Management Science Letters 4 (2014) 2011–2020

Contents lists available at GrowingScience

Management Science Letters

homepage: www.GrowingScience.com/msl

An executable software architecture model for response time and reliability assessment

Zahra Norouzi[a*] and Ali Harounabadi[b]

[a]*Department of Computer, Science and Research Branch Damavand, Islamic Azad University , Tehran, Iran*
[b]*Department of Computer, Central Tehran Branch, Islamic Azad University, Tehran, Iran*

C H R O N I C L E

Article history:
Received January 20, 2014
Accepted 5 July 2014
Available online
August 16 2014

Turnover intentions
Key Software Architecture
Performance Evaluation
Reliability
Fuzzy stereotypes
Fuzzy colored Petri net

A B S T R A C T

With the increasing use of the Unified Modeling Language (UML) diagrams to describe the software's architecture and the importance of evaluating nonfunctional requirements at the level of software architecture, creating an executable model from these diagrams is essential. On the other hand, the UML diagrams do not directly provide features to evaluate nonfunctional system requirements. Thus, these capabilities can be added to UML diagrams by applying efficiency and reliability stereotypes. Because the techniques used in the UML is able to deal with certain matters, we develop uncertain UML, stereotypes and tags. In this paper, the architecture of a software system is described by using use case diagram, sequence and deployment of unified modeling language diagrams with annotations fuzzy stereotypes related to response time and reliability. The proposed method for calculating the response time and reliability based on fuzzy rules are introduced, and the algorithm is implemented for an executable model based on colored fuzzy Petri net.

1. Introduction

Developing suitable products is always one of the main concerns among software engineers. While the cover functional and nonfunctional requirements of system are offered under little expense to the customer. Nevertheless, satisfying the nonfunctional requirements of software engineering is still in its early stages. The nonfunctional requirements especially performance and reliability have a great impact on the success of software systems. On the other hand, nonfunctional requirements are analyzed and evaluated during the development process especially in the early phases of development. If these needs are not met, properly, the system will probably need fundamental changes, it may even lead to the abandonment or redevelopment of the system and the nature of the users' needs is faced with uncertainty. Therefore, the primary concern of this article is the use of vague and uncertain information in evaluating the nonfunctional requirements such as performance and reliability. Among various markings, Unified Modeling Language (UML) has been used to

*Corresponding author.
E-mail addresses: zahra.norozi@gmail.com (Z. Norouzi)

doi: 10.5267/j.msl.2014.8.020

describe the software architecture, but UML deals with certain issues. In order to better exploit the features of this language, this paper considers uncertainty in parameters in stereotypes and then uses the stereotypes in UML diagrams. Finally, the UML diagrams are mapped to fuzzy colored Petri nets in order to evaluate the Performance and Reliability. In the second part of the paper, the platform research is described and in the third part, related works are reviewed. The proposed method of this paper to create executable model based on formal descriptions of architectural is described in fourth section. The fifth part presents a case study and the sixth part is devoted to conclusions.

2. Platform

Work Platform includes Unified Modeling Language, fuzzy Logic, petri nets, performance stereotypes and stereotypes based on quality of service.

2.1 Petri Nets

Carl Adam Petri is believed to be the first who introduced Petri nets Theories. In fact, he succeed to display the communication between system components by a graph. Petri nets are Bisection directed graphs, which means that the nodes in Petri nets include two types of circle nodes or place and bar nodes or transition. Being directed is also for the reason that these two elements are connected with arcs.

2.2 Color Petri Nets

Color Petri nets was introduced by Kart Jenson as an extended model of Petri nets. The concepts of color, guard and words are introduced in addition to places, transitions and tokens. Color petri nets use the ability of the simple petri nets and programming languages. Amounts of data in these networks are carried by tokens and tokens in these nets are distinguishable from each other despite the petri nets. Transitions in color petri net are the basis for the hierarchical structure. This feature allows the color petri net to be viewed in different levels of abstraction. We use this feature for extracting the color petri nets from unified Modeling Language diagrams.

2.2 Fuzzy Color Petri Nets

Color Petri nets deals with complicated systems, which got a degree of uncertainty in their descriptions, Color petri net is compatible with classical logic. To model such systems, it is necessary to show the uncertainty in Petri nets and for this, it is necessary to introduce the concept of fuzzy in Petri model. Therefore, different types of fuzzy Petri nets have been introduced. In fuzzy petri net, the ability to use the fuzzy variables is added and applied specifically for modeling fuzzy rules and fuzzy inference. In fuzzy petri net fuzzy, fuzzification is used on each element described as follows:

- Fuzzification in token level: fuzzy token is a generalization of the standard Petri net. In standard Petri net, the value of token belongs to the set {0, 1} but the value of fuzzy token is between distance [0, 1]. It is an interesting idea that fuzzy token values include phrases such as low, medium, high, etc.
- Fuzzification in place level: a fuzzy place has a proposition or attribute linked to that place.
- Fuzzification in transition level: a fuzzy transition for example is corresponding to a fuzzy production rule such as IF-THEN.

3. Related works

Recently, there have been several works on converting pragmatic models to formal models in order to evaluate nonfunctional requirement especially performance and reliability. Among formal models, the most active ones are extended version of Petri nets and queuing network (Balsamo & Marzolla, 2005). Through undertaken activities, Merseguer and Campos (2004) presented an approach with

high degree of functionality. The method is based on labeled generalized stochastic Petri net. It reviews the lower layer of performance standard presented by object management society and highlighted the role of UML diagrams in usability of software. In Balsamo and Marzolla (2005), the model describes the behavior of software architecture by using sequence diagram and finally extracts the queue network model from the Mentioned descriptions and software Architecture is evaluated by using scenario-based techniques. In addition to mentioned works, there are other works carried out in order to automatically convert annotated UML diagrams to types of petri nets. For example, Harounabadi (2011) converted simple UML diagrams to fuzzy extended UML diagrams and to model software performance, he created profile to use in fuzzy UML diagram and finally converted fuzzy UML diagram to fuzzy petri net software system evaluation. In Akbari et al. (2011) and Motameni et al. (2008), first fuzzy sequence diagram is converted to fuzzy petri net then an executable model based on fuzzy petri net is created for reliability evaluation. There are, however, very few works on Fuzzy logic and fuzzy rules to evaluate nonfunctional requirement especially performance and reliability from Vague and imprecise information. In fact, the difference between the present work with recent research is in applying fuzzy logic and fuzzy rules to evaluate performance and reliability.

4. The proposed method

4.1 Mapping of fuzzy rules to fuzzy colored Petri net

In proposed idea, the concept of fuzzification in fuzzy color petri net is used in two levels, first by using linguistic variables for tokens, second by using fuzzy rules for fuzzy inference for transitions. In order to map fuzzy rules to fuzzy color petri net we consider the following 6 stages,

Stage 1: For all of the input and output variables involved in the rules, we identify membership functions and linguistic variables. Between membership functions, we consider triangular membership function for each input and output variables involved in establishing the conditions for the occurrence.

Stage 2: For each rule, we will specify event conditions and linguistic variables involved in the rule. For example, the Table 1 demonstrates event conditions and several assumption laws.

Table 1
Specify Event conditions and linguistic variables

Rule	linguistic variable	Condition
Rule1	FuzzyIn1-1, FuzzyIn2-1	FuzzyIn2-1 and FuzzyIn1-1
Rule 2	FuzzyIn1-1 FuzzyIn2-2	FuzzyIn1-1 and FuzzyIn2-2
Rule 3	FuzzyIn2-1,FuzzyIn1-2	FuzzyIn1-2 and FuzzyIn2-1

Stage 3: For mapping each fuzzy system, we put initial place at the beginning of the mapping and then for each input variable we consider one place. Also for each value of the linguistic variables we put one place and one transition. At each transition, we identify conditions and domain of each place. Fig. 1 shows details of our proposed study.

Stage 4: For each rule, we put one place and one transition. We repeat each linguistic variable for the number of rules. Then we connected linguistic variables involved in creating in each rule to respective transition by arc. If we establish the correctness of conditions, the result is transferred to output respective place. Fig. 2 shows how to create rules.

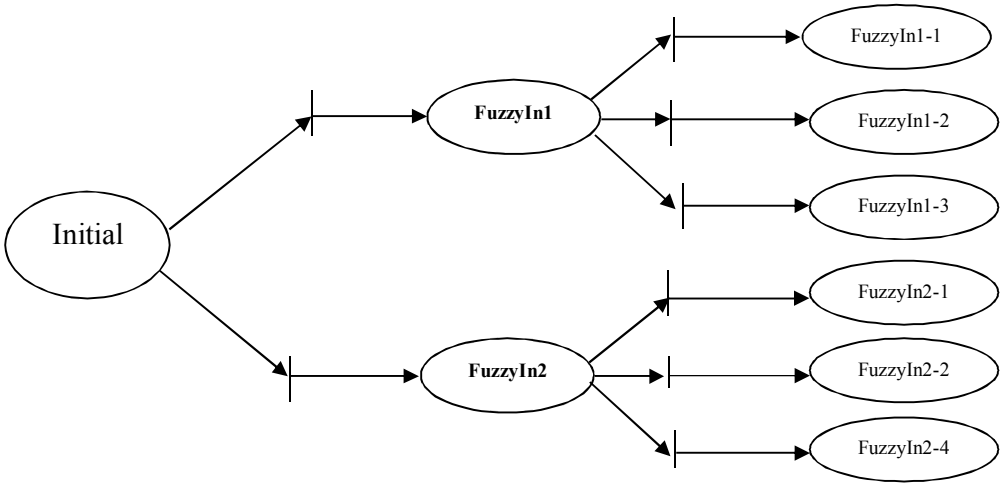

Fig. 1. Mapping input variables and membership functions to fuzzy color petri nets

Stage 5: In this stage the final output with crisp nature, leads from combining the results of places associated with respective rules. We use the "center of gravity" method for defuzzification the operations. In Eq. (1) the center of gravity method is shown. In this relation yi^{-} represents the maximum value of rule *i*. According to Fig. 3, we put one transition in order to calculate the final output in fuzzy color petri net.

$$y = \frac{y1^{-}\mu_{rule1} + y2^{-}\mu_{rule2} + y3^{-}\mu_{rule3}}{\mu_{rule1} + \mu_{rule2} + \mu_{rule3}} \quad (1)$$

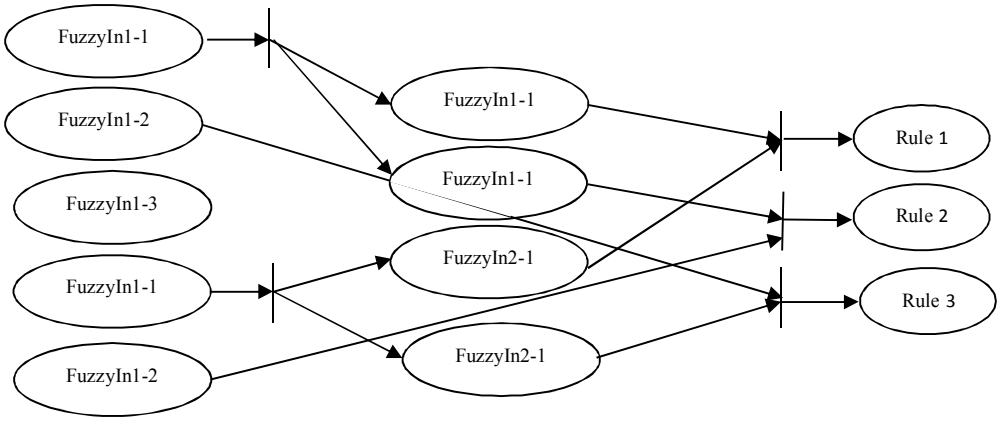

Fig. 2. How to create fuzzy rules in a fuzzy colored Petri net

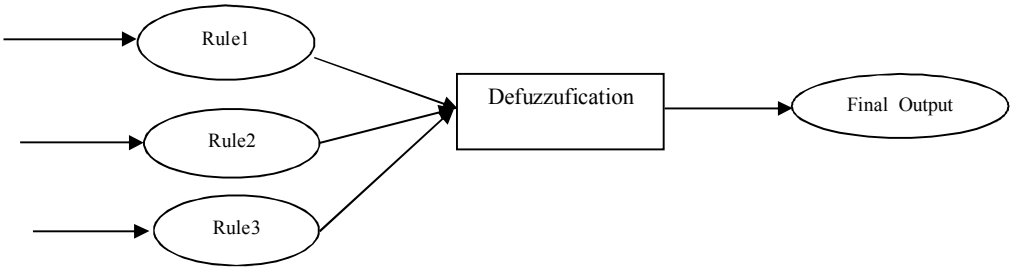

Fig. 3. Calculate the final output of the fuzzy rules

4.2 Evaluation Response time by using fuzzy performance stereotypes in sequence diagram

Response Time in sequence diagram means the execution time of scenario. It is assumed that messages in sequence diagrams are sent based on chronological order from the sender object to the receiver object in distributed networks. Hence, the time required sending and execution is affected by the following parameters.

1. Think Time: The time that system spends until the next request is sent by the user or objects.
2. Transition Time: The time required to send a message from the sender object to the receiver object. This time is under the influence of two parameters: message size and network band width.
3. Execution Time: Elapsed time for execution the method.

In other words, the time of each method is calculated according to Eq. (2) and response time according to Eq. (3).

Method Time= Think Time + Transition Time + Execution Time (2)
Response Time= Total time for all methods (3)

According to Eq. (2) and Eq. (3), linguistic variables involved in calculating response in a fuzzy system includes thinking time, message size, network bandwidth and execution time. Let FTT_i^l be the thinking time of model l in node i, FMS_i^l be the message size of model l in node i, FBW_i^l be the band width of model l in node i, and finally FET_i^l be the execution time of model l in node i. According to the following fuzzy rules, time of each method and response time of each sequence diagram is calculated.

- *The fuzzy rule of calculation method time*

IF FTT_i^l is FS1AndFMS_i^l is FS2AndFBW_i^l is FS3And FET_i^l is FS4 Then FMessageTime is FS5

- *The fuzzy rule of calculation response time of sequence diagram*

Sequence Diagram Response Time=μ(FMessageTimMethod**1**)+…+ μ(FMessageTimMethodn)

Four fuzzy listed linguistic variables in «PAStep» stereotypes are used by fuzzy tags PAThinkTime, PAMessageSize, PABandWidthNet, PAExecTime. Hence, in order to calculate the response time in sequence diagram we annotations each method with Mention «PAStep» stereotypes.

4.3 Evaluating reliability by using sequence and deployment fuzzy annotation diagram

In the proposed algorithm, the reliability of the entire system is considered a function of two parameters of failure rate of any component and failure rate of connector between components inspired by the proposed methods (Emadi & Shams, 2009; Cortellessa et al., 2002). For the proposed method, we assume all component consist and the components are distributed in existing nodes in deployment diagram. In time When there are two components in one node, we neglect the failure rate of connector failure and we consider that rate only when two components are in different nodes. Fir. 4 shows components distributed approach.

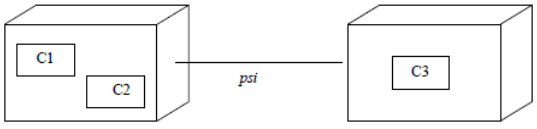

Fig. 4. components distributed approach

For the proposed algorithm, to evaluate reliability with fuzzy approach, each system can be considered as a collection of sequence diagrams, assuming that each use case diagram is shown with one sequence diagram. Hence, according to Eq. (4), the reliability of each method is calculated in a sequence diagram where θ_i^l is failure rate of method *i* and $\psi_{\text{Connectorid}}$ is failure rate of connector between sender and receiver object method. Also according to Eq. (5), reliability of the whole Sequence Diagram can be obtained from multiplying the reliability of each method. According to Eq. (6), the reliability of the whole system can be obtained. In the last equation, p_i is probability of using sequence diagram *i*.

$$\text{Method reliability}=(1-\theta_i^l)\ (1-\psi_{\text{Connectorid}}) \quad (4)$$

$$\text{Reliability of a Sequence Diagram}=\prod(1-\theta_i^l)\ (1-\psi_{\text{Connectorid}}) \quad (5)$$

$$\text{reliability of the whole system}=\sum_i^k \prod(1-\theta_i^l)\ (1-\psi_{\text{Connectorid}})\, p_i \quad (6)$$

In fuzzy system, the failure rate of each method and the failure rate which is the connector between the transmitter and the receiver object are expressed by linguistic variables, Assuming that failure rate of method l in sequence diagram i is θfm_i^l and failure rate of connector between the transmitter and the receiver object is $\psi\text{fc}_{\text{Connectorid}}$. Hence, the reliability and the sequence diagram reliability for the whole system reliability are calculated based on following fuzzy rules.

- *The fuzzy rule of calculation reliability of method*

  IF $\mu(\theta\text{fm}_i^l)$ is FS1 And $\mu(\psi\text{ffc}_{\text{Connectorid}})$ is FS2 THEN Method Reliability is FS3
- *The fuzzy rule of calculation reliability of sequence diagram*

  IF μ(Method Reliability1) is FS1And …And μ (Method Reliability n) is FSn THEN Sequence Diagram Reliability is FSx
- *The fuzzy rule of calculation reliability of whole system*

  IF μ(SequenceDiagramRelability1) is FS1And …And μ (Sequence Diagram Reliability n) is FSn THEN SystemReliability is Min(μ(SequenceDiagramRelability1),…, μ (Sequence Diagram Reliability n))

In order to write annotation of the failure rate of each method in sequence diagram, we use «Recomponent» stereotype with Recompfailprob tag with linguistic variable Nature, Also to write annotation of the failure rate of each connector between the transmitter and the receiver object we use «REconnector» stereotype with REconnfaiprob tag with linguistic variable Nature. To determine component of Transmitting method we use REcomponentSenderName tag and to determine component of Receiver method we add REcomponentReciverName tag to «REcomponent» stereotype. To write annotation for each node in deployment diagram we use «REhost» stereotype with REindexhosttag and also to annotation each connector link we use <<REconnector>> stereotype with REconnfaiprob tag.

4.4 Mapping performance and reliability annotation to fuzzy color petri net in sequence diagram

Sequence diagram include set of methods. Each sender and receiver component of the message in sequence diagram turns to place and messages, which are the connectors between these two components turn to transition. Fig. 5 shows mapping message in sequence diagram to color petri net. In order to map fuzzy performance and reliability annotation to fuzzy color petri net, we put two transitions for entry to evaluation performance and reliability fuzzy subnet in related receiver component. We next calculate the time and the reliability of sending the information based on pattern according. At the end in order to calculate response time of sequence diagram we consider total time of methods and in order to calculate the reliability of the sequence diagram of the diagram we consider the minimum reliability. In Fig. 7, the way to calculate response time and reliability of sequence diagram is shown. The transition associated with the component to message receiver enters

in two subnet based on fuzzy color petri net by arc in order to evaluate the message time and reliability of each method. In Figure 6 the fuzzy subnets related to evaluation performance and reliability of each method is shown.

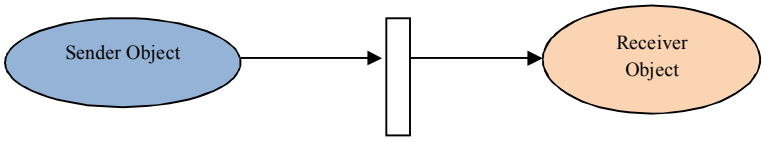

Fig. 5. Mapping message in sequence diagram to color petri net

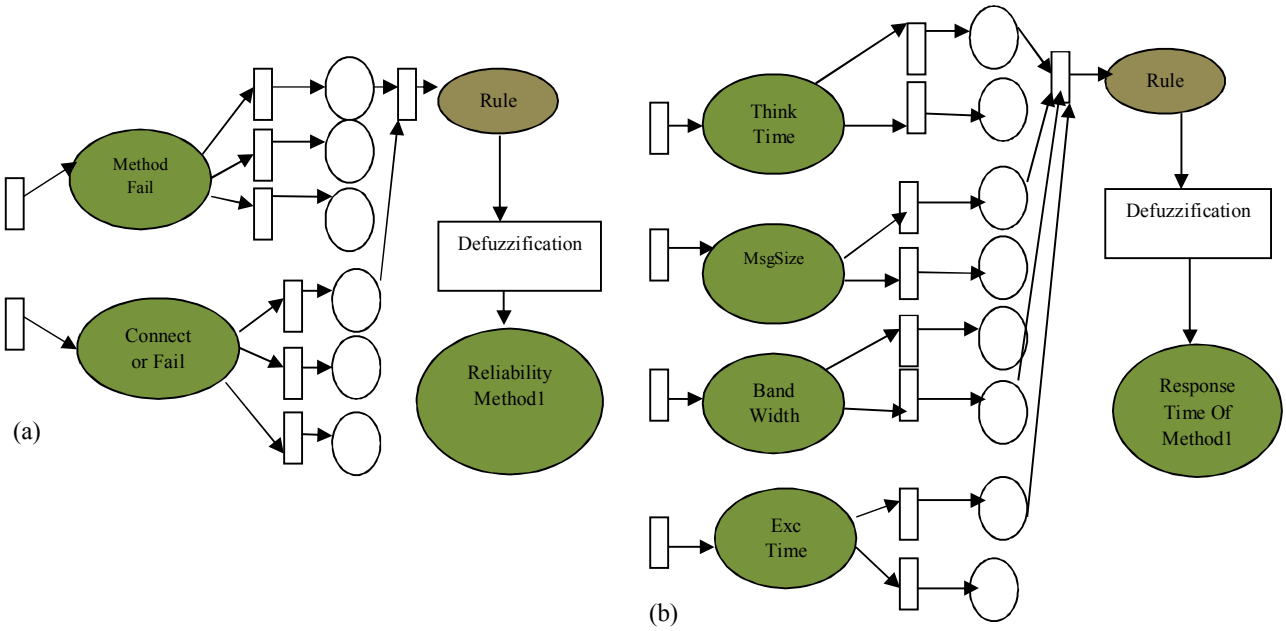

Fig. 6. Fuzzy subnets: (a) Fuzzy subnet for calculate reliability, (b) Fuzzy subnet for calculate response time

5. Case study

The case study of this paper is assigned to the facility department in a hypothetical bank shown in Fig. 8 where part of the sequence diagram with fuzzy annotation related to response time is shown. In addition, Fig. 9 shows the deployment diagram with fuzzy annotation of fuzzy reliability and Fig. 10 shows part of sequence diagram with fuzzy annotation of fuzzy reliability. In this case study, we have implemented the proposed algorithm on the sequence diagram of the loan process.

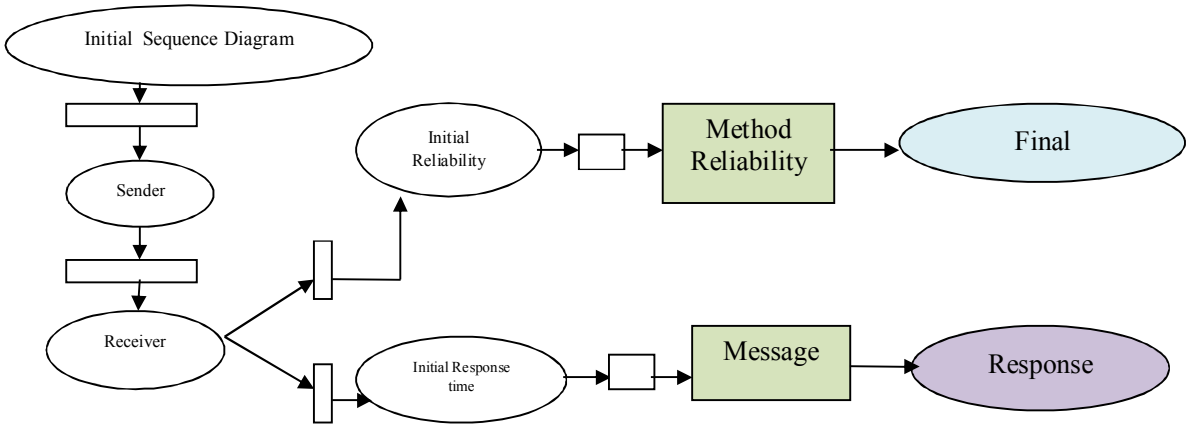

Fig. 7. Calculation response time and reliability in sequence diagram

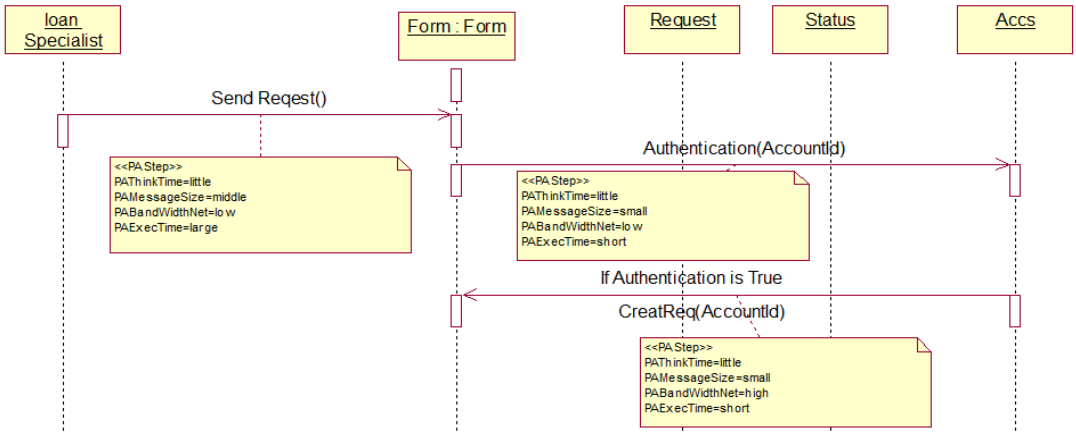

Fig. 8. Part of sequence diagrams with fuzzy annotation of response time in case study

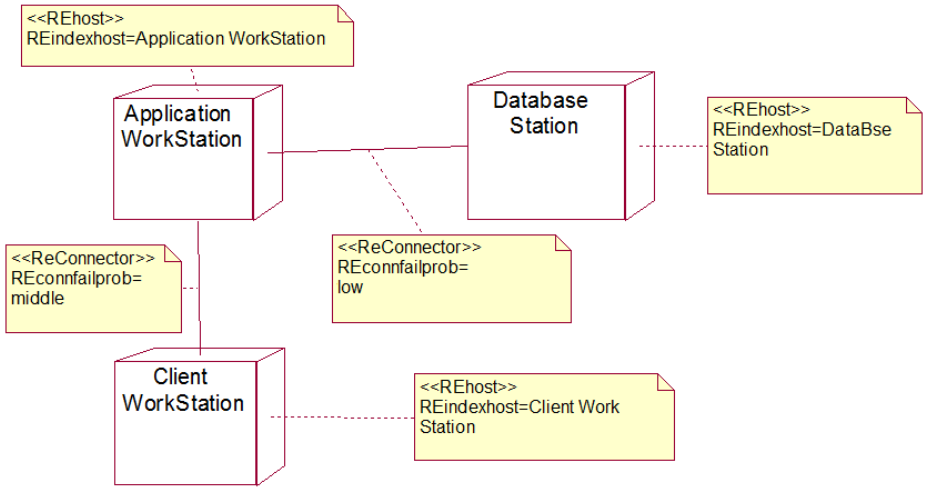

Fig. 9. Deployment diagram with annotations of fuzzy reliability in the Case Study

Fig. 11 shows the procedure on how to insert the use case related to sequence diagram of loan payment and workload entries to system. In this figure, tokens are generated with exponential distribution and stay in QUEUE place online order. Dis transition fires if there is a token in place QUEUE and also the time response of previous tokens is calculated and its work has been completed. Hence, when a token is fired from Dis transition, it enters the Sequence Diagram subnet. In this subnet, response time and reliability is calculated in fuzzy state. Since the time delay of the token, which remained in the queue to enter the sequence diagram, is effective on response time, then the Time delay Places in Delay place, its value sums with the value of obtained response time and it is considered as the final response time. In Fig.12, according to proposed algorithm in section 4.4, fuzzy performance and reliability annotation in sequence diagram of the loan process are mapped to fuzzy color petri net. In this figure with two fuzzy subnet of ResponseTimeM and ReliabilityM, response time and the reliability of each method are calculated. The Implementation tool for drawing the UML diagrams is Rational Rose (Ver7) and to simulate the fuzzy colored Petri net we used CPN Tools.

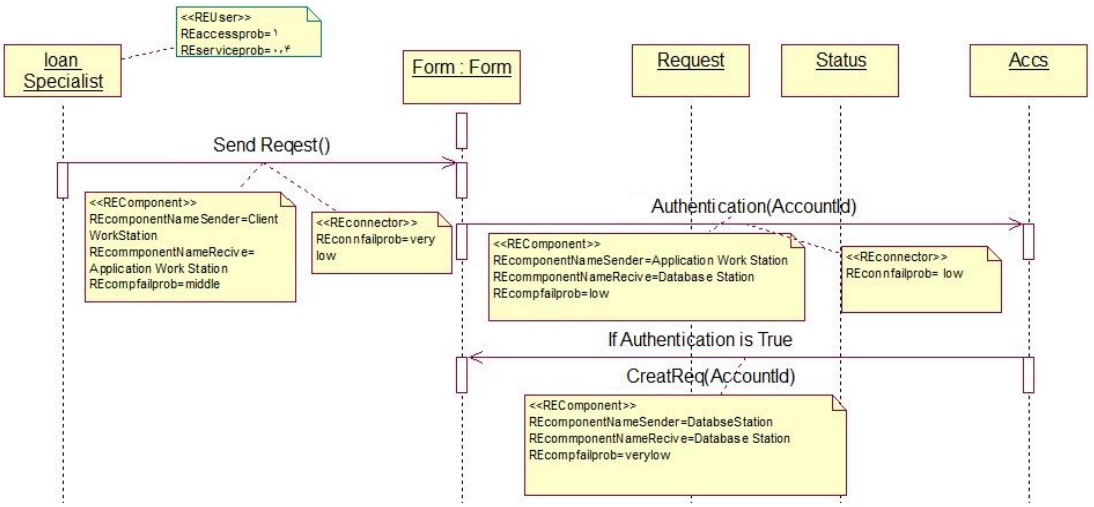

Fig.10. Part of sequence diagrams with fuzzy annotation for calculation reliability for the case study

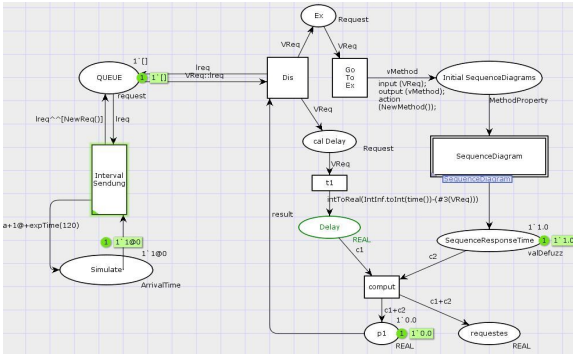

Fig. 11. The mapping of use case and work load

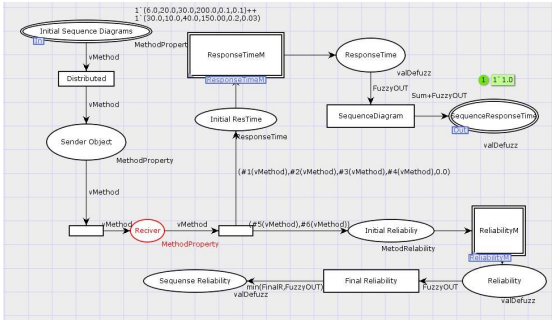

Fig. 12. Mapping patterns of each method in sequence diagram

6. Conclusions

In this paper, a new mechanism to evaluate the performance and the reliability of software systems has been developed. The proposed formal fuzzy color petri net based on pragmatic model (UML) has allowed us to develop systems with software process models. The research results show that the proposed method has the capability of supporting different criteria according Table 2.

Table 2
Supporting criterions of proposed method

Metric	Supporting level
Hierarchical definition of software architecture	yes
Level of flexibility in order to modify or extend the architecture	high
Creating an executable model of the architecture	yes
To obtain model from UML diagrams	yes
To obtain model from describe the behavior	yes

References

Akbarian, E., Noorian Talooki, H., & Motameni, H. (2011). Mapping sequence diagram in fuzzy UML to fuzzy Petri Net. *Iranian Journal of Optimization*, *2*(3), 492-505.

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.

Cortellessa, V., Singh, H., & Cukic, B.(2002). Early reliability assessment of UML based software models. In Proc of IEEE Workshop on Software and Performance (WOSP02), (PP.302- 309).

Emadi,S.,& Shams, F.(2009). Mapping use case and sequence diagrams to a petri net notation for performance evaluation. Second International conference on computer and Electrical Engineering, (PP.68-71), Shahid Behehshti University, Tehran, Iran.

Harounabadi, A. (2011). Modeling and evaluation of information systems by using fuzzy color petri Nets. 16th national conference on computer society of Iran,(pp.187-193, Sharif University, Tehran, Iran.

Object Management Group. (2002). UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms Specification.(2002), From www.omg.org. Accessed March 11,2014.

Merseguer, J., & Campos, J. (2004). Software performance modeling using UML and petri net. *Lecture notes in computer science*, 2965(1), 265-289.

Motameni, H., Movaghar, A., Daneshfar, I., Nemat Zadeh, H., & Bakhshi, J. (2008). Mapping to convert activity diagram in fuzzy UML to fuzzy Petri Net.*World Applied Sciences Journal*, *3*(3), 514-521.