Performance Evaluation of Service-Oriented Architecture using Generalized Stochastic Petri Net

Ali Ghasemi  
Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Bushehr, Iran

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

Seyed Javad Mirabedini  
Department of Computer Engineering, Tehran Center Branch, Islamic Azad University, Tehran, Iran

ABSTRACT
Service-oriented architecture (SOA) is a software architecture design pattern based on discrete pieces of software providing application functionality as services to other applications. Before application of SOA, if a model of problem is drawn and analyzed, then possible flaws in architecture implementation phase could be prevented. This paper presents a method for evaluating SOA based on formal models. To achieve this goal, products of C4ISR framework are calculated, then are marked on UML diagrams to get the real model. For this purpose, the UML elements should be indexed according to efficiency. Following that the UML elements based on presented algorithm are transformed to GSPN. The results suggest that the ATM is done on a case study demonstrates that modeling and evaluation in the design phase prevents the high cost of implementation phase.

Keywords

1. INTRODUCTION
SOA has emerged as a new paradigm for modeling the process which allows efficient development of organizational applications in which problems related to the efficiency of software development has always been an important aspect in the process [1]. Despite advantages such as flexibility and reuse of SOA, performance and accessibility of SOA applications cannot simply be predicted [3]. The ability to assess the performance of SOA applications with different workloads to predict undesirable behaviors is of great importance. Typically, application designers create a program and insist on it to evaluate its metrics of efficiency. This approach can be time consuming and costly, since the performance analyst needs an executable version of the system and the limited computational resources of a company to analyze the system. Thus, accepting an analytical modeling approaches can reduce these costs [2]. Thus far, many methods have been proposed for the evaluation of SOA, but most of them have taken place at the level of implementations. Few of them deals with the evaluation of the model on which we focus in this paper using a formal model of SOA evaluation methods. Although the UML has become widely available nowadays, there are a lot of gaps in the evaluation of software systems in UML since UML does not have a formal model, it is not possible to evaluate it directly [4]. In 2002, OMG which is in charge of the developing UML introduced UML profile to support efficiency concepts. However, in order to assess the performance, the real model should be converted into a formal model [10]. In section II of the paper, the context of the work will be considered. This section emphasizes on GSPN. In section III, studies done in line with subject of the article are examined. The proposed solution in providing a formal model and how to analyze it will be considered in section IV. Section V presents a case study and Section VI is devoted to conclusions.

2. BACKGROUND
Doing the implementation consists of the UML, Petri nets and GSPN, due to the importance of the latter two, they are briefly reviewed.

2.1 Petri net
Petri nets are a powerful tool for modeling concurrency and provide stronger description than line networks. In addition to formal structure and behavior, Petri nets are capable of displaying graphics which makes modeling easier. One of the reasons for the success of Petri nets is its simplicity which in modeling complex systems makes it difficult [5].

2.2 Generalized Stochastic Petri Net
Models of GSPN are composed of two types of transfer: A) scheduled transfers that are associated with activation delays profile like randomly distributed such as Petri nets. B) Immediate transfers at zero time and with priority over enabled timed transitions. In addition, arcs are allowed, multiple priority levels, immediate transfers can be used, and Weights are associated with immediate transitions [6].

3. RELATED WORKS
Today, numerous research has been done on converting a pragmatic model the formal model. In [7] ASIM bases of assessment services that are obtained from the matrix CRUD and using the metric CRUD matrix is computed. CRUD matrix is a matrix composed the rows of the underlying business processes, the basic pillars of the business entity, and its houses are made of semantic relations (C, R, U, D). In [8] provides a method of evaluation for service oriented architecture, so that it provides developments of the alignment of business and information technology. The method presented in [8], participation requirements nonfunctional services in IT-based systems can be quantitatively evaluated. In [8] the assessment, rather than a single value, the mean Variance defines the normal distribution made into a model. The usefulness of the normal distribution is that it is much closer to the actual value. In [9] it is a description of the development of the PCM nonfunctional features based on limited logic programming hierarchy. The method presented in [9] is a complete and simple method for the description of Web services based on the needs of the nonfunctional framework assessment and a description of the nonfunctional needs. In this paper, uses a simulation modeling approach to evaluate the performance of SOA applications, based on GSPN.
4. THE PROPOSED METHOD

Using the model proposed by Levis C4ISR products are converted into UML and then UML diagrams are converted to GSPN to evaluate SOA. In this study, UML is used to describe the use of SOA diagrams, and to place and settle the activities. And according to the purpose of evaluation using SOA is GSPN. At first outputs of SOA must be converted to GSPN, and then begin to evaluate of SOA. The purpose of this paper is to provide a formal model of UML diagrams and then analyze it before its implementation.

4.1 Using GSPN for Performance Model

Profiles in UML provide particular domains platforms to develop a general mechanism for customizing UML models. In other words, sets of Stereotypes, Tagged Values, and Constraints form a UML profile. To create GSPN model, UML diagrams that include performance annotation are considered as input. The diagram in use, shows the system workload. Deployed diagrams provide a plan of physical resources of the system, and the chart of activities shows requests of service for different activities.

4.2 The Proposed Algorithm

Petri nets are used for qualitative analysis of logical properties of systems. GSPNs are Petri nets that delay in firing transitions is identified by random variables with profile distribution. In these nets for each transition one delay in firing is randomly attributed which its probability density function is a negative exponential with rate \( \lambda \).

4.2.1 Algorithms for converting UML modeling of GSPN into C4ISR products

Step 1: Mapping C4ISR products to UML diagrams

Step 2: Annotating performance characteristics using performance profile.

Step 3: Converting UML diagrams with annotated the performance to GSPN.

Step 4: Analyzing of GSPN to obtain the performance results.

4.2.2 Conversion algorithm (high level view)

Input: annotate UML model with the performance (use case diagram, activity diagram, deployment diagram)

Step 1: Convert each activity diagram component to GSPN. For each activity diagram:

Step 1-1: Converting activity diagram to a GSPN

Step 1-2: Forming GSPN using GSPN components

Step 2: Forming GSPN system by combining all the components of GSPN

Output: System GSPN.

4.3 Mapping Activity Diagram to GSPN

Activity diagrams is used to describe the components of business and operational workflow step by step in the system. The basic computational step is called an action. To a group of actions and sub-activities are called activity node.

4.3.1 Action and control flow

For each activity in the activity diagram a Petri place is considered. In marking GSPN, a token is placed corresponding to the activity diagram, this means that an instance of this process is running an action. For each action inflows, the transition should be indicating the start of action. Similarly, transition of a token from the arc corresponds to a flow, which means that the edge of the activity is currently active. Therefore, when the transition is active, then the edge of activity is active, running action with a fire starts and takes the edge off activation. Similarly, the input arc from the place of action, and the outgoing arcs in the place of edge, transition should be indicative of the end of action (see Figure 1).

Fig 1: Mapping Control Flow of UML to GSPN

The first transition represents entering into the act, a token in place represents the state while performing the action, and the second transition represents a departure from the action. Then input transition of the action must be connected to the place by the input arc indicating the action edge activity and output transition of the action must be connected to the place by the output arc indicating the action edge end (see Figure 2).

Fig 2: Mapping a Action of UML to GSPN

In an action may be a certain amount of work require some sources. As a result, execution may be dependent on the availability of resources. Then two cases may take place in one having an open source, and is linked to the resources needed in other state sources are not available which causes the delay. Only possible if there is a link from/to host provided that the host is associated with a request for service and its characteristics is specified. Also according to the probability distribution function for a transition when the time delay is used especially when the service is busy. This behavioral model is as follows, (see Figure 3).

Fig 3: Activity Related to Resources

4.3.2 Initial and flow final nodes

In GSPN, mark of activation indicate the inflow edge and it should be eliminated by flow end nodes (see Figure 4).

Fig 4: Mapping a Flow Final Node of UML to GSPN

There are at least two ways to interpret an initial node. This marked place ascertains that the transition can fire only once, as the token will be removed upon the firing, rendering the transition disabled from that time on. The other interpretation can be achieved by simply omitting this place and the arc (see Figure 5).

Fig 5: Mapping an Initial Node of UML to GSPN

4.3.3 Join and Fork

Direct show of GSPN is equal of Fork node to an immediate transition, with input and outgoing arcs which hierarchically represent inflow and outflow edges (see Figure 6).

Fig 6: Mapping a Fork Node of UML to GSPN
Mapping join node to GSPN is similar to the fork node: an immediate transition with input and outgoing arcs to the place; they indicate the edge of inflows and outflows in order (see Figure 7).

**4.3.4 Decision and Merge**

To indicate the decision in GSPN form a timed transition for each outflow, and determine a probability to each timed transition by means of exponential distribution. The total of all the timed probabilities must be 1 (see Figure 8).

For merge mapping to GSPN make an immediate transition for each inflow and match a timed transition for outflow in order to have an action like a coordinator (see Figure 9).

**4.4 Performance Evaluation of SOA Using GSPN**

In closed Petri Net each actor is Annotate with “PAclosedLoad” expressing the time spent between end of an interaction with system and beginning of its next repetition. In fact each load actor indicates a different work. In such occasion, GSPN model can disintegrate into some sub-models with a different responsibility. Each model requests could have many classes. Open Petri Net is Annotate by “PAopenLoad” in use case diagram. Using the activity diagram help us to obtain the transition probability and routing. It define these features to support this issue in GSPN model. Assuming that system source is represented by RES = {res₁, res₂, ..., resₙ}, for each res ∈ RES source a feature with the name of total[res] would be defined. total[res] represents total actions in which service apply to res. To identify sources it define a unique index with the help of prefix “id” like the following format [2]:

\[
id(\text{res}_i) = i \text{ for each } \text{res}_i \in \text{RES}
\]

Total actions of res source are represented by ACT= {act₁, act₂, ..., actₙ}. Obviously it will have: total[RES] = m. for each res∈RES source label, all the actions in \{act∈ACT|resource(act)=res\} has a unique number in [1,2,...,total[RES]]. This unique number is known by the adjective mark[act]. It define an adjective of mark[act] for each act∈ACT action. Open Petri Net input enters a transition which was used in the first action of diagram. Input rate of GSPN is equal to Annotate “PAoccurrence” in the actor which used activity diagram to verify use case. The probability of transition GSPN, the label corresponding PAprob will be resulted in transfer activity diagram. Model conversion algorithm of UML to GSPN of performance evaluation of software is as follows (assumption: x actor Annotate by “PAopenLoad”) [2]:

a) Specify the following attribute rates

\[
total[\text{res}] \neq \text{res} \in \text{RES}
\]

\[
\text{mark}[\text{act}] \neq \text{act} \in \text{ACT}
\]

b) Calculate the transmission probability matrix.

c) Calculate the service rate to r class client in the i transmission. (m is an action in activity diagram.)

\[
\text{rate}(i) = \frac{\text{PArate}(\text{res})}{\text{PADemand}(\text{act})} \text{ where } i=\text{id(\text{resource(m)}), r = \text{mark}[\text{act}]}
\]

d) Define λ(r) vector representing the arrival rate for customers of class r, as follows:

\[
\lambda(r) = 1/\text{PAoccurrence}(\text{x})
\]

In other words the arrival rate is calculated according to the use case (tages PAoccurrence), which had lead to the use of an activity diagram. If an actor X is Annotate by the stereotype “PAclosedLoad” border, the following modifications to the above algorithm is applied:

a) Number of transfer is one more than available resources on the deployment diagram.

\[
\text{T} = |\text{RES}|+1
\]

b) The added transfer is shown by zero and as mentioned before it’s used for handling the delay. The transfer rate of firing for all the customer classes is as follows. (s is one the customer classes.)

\[
\text{rate}(s) = 1/\text{PAextDelay}(\text{X})
\]

c) N is the frequency of recorded requests in system and desired actor is calculated according to the Annotate PApopulation. Above mark is illustrated by population:

\[
\text{N} = \text{PApopulation}(\text{X})
\]

A complete algorithm is obtained by combining the above two algorithms in which number of GSPN sub-models are A+B. A and B are the actor numbers with stereotype “PAopenLoad” and “PAclosedLoad”.

5. EVALUATION AND IMPLEMENTATION OF THE PROPOSED METHOD

A case study devoted to the ATM to withdraw money from an account at a bank. Use case diagram reveals the scrutiny scenarios with the help of performance analysis. It can reveal the operating rate of each actor and workload of system with use case diagram. In (figure 10) assumed the use diagram with open workload. For this purpose it use stereotype “PAopenLoad” to mark the actor, PAoccurrence for arrival rate of requests.
Fig 10: Use Case Diagram and Open Workload Modeling
Activity diagram displays the scenario of dynamic perspective. This diagram represents a service demand from sources for different activities. It’s used for scenario modeling and software behavior. The diagram consists of all the actions of system (see figure 11).

Fig 11: A Action form Activity Diagram with Annotate Performance
Deployment diagram provides a system design of physical resources and it indicates the moved resources. PArate indicated the relative speed factor for source CPU as a percentage or part of standard CPU. PAeschdpolicy is the time policy of CPU for giving the source. PAutilization displays efficiency of the source (see figure 12).

For analyzing this model the following method was performed. Assuming the related use case to activity has Annotate "PAoccurrence= (exponential, 8)" then:

\[ \lambda = 1/8 \Rightarrow \lambda = 0.125 \]  

(9)

Tag PAdemand statistics for the action using ATM client source are as follow:

\[ PA_{demand}(act) = \{2.0, 1.0, 3.0, 2.0, 4.0, 1.0\} \]  

(10)

Due to the ATMclient, source value is PArate(res) = 2.0, the fire value for ATM client source in GSPN transition is:

\[ rate(1) = \{1.0, 2.0, 0.6, 1.0, 0.5, 2.0\} \]  

(11)

Fig 12: deployment diagram and Annotate performance
Now, according to the procedure has been described in section IV, UML activity diagrams mapping to GSPN model. Convert each activity diagram to GSPN, suppose a GSPN for each activity diagram, and change each component of the activity diagram to GSPN components. At last phase, all components with GSPNs are connected and combined together and the final GSPN system can be produced (see Figure 13).

Also, it use PAdemand for the actions using ATMs server and its value is PA_{demand}(act) = \{6.0, 5.0\}. Because the source value is PArate (ATMserver) = 1.0, the fire value in GSPN transition for ATM source is:

\[ rate(2) = \frac{PArate(res)}{PA_{demand}(act)} = \{0.1, 0.2\} \]  

(12)

Mark PA demand value in actions using Database Server are rate(res) = 3.0. Because the value of Database Server is:

\[ rate(3) = \frac{PArate(res)}{PA_{demand}(act)} = \{0.6, 0.3\} \]  

(13)

With specific arrival rate \( \lambda \) and the rate of fire (service), it can attempt to analyze GSPN. Two meters Utilization, throughput has been emphasized. By measuring the source activity time...
and actions messaging time, the Utilization can be gained. Figure (14) demonstrates the Utilization of three resources, i.e. DatabaseServer, ATMServer and ATMClient. Y-axis represents the Utilization and x-axis represents the time in terms of microseconds; the Utilization will be demonstrated based on the actions activity and messaging period.

Throughput is the average exchange rate associated with the source which is one of the important meters in performance evaluation. As figure (15) demonstrates, three sources ATMClient, ATMServer, DatabaseServer throughput has been studied. X-axis represents time in terms of microsecond and y-axis represents the throughput; the throughput can be calculated by means of the rate of message exchange between resources and actions.

In (Table 1) the throughput and utilization of the proposed method with the results presented in [11] has been compared in order to evaluate the performance of the proposed method that by the symbol PM it has been shown.

Table 1. Result of the Proposed Method Compared to Marzolla Method

<table>
<thead>
<tr>
<th>Resource</th>
<th>PM Utilization</th>
<th>Marzolla Utilization</th>
<th>PM Throughput</th>
<th>Marzolla Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATMClient</td>
<td>57.46117</td>
<td>56.53456</td>
<td>29.31923%</td>
<td>28.26733%</td>
</tr>
<tr>
<td>ATMServer</td>
<td>38.01553</td>
<td>35.93136</td>
<td>18.52199%</td>
<td>17.96572%</td>
</tr>
<tr>
<td>DatabaseServer</td>
<td>48.13057</td>
<td>46.87162</td>
<td>25.20743%</td>
<td>23.43587%</td>
</tr>
</tbody>
</table>

Below numbers are the mean of ten Utilization assessment repetition and throughput by CASE tools. They also represent the better Utilization and throughput of proposed method than Marzolla method. Thus, the architecture can easily be evaluated before implementation of architecture with the help of the proposed model. In case of not meeting the quality attributes, the architecture design can be revised so that its quality will be improved. Ensuring the hardware configuration of system in the early phases of system development prior to implementation reduces the cost of implementation. Reviewing the feedback and analyzing the model allow the changes to be done on the model and system to be implemented after desired responses are met.

6. CONCLUSION

In this paper, a new algorithm has been proposed for converting actual model to formal model to evaluate the performance of software systems. Proposing a formal model GSPN based on an actual model UML makes it possible not only to develop system along with software process model, but also to handle non-functional requirements of software. Simulation results help us to evaluate the model before implementation of system. The results suggest that the ATM is done on a case study demonstrated that modeling and evaluation in the design phase prevents the high cost of implementation phase.

7. REFERENCES