needs and function load applied on the system during software architecture description. Activity diagram describes also the system behavior and shows the control process from one activity to another one.

Third stage: Evaluation of the blackboard-based architecture Whereas various agents have varied function loads in the system, T-CPN model contains following models which are independent from each other and each one will has their own functional load. Furthermore, requests related to one sub model may have several classes that each one is shown with different colors in T-CPN. Each color is unique and may not be repeated in other classes.

Open Petri net contains input and output to the external environment and shown by <<PAopenload>> stereotype that is used in the case use diagram. Whereas several methods can use a source in the system, we have following definitions in T-CPN model:

If it is assumed that sources are exhibited as RES= {res1, res2 ... res n} for each source, res ϵ RES is defined as a feature called [count [res]]. [Count [res]] denotes total requests that request service from res, index feature is a unique index for identification of sources. Places which use res resource are shown by}ACTION={ action1,action2,..., actionn}, it is obvious that count[res]=a for each source labels total requests in {action ϵ ACTION | resource(action)= res} set by a unique number in range [1,2,...,count[res]]. This unique number is shown by index [action] feature.

If agent x is noted by <<PAopenload>> stereotype, feature values are determined as below:

Count [res] $\forall$ res ϵ RES

Index [action] $\forall$ actionϵ ACTION

C = MAX res ϵ {COUNT [res]} (1)

To show the customers service rate with class r, SR [i,r] in transfer i is used. M is an action in activity diagram.

SR [i,r]= rate[r]/demand [action] where i = index [resource[m]]. (2)

r = index [action]. (3)

λ]r] for is considered for showing the customer input rate with class r that is defined as below:

λ [r] = arrival rate [x] (4)

The input rate is used based on labeled case use that resulted in use of activity diagram.

4.3 Evaluation of response time in software architecture Level

Performance metrics such as response time, queue length etc. may be evaluated using the said evaluation method. To compute the response time, time interval between request and first received response by the other side must be analyzed. In fact:

$$T_R=T_S+T_D \quad (5)$$

T_R: Response time

T_S: Service time

T_D: Delay time

Delay time may be defined as delay time in processing queue To analyze the queue length, tokens number in place must be calculated.

5. CASE STUDY

In this paper, hotel reservation system was assumed as case study, so that this system was implemented on blackboard technique and ultimately is evaluated using the offered method. Blackboard technique is shown in Figure 2. Figure 3,4 and 5 show case use, component and activity diagrams of unified modeling language of hotel reservation system. Figure 6 show activity diagram of hotel reservation system. In this scenario, firstly the user declares its request on hotel reservation and the system during some stages responds by its agents in consideration of the user request. For evaluation of nonfunctional needs (such as performance), diagrams shown in Figure 4 and 5 are converted to time colored Petri net model. Final model of time colored Petri net is exhibited in Figure 6. To evaluate the performance (such as response), 4 requests are input to the system by users and upon their execution on time colored Petri net, valuable results are obtained for evaluation of nonfunctional needs on SA level. Table 1 shows the response time related to users.

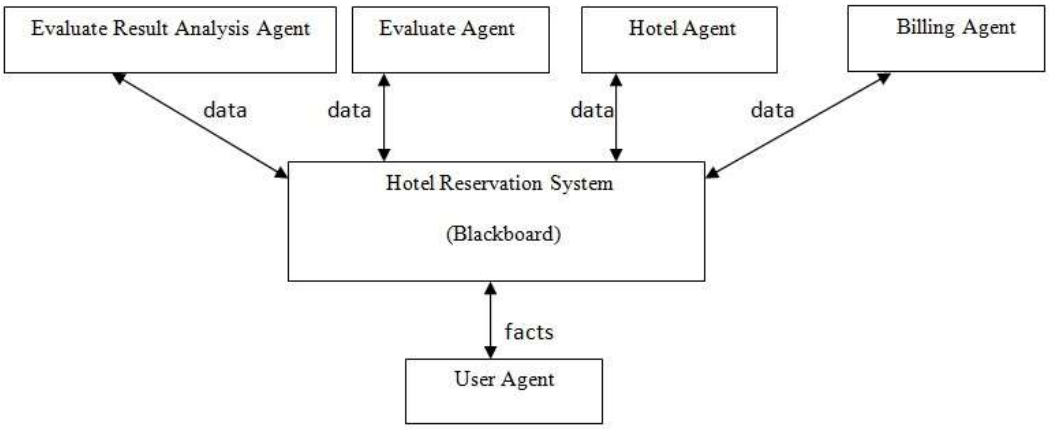


Figure 2. Abstract model of hotel reservation system

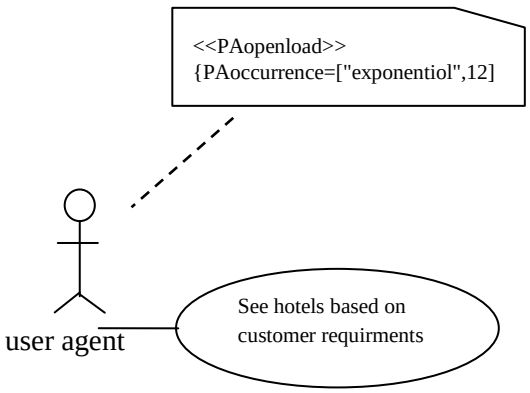


Figure 3. Exhibition of hotel reservation system with CR card

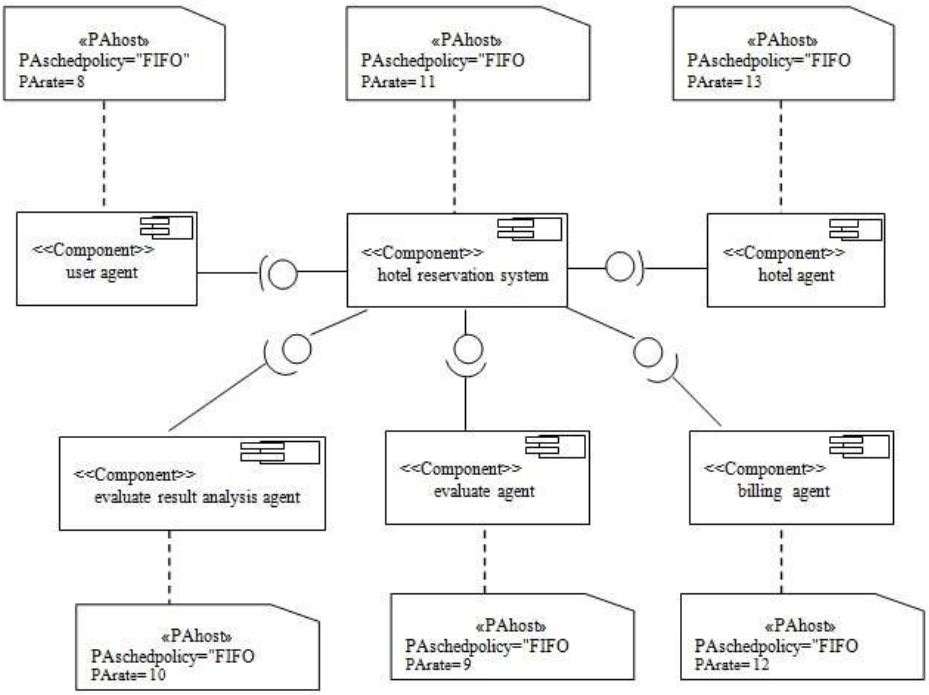


Figure 4. Annotated of performance in component diagram

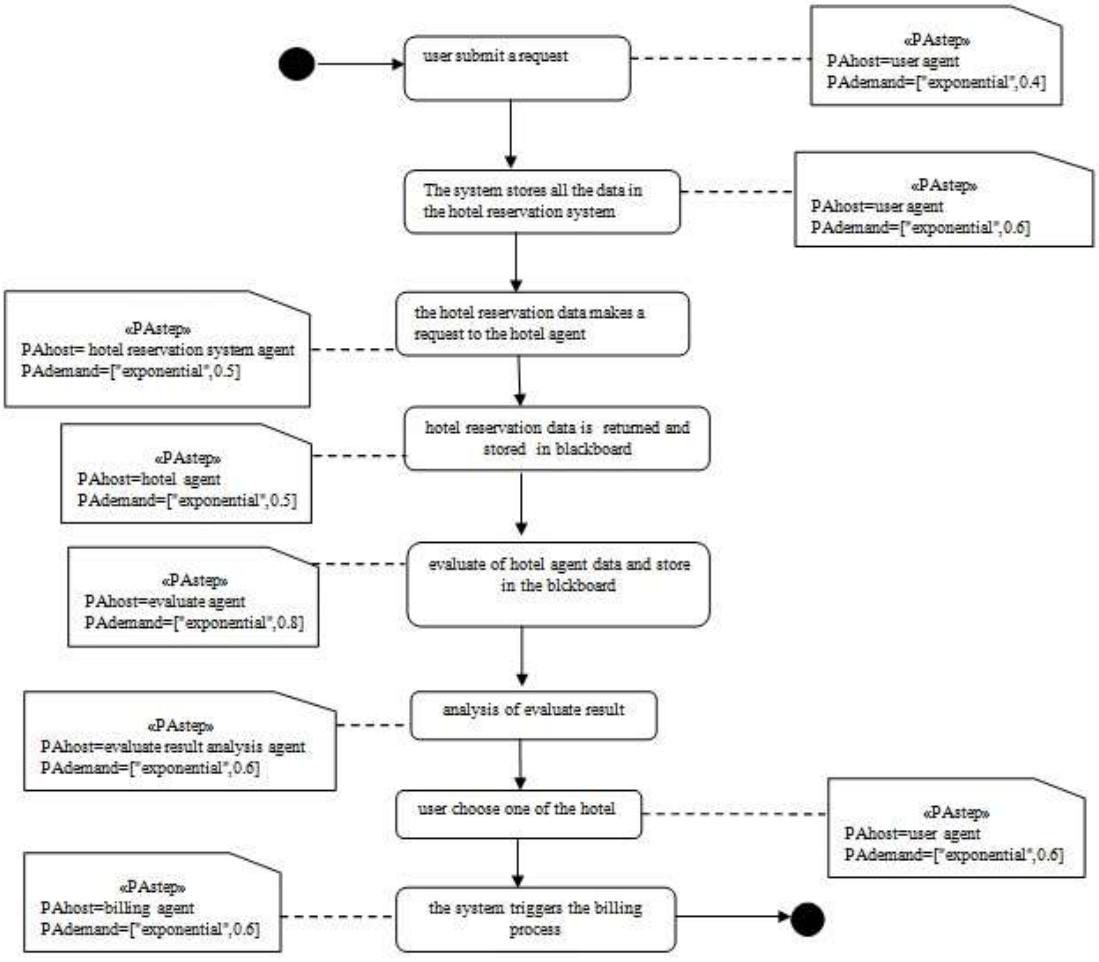


Figure 5. Hotel reservation activity diagram

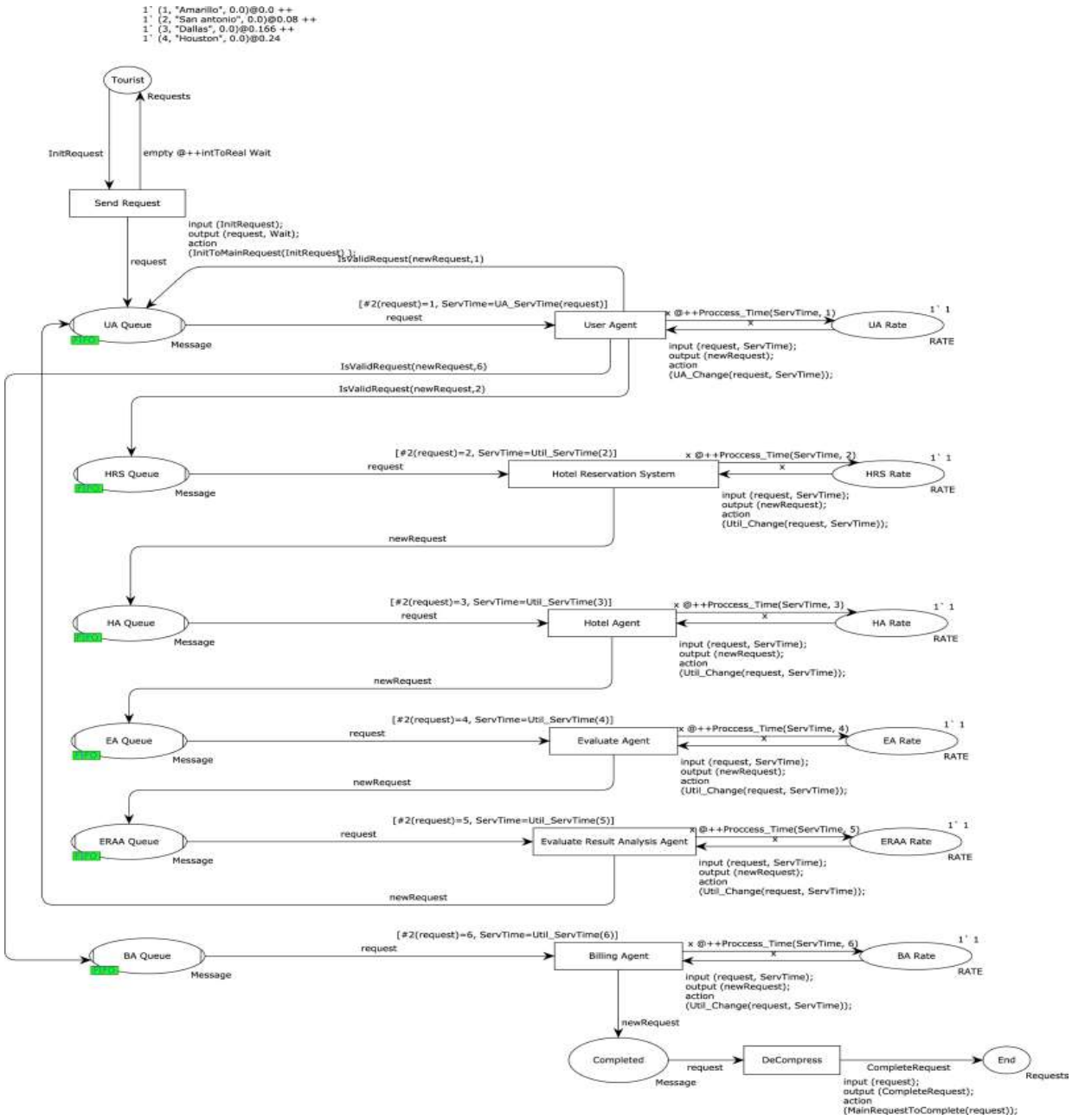


Figure 6. Time colored Petri net

Table 1. Response time

Number of request	Response time
1	0.16861
2	0.56993
3	0.52483
4	0.295

6. CONCLUSIONS

In this paper, we have presented a strategy for evaluation of performance of nonfunctional requirements in software architecture using blackboard technique modeled by UML diagrams. So, the software system may be validated for meeting or not meeting the nonfunctional requirements of case at the primary stages of software systems development cycle. The general analysis framework in this method is formed based on formal models (TCPN) that accordingly is free of ambiguity. Whereas in this method, UML diagrams are used for description of software architecture based on blackboard technique, therefore description of SA by means of achievements of analysis and design stages will be very reasonable and low-cost. on the other hand, a transformation has been presented for establishment of a TCPN-based executive model from UML model .There are a lot of tools for working with UML models and UML models may be transformed to TCPN-based executive model automatically. In addition, other nonfunctional requirements may be evaluated by means of other architectural specifications.

7. REFERENCES

[1] Technical Report IEEE P1471-2000. Recommended Practice for Architectural Description of Software Intensive Systems, IEEE Standards Department, The Architecture Working Group of the Software Engineering Committee, (September 2000).

[2] Clements, P., Bass, L., Garlan, D., Ivers, J. Little, R. Nord, R. and Stafford, J. 2010. Documenting Software Architectures: Views and Beyond. Second Edition, Publication City/Country New Jersey, Addison Wesley.

[3] Object Management Group (OMG). 2002. UML Profile for Reliability, Schedulability, Performance and Time Specification.

[4] Jensen, K. and Kristensen, L. 2009. Coloured Petri nets: modeling and validation of concurrent systems. Springer-Verlag.

[5] Fukuzawa, K. and Saeki, M. 2002. Evaluating Software Architectures by Coloured Petri Nets. in SEKE02 14th International Conference on Software Engineering and Knowledge Engineering, ACM, Ischia, Italy.

[6] Balsamo, S. and Marzolla, M. 2005. Performance Evaluation of UML Software Architectures with Multiclass Queueing Network Models. ACM Workshop on Software and Performance (*WOSP*).

[7] Pettit, R. G. and Gomaa, H. 2004. Improving the Reliability of Concurrent Object Oriented Software Designs. proceeding of the ninth IEEE international workshop on object oriented real time dependable systems.

[8] Gyarmati, E. and Strakendal, P. 2002. Software Performance Prediction-Using SPE. Master Thesis Software Engineering, Department of Software Engineering and Computer Science Blekinge Institute of Technology, Sweden.

[9] Object Management Group (OMG). 2005. Unified Modeling Language (UML). Version 2.0.

[10] Merseguer, S. Bernardi, S., Campos, J. and Donatelli, S. 2002. A Compositional Semantics for NML State Machines Aimed at performance Evaluation. proce. Of the 6th International Workshop on Discrete Event Systems, 295-302.

[11] Merseguer, J., Campos, J. and Mena, E. 2003. Analysing Internet Software Retrieval System: Modeling and Performance Comparison. Wireless Networks: the Journal of Mobile Computation and Information, vol. 9, no. 3, 223-238.