International Journal of Computer Applications Technology and Research
Volume 4– Issue 11, 865 - 870, 2015, ISSN: 2319–8656

A framework for Performance Prediction of Service-Oriented Architecture

Haitham A.Moniem,
Department of Information Technology,
College of Information Technology,
Future University
Khartoum, Sudan

Hany H Ammar,
Lane Department of Computer Science and Electrical Engineering,
College of Engineering and Mineral Resources,
West Virginia University
Morgantown, USA

Abstract: Service-Oriented Architecture (SOA) considered as one of the important architectural styles to build future applications. This architecture consists of a group of homogenous and autonomous components that interact with each other to accomplish a task. However, performance prediction of SOA based applications is regarded one of the complex tasks that face software developers and engineers. This paper presents a novel approach for SOA performance prediction at early stages of SDLC by using Machine Learning technique. Firstly, annotated UML diagrams are presented. Secondly, translate the UML diagrams into (QNM) model in order to extract performance indices such as Response Time, Throughput, and Utilization. Finally, machine learning technique used to predict the application model performance. The prediction result "Risk" means design does not meet customer requirement and "No Risk" means the design satisfies the customer requirements. Machine learning technique predicts the performance based on the training set against the extracted test set of the application model performance indices. The new method has many advantages, such as reducing time, scale with large system size, and avoiding problems before the service put into the production environment. To illustrate our approach, we present the results of a simple practical example.

Keywords: SOA; Queuing Network Model; Machine Learning; Performance; Prediction; Architecture

1. INTRODUCTION

Service-Oriented Architecture (SOA) provides excellent features for designing distributed internet applications include reusability, flexible configuration, and easy implementation. The Service defined as functional logical part that performs a unique business task. This task could be for consumer or for another service connected to it. Many challenges come with SOA as a new architecture style. Firstly, challenges related to finding methods to analyze and predict the Quality of Services (QoS) such as security, performance, service availability, and standardization during designing time. Secondly, challenges related to applying QoS standers on SOA based applications such as centralization and service integrity in environment exhibits autonomous, encapsulation, and privacy [1].

The proposed approach considers the report "Risk" if the performance indices extracted from the architectural design does not meet the customer requirements and "No Risk" if the performance indices satisfy the customer requirements. We assume that customer has non-functional requirements such as response time, resource utilization, and throughout. According to ISO 9126 all the previous requirements called Performance [9].

Generally, there are two types of software performance prediction, at the design time (Model-based) and at running time. Our proposed approach concerns about model-based performance prediction from the initial stages of Software Development Life Cycle (SDLC). Model based performance prediction has many advantages such as low cost, easy, and practical.

The proposed framework consists of three major steps as presented in Fig.1.

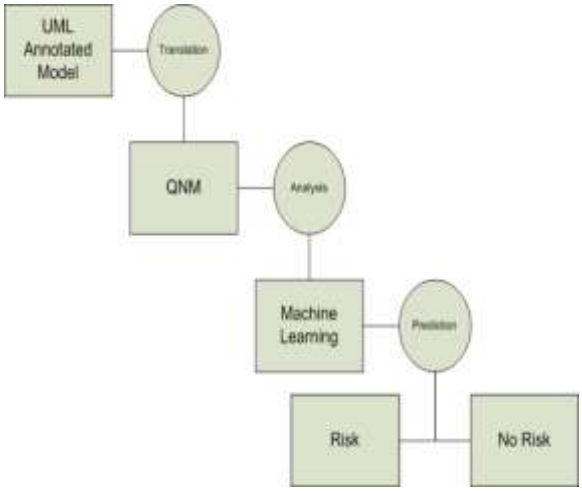
First step: annotated UML diagrams will be used to describe software system as follows:

- Use Case diagrams represent workloads applied to the system [7].
- Deployment diagrams describe available physical resources where computations take place.
- Activity diagrams describe both the order in which resources are used, and corresponding service demand.

Second step: mapping annotated UML model into Queuing Network Model (QNM) to simulate the SOA application and generates performance indices.

Final step: take the output of the Queuing Network Model (QNM) as a new instance and provides it to the machine learning, then based on the training set the machine learning algorithm will predict whether there will be a risk or no risk if we implement the SOA application by current performance indices.

The reminder of this paper structured as follows section 2 presents the related work. Section 3 explains the term SOA. Section 4 demonstrates SPT UML profile (Schedule, Performance, and Timing). Section 5 states the translation from UML to QNM. Section 6 defines QNM. Section 7 presents the machine learning technique. Finally, section 8 presents the case study.



Figurer 1. The Proposed Framework

2. RELATED WORK

[Ganapathi, 2009] proposed a statistical machine learning technique to predict and optimize multi components and parallel system utilization and performance. The proposed technique extracts correlation between a workload's pre-execution characteristics or configuration parameters, and post-execution performance observations [2]. The correlation has been used for performance prediction and optimization.

[Dubach, 2009] used machine-learning to efficiently explore the compiler architecture design. The researcher firstly, developed two performance models and used them to increase efficiency of searching the design space of micro-architecture [3]. These models accurately predict performance metrics such as cycles or energy, or a tradeoff of the two.

[Malhotra et. al, 2012] have employed machine learning to measure the maintainability, number of CHANGE is observed over a period of three years on dataset [4]. Change can be defined as the number of lines of code which were added, deleted or modified.

[Mohanty et. al, 2012] have employed machine learning technique to classify and rank web services [5]. The researchers proved that by using Naïve based Bayesian network the classification performs better than other techniques.

[Ipek et. al, 2005] used multilayer neural networks trained on input data from executions on target platform. The approach is useful for predicting many aspects of performance, and it capture full system complexity. The study focuses on the high performance, parallel application SMG2000 [6]. The model has predicted performance within error 5% - 7% error across a large, multidimensional parameter space.

3. SERVICE-ORIENTED ARCHITECURE (SOA)

Service-oriented architecture (SOA) is an architectural style where a system comprises of three major components: First, Service Provider is the service or entity that accepts and executes request from service user and service registry. Second, Service User is an application or service that requires a service. Third, Service Registry is a network based directory that contains available services [8, 9]. An architectural style characterizes the types of components, connectors, and configuration. Fig 2 explores SOA architecture.

Web service technology is a type of the implementations of SOA. Web Service consists of many published standards such as Service Oriented Architecture Protocol (SOAP) and Web Service Description Language (WSDL).

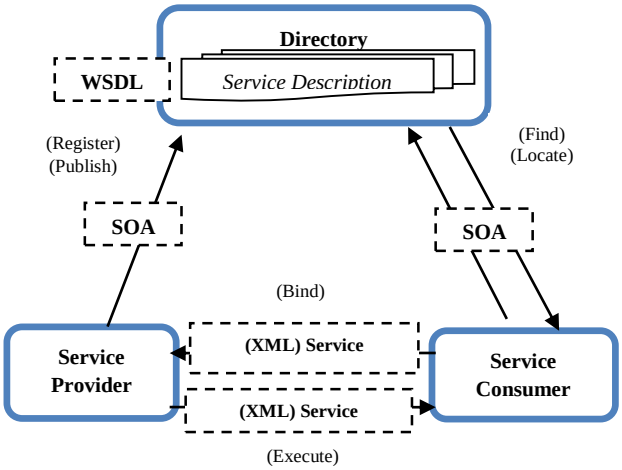


Figure 2. Service-Oriented Architecture

4. UML SPT PROFILE

UML profile for scheduling, performance, and Time has been released by OMG standard. The basic objectives of the UML profile are to declare the requirements for system performance and a tool to model physical time, timing specifications, timing services, logical and physical resources, concurrency and scheduling, software and hardware infrastructure, and their mapping [10].

UML profile provides an extension of the UML standard with special modeling components and their semantics. The main components of profile are new Stereotypes. Stereotype provides a style of extending UML by declaring simple terms and using them to explain UML components and cooperation in a system. Stereotypes are implemented to existing UML entities, such as class and association, and increase semantics of these elements with newly predefined meaning.

5. UML TRANSLATION INTO QNM

Based on the roles stated by (Simonetta et. al, 2004), we use algorithm named UML-QNE to translate an annotated UML diagrams specification into QNM. UML model components are translated into the corresponding QNM model elements as follows. From actors in Use Case diagrams we identify the type of QNM model. From activity diagrams we derive the network topology that is the behavior of the classes of users circulating through the system. Finally, each node in the Deployment diagrams defines a service center. The mapping between UML and performance model element is illustrated in Fig.3.

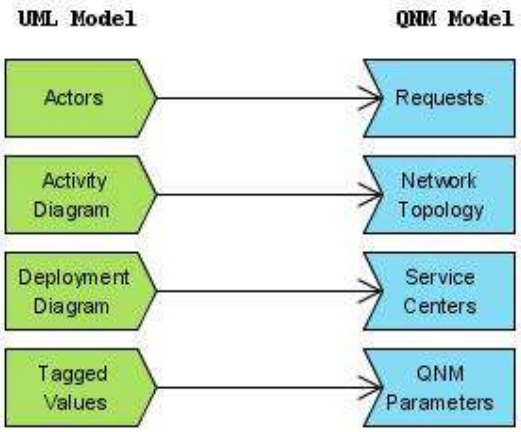


Figure 3: Mapping UML into QNM

6. QUEUINGNETWORK MODEL (QNM)

Queuing network model is a mathematical model used in computer systems performance analysis to predict quality attributes of a system and its parts [11]. The basic parts of QNM are Queues and Service Center. A queue is a buffer, which similar to any queuing system. A service center provides services to the queue's customer. Each service center has an associated queue containing jobs to be processed by that service center.

To get performance characteristics for any individual service center, two kinds of information must be presented. Firstly, the average rate ***R*** at which new jobs arrive in service center. Secondly, the average time taken by service center ***S*** to perform one job. Based on this information the following quality of services could be calculated:

- ***Utilization of each Service Center*** $(U_i) = R * S_i$
- ***Average Response Time*** $(Ri) = S_i/(1-u_i)$
- ***Average Number of users at each Service Center*** $(Pi)= u_i/(1 – u_i)$

From above equations it is possible to make calculations for latency, throughput, and highly utilized service centers.

7. MACHINE LEARNING

Machine learning is the capability of the computer program to acquire or develop new knowledge or skills from existing or non existing examples for the sake of optimizing performance criteria [12]. Software engineers and researchers have been started using machine learning techniques in the area of quality of service classification and prediction. Moreover, machine learning has proved it is efficiency to asset and optimizes model based performance prediction.

Machine learning can be categorized into two groups that are, supervised and unsupervised machine learning. These two learning categories are associated with different machine learning algorithms which represent how the learning method behaves [12].

7.1 Supervised Learning

Supervised learning comprises of algorithms that reason from externally supplied instances to produce general hypothesis which then make predictions about unseen instances. Moreover, with supervised learning there is presence of the outcome variable to orient the learning process. There are many machine learning algorithms for supervised learning such as Support Vector Machine (SVM), K-Nearest Neighbor, and Random Forests.

7.2 Unsupervised Learning

Opposite to supervised learning where there is presence of the outcome variable to orient the learning process, unsupervised learning builds models from data without predefined example [12]. This means no guidance is available and learning must perform heuristically by the algorithm examining different training data.

8. A CASE STUDY

Step 1: UML Annotated Diagrams

In order to make our proposed approach of a model based performance prediction more understandable, an example is represented describing a commerce system.

The application scenario starts as the client browses the system to obtain information about products and prices via commercial server. The commercial server connects the database server in order to get the information required and complete the task. Moreover, the client can use the system to purchase products via the commercial server. In this step, the commercial server connects the online shopping mall server to make the purchase operation and accomplish the job.

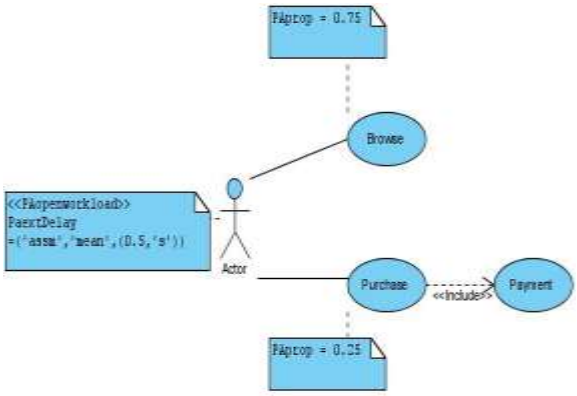


Figure 4. The system described by Use Case Diagram

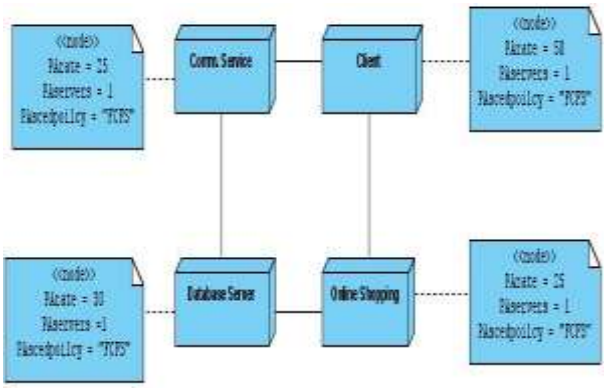


Figure 5. The System described by Deployment Diagram

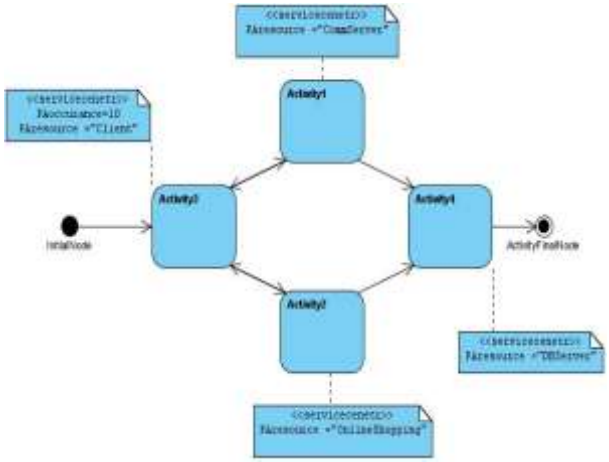


Figure 6.The System described by Activity Diagram

Fig.4 shows the annotated Use Case diagram. Each actor in a use case diagram may represent a stream of requests arriving at the system. There may be an unlimited sequence of requests (open workload) stereotyped as <<OpenUser>>, or a fixed population of users requiring service from the system (closed workload) stereotyped as <<ClosedUser>>. The tagged value can be linked with use case diagrams:

PApopulation specifies the total number of requests circulating in the system.

Fig.5 represents the annotated deployment diagram. Deployment diagrams are used to model the physical resources available in the system. Each resource is represented by a node in the deployment diagrams. Each node which must be stereotyped as <<node>> represents a processor with a given scheduling policy. The tagged values can be link with deployment diagrams:

PAschedpolicy denotes the scheduling policy of the processor, which can be one of: First in First out (FIFO), or Last in First out (LIFO), or Processor Sharing (PS).

PArate denotes the processing rate of the processor or time taken by the processor to complete a task.

PAservers denotes the number of processors simultaneously executing the requests.

Fig.6 shows the annotated Activity diagram. Activity diagram describes the workloads and the physical resources available in the system, also it's specifies how the resources are used in the meaning of computations are performed in the system.

Each action state of an activity diagram, stereotyped as <<ServiceCenter>>, represents a computation which requires service to one resource. Tagged values below can be specified to convoy information for making up the performance model:

PAresource denotes the name of the resource from which service is requested, which called node in Deployment diagram.

PAoccurrence denotes the inter-arrival time of this service request. It can be used with Queuing Network Model to specify open queuing model.

Step 2: Transformation to QNM

Based on Table 1 notation algorithm 1 is used to translate annotated UML elements into the corresponding QNM components. Each node in the Deployment diagram defines a service center. Use case diagram used to categories the type of model (open or closed), and from Activity diagrams network topology is derived.

Algorithm 1: QNM Generation Algorithm 1

for all Deployment diagram node *Ri, i = 1 .. N* **do**
  S_i := *New Service Center*
  $\mu_i := 1/PAserviceTime(R_i)$
  $NS_i := PAservers(R_i)$
end for
Let $C \leftarrow 0$
for all Actor *A* **do**
  Initialize routing matrix $\mathbf{P^c}$ for class *C* to zero
  AD := Active diagram associated to *A*
  for all Transition t from action state a_i to a_j of Activity diagram *AD* **do**
    $R_k := PAhost(a_i)$
    $R_l := PAhost(a_j)$
    $P^c_{k,l} := PAprob(t)$
  end for
if *A* is a ClosedUser **then**
    Set class *C* as a closed chain with *PApopulation (A)* requests
else
  Set class *C* as an open chain
  for all Action state *a* of Activity diagram *AD* **do**
    if *PAoccurrence(a)* is defined **then**
      $R_i := PAhost(a)$
      $\lambda^c_i := PAoccurrence(a)$
    end if
  end for
end if
  Let $C \leftarrow C + 1$ {New Customer Class}
end for

Table1. Notation used for Algorithm1

N	Number of nodes in the UML deployment diagram.
S_i	i^{th} service center
μ_i	Service rate of service center S_i
λ^c_i	Arrival rate of class *C* customers at service center S_i
NS_i	Number of servers in service center S_i
P^C	$N \times N$ routing matrix of class *C* customers

Step 3: QNM Analysis

The translation algorithm works on the annotated UML diagrams, and starts mapping the UML into QNM model. Fig7 presents the derived QNM from the Use case, Deployment, and Activity diagrams of respectively, assuming that the Activity diagram is associated to an actor stereotyped <<OpenUser>>.

We used a tool for drawing and calculating performance indices from queuing networks called **P**erformance **E**valuation and **P**rediction **SY**stem for Windows platforms (**WinPEPSY-QNS**).

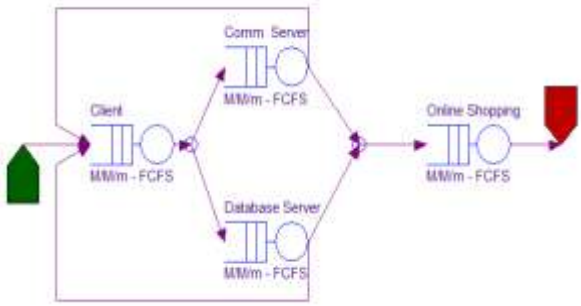


Figure 7. QNM - Output of the Transformation Algorithm

WinPEPSY-QNS analyzes the QNM and gets the result by using Mean Value Analysis (MVA) algorithm.
The analysis of the QNM of Fig. 7 provides a set of average performance indices that include utilization, throughput, response time, and latency. Table 2 shows numerical results for the set of parameters value for the whole system. The performance analysis can provide indication for possible system modification and further software performance evaluation can be iterated by choosing a different set of parameters value, to be inserted in the UML annotation.

Table 2. Performance Result for the QNM Fig. 7

Utilization (U_i)	**0.4**
Throughput (X_i)	**10.0**
Response Time (R_i)	**1.42**

Step 4: Machine Learning Prediction

According to above performance indices presented on table 2 we will consider the results as a test set. To complete the test set we assumed availability, success-ability, reliability, and number of operations performed by the web service the complete test set will be as Table 3. We used **W**aikato **E**nvironment for **K**nowledge **A**nalysis (**WEKA**) as a tool for machine learning. In order to classify the test set as "**Risk**" or "**No Risk**".

Table 3. The Complete Test Set

Utilization (U_i)	**0.4**
Throughput (X_i)	**10.0**
Response Time (R_i)	**1.42**
Availability (Av_i)	**100**
Success-ability (Su_i)	**100**
Reliability (Re_i)	**100**
Operation (Op_i)	**9**
Class	**No risk**

Utilization (U_i) means total amount of resources required in order to complete a task, Throughput (X_i) is a total number of invocations for a given period of time, Response time (R_i) is the time taken to send a request and receive a response, Success-ability (Su_i) measured as number of response / number of request messages, Operation (Op_i) is the number of operations performed by the web service.

Step 5: The Result

We have used a training set contains 166 web services and their measurements collected by (Al-Masri, 2007) to train our model. After feeding the test set on the machine learning the result confirms our claim that there will be "No Risk" if we apply the current web service architecture. The result is promising as we get relative absolute error 15.4 % fig.8.

Figure8. Result

9. CONCLUSION

In this paper, we proposed a new technique to predict the performance of service oriented architecture based applications. The study focuses on web service as implementation of SOA. This approach avoid the needs of transforming UML model into queuing network at each time we want to predict the performance at early stages of development process.

We extract utilization, response time, throughput, and number of operations from the architectural description after mapping application architecture into QNM model. Machine learning technique has been used to predict is there will be risk if we implement the web service with same indices or no risk.

As a result the prediction model gives error percentage 15.4 % which considered as promising result.

REFERENCES

[1] Mani, N., Petriu, D. C., & Woodside, M. (2011, March). Towards studying the performance effects of design patterns for service oriented architecture. In*Proceedings of the 2nd ACM/SPEC International Conference on Performance engineering* (pp. 499-504). ACM.

[2] Ganapathi, A. (2009). Predicting and optimizing system utilization and performance via statistical machine learning.

[3] Dubach, C. (2009). Using machine-learning to efficiently explore the architecture/compiler co-design space.

[4] Malhotra[1], R., & Chug, A. (2012). Software maintainability prediction using machine learning algorithms. *Software Engineering: An International Journal (SEIJ)*, *2*(2).
[5] Mohanty, R., Ravi, V., & Patra, M. R. (2012). Classification of web services using Bayesian network.

[6] Ipek, E., De Supinski, B. R., Schulz, M., & McKee, S. A. (2005). An approach to performance prediction for parallel applications. In *Euro-Par 2005 Parallel Processing* (pp. 196-205). Springer Berlin Heidelberg.

[7] Balsamo, S., Mamprin, R., & Marzolla, M. (2004). Performance evaluation of software architectures with queuing network models. Proc. ESMc, 4.

[8] Punitha, S., & Babu, C. (2008, September). Performance prediction model for service oriented applications. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on* (pp. 995-1000). IEEE.

[9] Moniem, H. A., & Ammar, H. H. (2014) Performance Prediction of Service-Oriented Architecture-Asurvey.

[10] Youn, H., Jang, S., & Lee, E. (2007, August). Deriving queuing network model for UML for software performance prediction. In *Software Engineering Research, Management & Applications, 2007. SERA 2007. 5th ACIS International Conference on* (pp. 125-131). IEEE.

[11] Punitha, S., & Babu, C. (2008, September). Performance prediction model for service oriented applications. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on* (pp. 995-1000). IEEE.

[12] Omary, Z., & Mtenzi, F. (2010). Machine learning approach to identifying the dataset threshold for the performance estimators in supervised learning.*International Journal for Infonomics (IJI)*, *3*, 314-325.

Haitham A.Moniem is a lecturer of Computer Science and Information Technology in College of Information Technology at Future University, Sudan. He is a PhD student at Sudan University of Science and Technology. He received B.Sc. and M.Sc. from Gizera University, Sudan; (2002, 2006). His research interest includes Software Engineering, Service-Oriented Architecture, Machine Learning, Cloud Computing, and Software Architecture.

Dr.Ammar is a Professor of Computer Engineering in the Department of Computer Science and Electrical Engineering at West Virginia University. He has published over 170 articles in prestigious international journals and conference proceedings. Dr. Ammar is currently Editor in Chief of the Journal of Computer Science and Engineering, in Arabic. Dr. Ammar has been teaching in the areas of Software Engineering and Computer Architecture since 1987. He has been Principal Investigator on a number of research projects on Software Risk Assessment and Software Architecture Metrics funded by the NASA IV&VFacility and NSF, and projects on Automated Identification Systems funded by NIJ and NSF. Dr. Ammar is a member of the IEEE and the ACM professional organizations. He has served as Chairman and member of Steering Committees and Program Committees of several International Conferences and Symposia. He previously served as the Chairman of the Upper Monongahela Subsection of the Pittsburgh section of the IEEE, a Director of the Pittsburgh section of the IEEE, and the student section advisor of the IEEE Computer Society at WVU.