



**Sudan University of Science and Technology**  
**College of Graduate Studies**

**Model-Based Prediction of Resource Utilization  
and Performance Risks**

التنبؤ المبني على نموذج لإستغلال الموارد و مخاطر الاداء

A Dissertation Submitted to the College of Graduation Studies,  
Faculty of Computer Science and Information Technology,  
Sudan University of Science and Technology  
In Partial Fulfillment of the Requirements for the Degree of  
**DOCTOR OF PHILOSOPHY**  
Major Subject: Software Performance Prediction

**By**

**Haitham Abdel Moniem Mohamed**

**Supervisor**

**Hany Ammar, Professor**

**May, 2018**

الآية

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ  
فَدَلَّ عَلَاقَةَ الْأَمْرِ وَالْمَرْكُورِ الْعَمَلِ  
بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

صَدَقَ اللَّهُ الْعَظِيمُ

# Dedication

I would like to dedicate my work to

My family,

My teachers, and

My friends ....

# ACKNOLEGEMENT

Firstly, *Alhamdulillah*

Secondly, I would like to express my deepest thanks, respect, and appreciation to **Prof. Hany Ammar**, my supervisor for his ideas, inspiration, encouragement, support, and patient.

Many thanks for **Sudan University of Science and Technology** for hospitality, generosity, and attention.

I also place my sincere thanks to **Future University** for helping, supporting, and funding my PhD.

Haitham A.Moniem

# ABSTRACT

The growing complexity of modern software systems makes the prediction of performance a challenging activity. Many drawbacks incurred by using the traditional performance prediction techniques such as simulation, guessing, and depending on previous experience. Moreover, performance assessment and prediction is time consuming activity and may produce inaccurate results especially in complex and large scale software applications.

To contribute to solving these problems, we adopt a model-based approach for resource utilization and performance risk prediction. The steps of the approach can be stated as follows: Firstly, we model the software system into annotated UML diagrams. Secondly, performance model is derived from the annotated UML diagrams in order to be evaluated. Thirdly, we run the performance model to generate and record performance indices such as response time, system throughput, and resources utilization into a large dataset by different values of workload. Finally, we can predict different performance indices for new workloads based on previously observed performance dataset. In addition to this, we can assess the software performance risk incurred on a given workload into three classes of performance risk level either low, or medium, or high. The approach could be used to enhance the work of human experts and improve efficiency of software performance prediction and risk assessment.

In this research, we validate the approach by applying three different case studies: a hospital system, an e-commerce system, and an online timetable system. The results were compared of three machine learning techniques for performance risk prediction and the approach shows prediction accuracy between 93.1 % and 97.6 %.

## المستخلص

إن التعقيد المتنامي لأنظمة البرمجيات الحديثة يجعل التنبؤ بأداء البرمجيات من المهام الصعبة لمهندسي البرمجيات. وهناك العديد من العيوب في تقنيات التنبؤ بالأداء التقليدية مثل المحاكاة، التخمين، والاعتماد على الخبرة السابقة ذلك لاستغراق هذه التقنيات وقتا طويلا في التنبؤ بأداء البرمجية. وقد يؤدي أيضا إلى أن تكون هذه النتائج غير دقيقة خاصة في التطبيقات المعقدة والكبيرة الحجم.

للمساهمة في حل هذه المشاكل، نقترح منهجا قائما على التنبؤ باستغلال الموارد و مخاطر أداء البرمجية من تخطيط لغة النمذجة الموحدة. و تتبع المنهجية المقترحة الخطوات الآتية: أولاً، نقوم بتصميم مخططات البرمجية باستخدام لغة النمذجة الموحدة التوضيحية. ثانياً، يتم تحويل نماذج و مخططات البرمجية إلى نماذج الأداء لتقييمها ودراستها. ثالثاً، نقوم بتشغيل نموذج أداء البرمجية لإنشاء وتسجيل مؤشرات أداء البرمجية و حفظها في فئة بيانات كبيرة تحتوي على سرعة استجابة النظام، إنتاجية النظام، و استغلال موارد الأجهزة و ذلك مع تغير عبء العمل. أخيراً، يمكننا التنبؤ بمقاييس أداء البرمجيات بقيم و قراءات متعددة استناداً على اعباء العمل السابقة الموجودة في فئة البيانات. و علاوة على ذلك، يمكننا أيضاً تصنيف المخاطر المرتبطة بالأداء بسبب عبء عمل معين. و يمكن استخدام هذه المنهجية لتعزيز عمل الخبراء و تحسين كفاءة التنبؤ بأداء البرمجيات.

في هذا البحث، تم التحقق من صحة المنهجية المقترحة من خلال تطبيق ثلاث دراسات و تجارب مختلفة هي: نظام معلومات مستشفى، و نظام التجارة الإلكترونية، و نظام جدول زمن المحاضرات للجامعة. و تمت مقارنة نتائج ثلاث تقنيات تعليم الآلة للتنبؤ باستغلال الموارد و مخاطر الأداء و تظهر المنهجية دقة تنبؤ بعد تنفيذ التجارب بين 93.1% إلى 97.6%.

# TABLE OF CONTENTS

الإهداء .....	ii
Dedication .....	iii
ACKNOLEGEEMENT .....	iv
ABSTRACT.....	v
المستخلص .....	vi
TABLE OF CONTENTS .....	vii
LIST OF TABLES .....	xii
LIST OF FIGURES.....	xiv
LIST OF ABBREVIATIONS.....	xvi
PUBLICATIONS .....	xvii
CHAPTER I .....	1
Research Introduction.....	1
1.1 Introduction:.....	1
1.2 Problem Statement: .....	2
1.3 Research Questions .....	3
1.4 Research Hypothesis .....	3
1.5 Research Objectives .....	4
1.6 Scope of the Research .....	4
1.7 Research Contribution.....	4
1.8 Thesis Outline: .....	5

<b>CHAPTER II.....</b>	<b>7</b>
<b>Background and Related Work.....</b>	<b>7</b>
2.1 Introduction.....	7
2.2 An Overview of Software Performance Risk Assessment.....	7
2.3 Software Modeling Notation.....	8
2.3.1 Unified Modeling Language .....	8
2.4 Performance Modeling Notations .....	9
2.4.1 Queuing Networks .....	10
2.5 UML Profile for Schedulability, Performance and Time.....	12
2.5.1 <i>PAprofile</i> : Stereotypes and Tagged Values.....	12
2.5.2 << <i>PAcontext</i> >> Stereotype .....	12
2.5.3 << <i>PAClosedLoad</i> >> and << <i>PAopenLoad</i> >> stereotypes.....	13
2.5.4 << <i>PAstep</i> >> Stereotype .....	15
2.5.5 << <i>PAhost</i> >> and << <i>PAresource</i> >> Stereotypes.....	16
2.6 Machine Learning .....	17
2.6.1 Machine Learning Categories .....	18
(a) Supervised Learning .....	18
(b) Unsupervised Learning.....	20
2.6.2 Machine Learning Techniques .....	20
(a) Multivariate Regression.....	20
(b) Naïve Bayes Classifiers.....	20
(c) Bayesian Networks .....	21
(d) K-Nearest Neighbor (KNN) .....	24
(E) Support Vector Machines (SVMs):.....	24
2.7 Related Work: .....	26
2.8 Summary .....	29
<b>CHAPTER III .....</b>	<b>30</b>



<b>Methodology for Model-based Prediction of Resource Utilization and Performance Risk .....</b>	<b>30</b>
3.1 Introduction.....	30
3.2 Challenges in Performance Prediction .....	30
3.3 The steps of the Approach .....	32
3.4 Summary .....	36
<b>CHAPTER IV .....</b>	<b>37</b>
<b>Predicting Performance of Hospital Information System.....</b>	<b>37</b>
4.1 Introduction.....	37
4.2 The Architecture and Model .....	38
4.3 Transforming UML diagrams into Performance Model .....	42
4.4 Running the Performance Model .....	43
4.5 Generating the dataset.....	45
4.6 Resource Utilization Prediction using Machine Learning .....	48
4.7 Performance Risk Classification using Machine Learning .....	49
4.8 Results and Discussion.....	49
4.8.1 Database Server Utilization Prediction .....	49
4.8.2 Performance Risk Level Classification .....	50
4.9 Summary .....	54
<b>CHAPTER V .....</b>	<b>55</b>
<b>Predicting Performance of E-commerce System .....</b>	<b>55</b>
5.1 Introduction.....	55
5.2 The Architecture and Model .....	55
5.3 Transformation of UML Diagrams into Performance Model .....	60
5.4 Running the Performance Model: .....	61
5.5 Generating the dataset.....	64
5.5.1 Performance Prediction dataset.....	64

5.5.2 Performance Risk Classification dataset .....	67
5.6 Performance Prediction using Multivariate Regression.....	69
5.7 Performance Risk Assessment using Machine Learning Techniques.....	70
5.8 Results and Discussion.....	70
5.8.1 Prediction of System Response Time .....	70
5.8.2 Performance Risk Level Classification .....	71
5.9 Summary .....	76
<b>CHAPTER VI .....</b>	<b>77</b>
<b>Prediction Performance of Online Time Table (OTT).....</b>	<b>77</b>
6.1 Introduction.....	77
6.2 The System Architecture.....	77
6.3 Collecting Performance Dataset using Apache JMeter.....	80
6.4 Using Statistical Machine Learning Multivariate Regression: .....	81
6.5 Using Machine Learning for Performance Risk Assessment: .....	84
6.6 Using WEKA to Visualize the Dataset .....	85
6.7 Using Multivariate Regression.....	88
6.8 Prediction Performance Risk Class.....	89
6.8.1 Using Naïve Bayes Technique .....	89
6.8.2 Using KNN Technique.....	90
6.8.3 Using SVM Technique.....	91
6.9 Summary .....	93
<b>CHAPTER VII.....</b>	<b>94</b>
<b>DISCUSSION AND RESULTS .....</b>	<b>94</b>
7.1 Introduction.....	94
7.2 Dataset Preparation .....	95
7.2.1 Selecting Data .....	95
7.2.2 Processing Data.....	96

7.3 Comparison with Existing Studies .....	96
7.4 Result Discussion .....	99
7.5 Summary .....	100
<b>CHAPTER VII.....</b>	<b>101</b>
<b>CONCLUSION AND FUTURE WORK.....</b>	<b>101</b>
<b>REFERENCES.....</b>	<b>104</b>

# LIST OF TABLES

Table 3.1: Attributes of the performance dataset.....	35
Table 4.1: Service demands in milliseconds for each station in the system.....	39
Table 4.2: Resource utilization prediction .....	45
Table 4.3: Performance level classification dataset .....	47
Table 4.4: Database server utilization prediction accuracy .....	49
Table 4.5: Naïve Bayes performance risk classification accuracy .....	50
Table 4.6: Naïve Bayes performance risk classification confusion matrix .....	50
Table 4.7: KNN performance risk classification accuracy .....	51
Table 4.8: KNN performance risk classification confusion matrix .....	51
Table 4.9: SVM performance risk classification accuracy .....	52
Table 4.10: SVM performance risk classification confusion matrix .....	52
Table 4.11: Comparison between classification algorithm .....	53
Table 5.1: Service demands in milliseconds for each station in the system .....	56
Table 5.2: ECS response time prediction using regression model dataset .....	65
Table 5.3: Sample of classification dataset for performance risk level .....	67
Table 5.4: ECS response time prediction accuracy .....	71
Table 5.5: Naïve Bayes performance risk classification accuracy .....	72
Table 5.6: Naïve Bayes performance risk classification confusion matrix .....	72
Table 5.7: KNN performance risk classification accuracy .....	73
Table 5.8: KNN performance risk classification confusion matrix .....	73
Table 5.9: SVM performance risk classification accuracy .....	74
Table 5.10: SVM performance risk classification confusion matrix .....	74
Table 5.11: Comparison between classification techniques .....	75
Table 6.1: Sample of OTT response time dataset .....	81
Table 6.2: Sample of OTT for performance risk classification .....	84
Table 6.3: Using multivariate regression to predict OTT response time .....	88
Table 6.4: Using Naïve Bayes to classify performance risk level .....	89

Table 6.5: Confusion matrix for Naïve Bayes algorithms .....	89
Table 6.6: Using KNN algorithms to classify performance risk level .....	90
Table 6.7: Confusion matrix for KNN algorithms .....	91
Table 6.8: Using SVM algorithm to classify performance risk .....	91
Table 6.9: Confusion matrix for SVM algorithm .....	91
Table 6.10: Comparison of classifiers .....	92
Table 7.1: Comparison with Existing Studies .....	98
Table 7.2: Comparison between Case Studies using KNN technique .....	99

# LIST OF FIGURES

Figure 1.1 How to predict resource utilization and performance risk from a model? .....	2
Figure 2.1 Queuing Network basic elements .....	10
Figure 2.2 Queuing Network example with multiple class.....	33
Figure 2.3 Example of <<PAcontext>> annotation .....	13
Figure 2.4 Example of <<PAopenLoad>>annotation.....	14
Figure 2.5 Example of <<PACloseLoad>>annotation .....	15
Figure 2.6 Example of <<PAstep>> annotation .....	16
Figure 2.7 Example of <<PAhost>> and <<PAresource>> .....	17
Figure 2.8 The process of supervised learning .....	19
Figure 2.9 The structure of Bayes Network.....	23
Figure 2.11 SVM classification .....	25
Figure 3.1 Our goal to mimic the behavior of real system by using model.....	31
Figure 3.2 The framework of the approach .....	33
Figure 4.1 Use case diagram for Hospital system .....	40
Figure 4.2 Sequence diagram for Hospital system .....	41
Figure 4.3 Deployment diagram for Hopital system .....	41
Figure 4.4 Queuing Network Model for Hospital system.....	42
Figure 4.5 The system response time.....	43
Figure 4.6 System throughput .....	44
Figure 4.7 System server utilization .....	44
Figure 4.8 Detaield accuracy by class .....	54
Figure 5.1 ECS use case diagram .....	57
Figure 5.2 ECS component diagram.....	58
Figure 5.3 ECS sequence diagram.....	59
Figure 5.4 ECS deployment diagram.....	60
Figure 5.5 Queuing Network Model for ECS .....	61

Figure 5.6 ECS response time.....	62
Figure 5.7 ECS throughput .....	63
Figure 5.8 ECS server utilization.....	64
Figure 5.9 The detailed accuracy by class .....	75
Figure 6.1 OTT use case diagram .....	78
Figure 6.2 OTT sequence diagram for student .....	79
Figure 6.3 OTT system deployment diagram .....	79
Figure 6.4 OTT system response time .....	86
Figure 6.5 OTT system throughput .....	86
Figure 6.6 OTT application server utilization .....	87
Figure 6.7 using multivariate regression to predict OTT response time .....	88
Figure 6.8 Detailed accuracy by class .....	93

# LIST OF ABBREVIATIONS

BN	Bayesian Network
JMT	Java Modeling Tool
KNN	K-Nearest Neighbor
MLR	Multi linear Regression
MVA	Mean Value Analysis
NB	Naiïve Bayes
OMG	Object Management Group
OTT	Online Time Table
QNM	Queuing Network Model
SDLC	System Development Life Cycle
SVM	Support Vector Machine
UML	Unified Modeling Language
WEKA	Waikato Environment for Knowledge Analysis



## **PUBLICATIONS**

Salih, H.A.M. & Ammar, H.H., 2017. Model-Based Resource Utilization and Performance Risk Prediction using Machine Learning Techniques. , 1(3), pp.101–109. International Journal on Informatics Visualization

Moniem, H.A., 2015. A framework for Performance Prediction of Service-Oriented Architecture. , 4(11), pp.865–870. International Journal of Computer Applications Technology and Research

Moniem, H.A., 2014. Performance Prediction of Service-Oriented Architecture - A survey. , 3(12), pp.831–835. International Journal of Computer Applications Technology and Research

# CHAPTER I

## Research Introduction

### 1.1 Introduction:

Nonfunctional requirements validation of software systems does not meet a proper consideration from software developers yet. Minimum time and effort are given to this part during the software development life cycle, and the approach of “fix-it-later” is still dominant (Cortellessa et al. 2005). This benefits the software developers to shorten time to the market, but software system ability to meet maintainability may degraded after software system implementation.

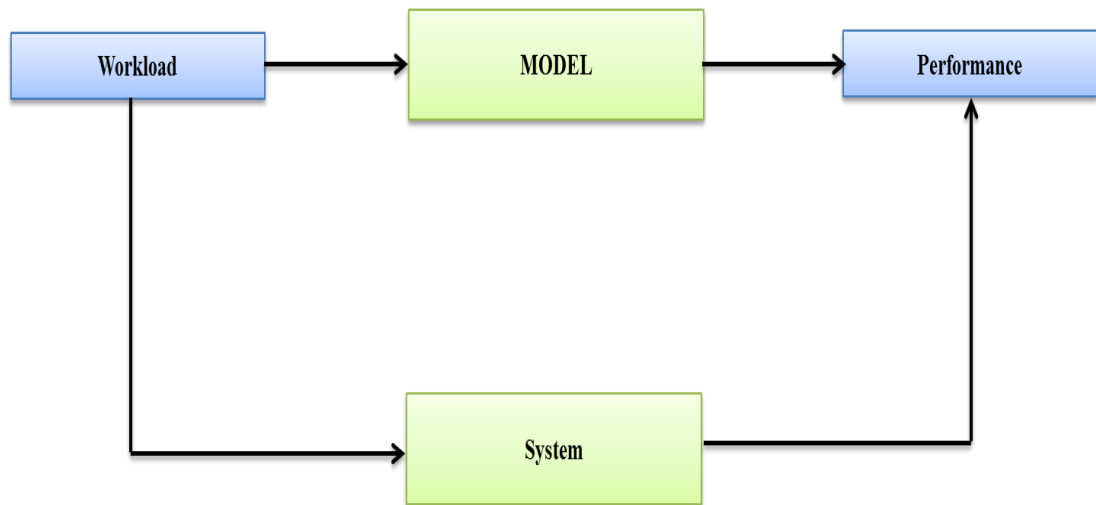
Performance is one of the important nonfunctional requirements defined as the amount of resources needed by the software to accomplish its functionality under all possible environmental conditions. Software performance risk is defined as a probability that some adverse circumstance will occur in a system. Performance risk is a combination of two factors: the likelihood of the occurrence and the consequences of this risk (Radhakrishnan & Virginia 2007). Moreover, performance failure defined as an unexpected performance result originated from the violation of a performance requirement. Early identification of performance metrics such as response time, utilization, throughput etc. is key step to risk management of software system. There are two basic types of performance requirements:

- Time-related, defined as the completion time of a specific operation must be less than a certain threshold.
- Resource-related, defined as the utilization of a specific resource must fall into a certain range.

## 1.2 Problem Statement:

Performance has a significant impact in software systems. However, many software products fail to meet their performance requirement on performance when they are initially built. Overcoming and fixing these problems is costly for two parts, software developer and software customer. For the former part it causes schedule delays, and cost overruns, equally for the second part it leads to productivity lost, damaged customer relations, missed opportunities, and host other difficulties (Smith 1981).

Traditional software development approaches focus on software correctness, handling performance issues comes later in the development process. This style of developing defined as “fix-it-later” which is the dominant style in the software industry. Furthermore, in the researchers community there is a growing interest in the performance field and there are several approaches to early software performance predictive assessments have been proposed (Balsamo et al. 2004).



**Figure 1.1 How to predict resource utilization and performance risk from a model?**

Several approaches have been created to evaluate and predict software system performance. Most of them are based on Petri-Nets, stochastic algebra, simulation technique or queuing networks. A good background has been published by a survey paper

(Moniem 2014). Also several proposals have been made to apply performance modeling to the cycle of software development process. However, work on model based performance risk assessment has been limited due to its complexity.

This thesis has recognized the need for a special-purpose of performance risk evaluation for systems that considered as important and crucial such as hospital systems, e-commerce systems, and online timetable systems (OTT). The proposed approach takes into account the changing of workload and number of customer, the response time of the system, and the utilization of server's hardware. Therefore, the new performance assessment approach can be used by the software engineers for evaluating different parameters before system implementation to determine the configuration of the system to meet user needs after deployment(Izzeldin & Osman 2010). From Fig 1.1 the research problem can be stated as how to:

- Find a relationship between changing workloads and performance metrics.
- Predict software performance and classify performance risk into levels low, medium, or high performance risk.

### **1.3 Research Questions**

In this research we tried to answer the following questions regarding model-based performance prediction and performance risk assessment:

- Q1: What are performance metrics and class of performance risk given a certain workload?
- Q2: Which suitable Machine Learning technique to be used for predicting system performance and risk assessment?

### **1.4 Research Hypothesis**

The hypothesis of this research is that if we use queuing theory and machine learning techniques we can predict software system performance and assess performance risk into levels low, medium, or high on given workload. Moreover, the proposed

methodology is expected to give promising results as it will be extend queuing theory by adding machine learning as intelligent tool that will enrich performance prediction process.

### **1.5 Research Objectives**

The general objective of this research is to easy software performance prediction at early design time in order to able software engineers and practitioners focusing on delivering good software. In addition, the specific research objectives can stated as follows:

- To propose a framework for a methodology to simplify software performance assessment.
- To predict multiple software performance metrics for new workloads based on previous observed workload. All these activities done during software designing and before software implementation. Software performance metrics agreed on in the research are response time, throughput, and resource utilization according to ISO 9126(Adhianto et al. 2010).
- To assess the risk incurred by specific workload into either one of three levels low, medium, or high risk.
- To verify and validate the framework using three case studies
- To find suitable machine learning techniques that assist software engineers and practitioners to evaluate performance risk incurred by given workload.

### **1.6 Scope of the Research**

The proposed approach focuses on prediction and assessment of performance risk based on models. We used annotated UML diagrams to understand software system at early design stages. UML has become the common language of software development, allowing engineers to exchange their designs freely(Ram et al. 2011b).

### **1.7 Research Contribution**

This thesis contributes a model-based performance risk prediction by using machine learning techniques at modeling time. By using machine learning techniques we will mimic

the dynamic behavior of software systems for enabling software engineers to evaluate the expected performance indices and performance risk at designing phase, and before coding and implementing of the software system. The contributions of this thesis are:

- Propose a framework for a methodology to simplify software performance assessment.
- Predict performance metrics from UML models by applying ML techniques to benefit parameter estimation even with noisy and missing data.
- Assess the risk incurred by specific workload into either one of three levels low, medium, or high risk.
- Verify and validate the framework using three case studies
- Compare different machine learning techniques to find the suitable techniques.

### **1.8 Thesis Outline:**

The rest of thesis is structured as follows:

**Chapter 2** presents background material for model-based performance risk assessment. A categorization of performance models used in software performance assessment. In addition, the uses of machine learning techniques in software performance prediction field are discussed in detailed.

**Chapter 3** introduces our approach for the model-based performance risk prediction using machine learning techniques. The phases of implementation the method are explained. A diagram of the framework is used along with the methodology to easy understanding of the approach.

**Chapter 4** this chapter introduces a case study of a hospital system that describes the use of this technique. The case represents applying the approach on a small system. We modeled the system into annotated UML diagrams, then transformed to performance model to generate the dataset, and we apply three machine learning techniques to validate the results.

**Chapter 5** in this chapter the approach applied on a case study of an ecommerce system. We applied our approach on a system that considered as an open large system. We apply the same steps as mentioned on chapter four, and we considered this chapter as second validation case study for the approach.

**Chapter 6** in this chapter the approach has been validated applying the approach on already running system for a university. The system is online timetable (OTT) serves teachers, students, administration, and parents. We used a tool called JMeter to generate a performance dataset in order to visualize, predict, and classify performance risk.

**Chapter 7** in this chapter we stated the results and discussions of the research. We compare the results with two related works from the literature. Moreover, we conducted a comparison between the results of the three cases in order to come with main findings.

**Chapter 8** in this chapter we presented the conclusion of our work, also we mentioned the open issues for new researchers on the same field.

## **CHAPTER II**

### **Background and Related Work**

#### **2.1 Introduction**

In this chapter, the context for the thesis is set to review the model-based performance risk assessment and machine learning in the literature. The chapter begins with an overview of model-based performance prediction in software development. Then, software modeling notation and performance modeling notation are discussed. The software life cycle and performance analysis are stated in details. Finally the chapter concludes with the justification of our approach and our contribution.

#### **2.2 An Overview of Software Performance Risk Assessment**

The increasing complexity of software and its spread out in everyday life in the last years encouraged growing interest for software performance analysis. This has basically directed to evaluate functional attributes of the software system and, in the case of safety critical systems, known as dependability property (Ionita & Hammer 2002). The quantitative behavior of a software system has gained popularity recently with the emerging of software performance analysis. This type of analysis aims at evaluating quantitative behavior of a software system by deep analyzing its structure and its quality, from design to implementation.



Software performance is defined as the amount of resources needed by the software system to provide full functionality under all possible environmental conditions. Software performance risk is defined as undesired event or any uncertainty happened in the system. The Software performance is defined as a combination of two factors, probability of performance failure and the severity due to this failure. Performance failure happens when a software system violate the required function of the performance requirements (Radhakrishnan & Virginia 2007).

### **2.3 Software Modeling Notation**

Software architects describe the static and dynamic aspects of the system by using models. The static description comprises the software components or modules, while the dynamics description of a software system states the system behavior at run time. There are many types of notations to describe the static or the dynamic behavior of software system. This part of the research will focus on notations that permit for the dynamic behavior description since performance is a dynamic attribute of the system. In addition, the system software behavior is important but not enough to fill out performance assessment of a system, moreover the behavioral description of the software system has to be supported by additional information such software system service demands and operational profiles.

According to (Cortellessa et al., 2011) there are two types of software model notations: (a) basic notations inherited from computer scientists to model software system, such as Automata, Process Algebras, Petri Nets, and Message Sequence Charts; (b) Unified Modeling Language (UML) that has become dominant standards for modeling software systems. Unlikely, modeling complex and real software system with pervious notations turns out to be very complex, given that the research will focus on UML.

#### **2.3.1 Unified Modeling Language**

UML, declared by the Object Management Group (OMG), is a modeling language for visualizing, specifying, constructing and documenting artifacts of software system. Examples of these artifacts are requirements, architecture, design, source code, test cases, and prototype. Most of the basic notations describing software dynamics, such as Petri

Nets, Message Sequence Charts, and Automata have been inspired UML diagrams(Canevet et al. 2003).

UML diagrams model the software system either statically or dynamically in object oriented style. The dynamics of a software system can be stated by using interaction diagrams which describe the message exchange among instances, or by using state diagrams to specify the internal behavior of each software components, or by using activity diagrams to show the flow of activities performed by all the computation of interest, or by using combinations of both interaction and state diagrams.

The system structure can be modeled by component diagram and class diagram. The component diagram describes general view of the software system as combination of software subsystem or components; whereas the class diagram presents a view that describe how the software system will be constructed in an object oriented style defining classes and their relations(Ram et al. 2011a). After all, UML offers the description of deployment of software components to hardware nodes by using the deployment diagram.

## **2.4 Performance Modeling Notations**

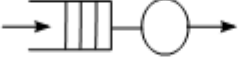




One of the major problems of emerging software performance modeling with software development life cycle is the large gap between notations for static and dynamic modeling (such as UML) and notations for modeling performance (such as Queuing Networks).As per ISO 9126, the performance indices that may be required for software systems are mainly: response time, throughput and utilization (Moniem 2015).

There are many performance modeling notations, such as Stochastic Petri Nets Markov processes, Queuing Networks, Layered Queuing Networks, Execution Graphs, Stochastic Process Algebras, and Simulation Models. The performance notation we will use it in our research is Queuing Network Modeling (QNM), this is for the purpose of popularity as performance model represent and analyze resource sharing system(Kattepur et al. 2015). Furthermore, QNM has a good combination of enough satisfying accuracy in performance results and the powerfulness in model analysis and evaluation.

### 2.4.1 Queuing Networks

Queuing Network is a collection of interacting Service Centers representing system resources and customers representing the users receiving services and share the resources. The building of a QN can be divided into two steps: Definition, this is includes representation of service centers, number of service centers, and the interconnection topology of the network. Parameterization, which marks the input values of the network, which are the arrival processes, number of customers, the classes of customers, the service rates, policy of scheduling, and length of the queue.

From Fig. 2.1 the basic elements are of queuing network illustrated. The Queued center is a node where processes/jobs arrive, if the server busy then stays in the queue for their turn. After each job finished a new job is fetched from the queue based on a certain scheduling style. A delay center is a node that makes each process/job passing it afford a specific delay time (Canessane & Srinivasan 2013). Queuing network can be open or closed, as a result Source, Sink, and Terminals nodes will be considered.

QN symbol	Graphical notation	Open QN	Closed QN
Queued center		×	×
Delay center		×	×
Source		×	
Sink		×	
Terminals			×

**Figure 2.1: Queuing Network: basic elements(Cortellessa et al. 2005)**

The number of customers or the arrival processes represents the QN workload, which is considered as the amount of requests that are addressed to the QN. There are two types of workload: Open workload; which completely defined by the stochastic process that

identifies the arrival rate of customer requests. One or many sources of requests produce arrivals to the QN (Franks et al. 2013). One or many sink nodes takes in the jobs corresponding to requests from the QN. Moreover, the number of customer requests that disseminate in an open QN in any time is not determined. Closed workload of QN in state is totally determined by constant number of customers that disseminate in the QN. The QN is stated to be closed because there is no entry or exit points (Tertilt & Krcmar 2011). Terminal is a special node that represents the customers. A specific amount of time after receiving the response called thinking time, whereas request is generated and leaves the terminal node. Thus, the number of requests/jobs that disseminate in a closed QN is constant at any time.

In each cases open or closed QN the classes of customers have to be identified. From Fig 2.2 a customer class is a group of customer that sends the same type of requests to the QN. In other words, a specific class of customer follows the same service rate and routing probability distributions.

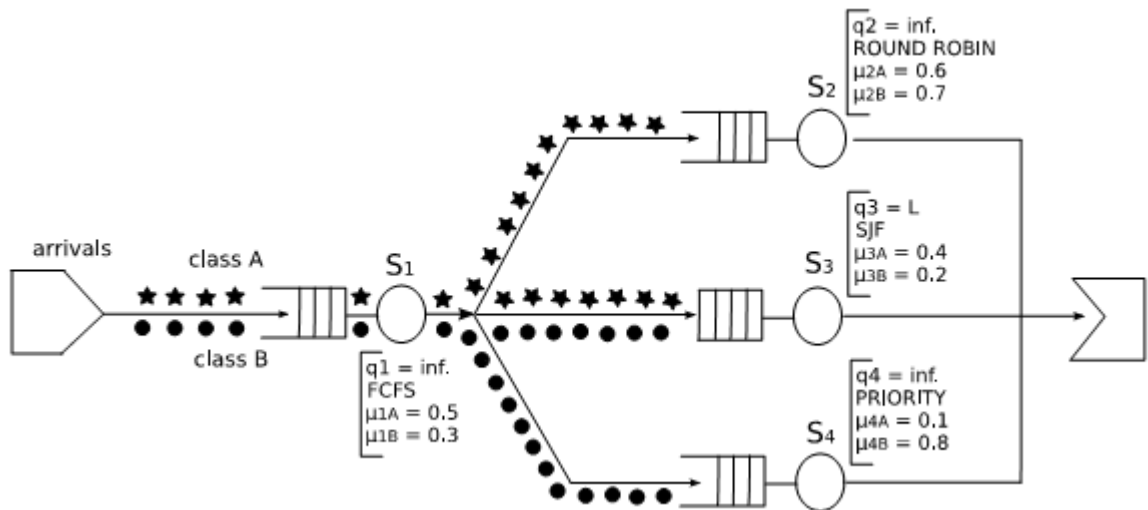


Figure 2.2: Queuing network example with multiple classes(Cortellessa et al. 2005)

Product-Form QN is a special type of QN that can be simply solved by stable algorithm to evaluate performances indices. For example algorithm such as Mean Value Analysis (MVA) and Convolution that have a polynomial computational complexity in the

number of QN components. So as to classify the Product-From class as QN it must confirm a set of attributes on its types of node, scheduling policy, and customer class.

## **2.5 UML Profile for Schedulability, Performance and Time**

OMG has been adopted the Schedulability, Performance and Time Profile as standard profile to identify the requirements for UML models analysis (Tribastone et al. 2010). It states the standard approach to model physical time, timing specifications, resources, scheduling, and hardware and software infrastructure. It enables software architects to give quantitative information in UML models allowing quantitative analysis and modeling assessment. The analysis approaches considered in the profile are scheduling analysis and performance analysis depends on the theory of queuing (Moniem 2015).

### **2.5.1 *PAprofile*: Stereotypes and Tagged Values**

Performance analysis is basically instance based and it is implemented to models that collect either actual or approximated execution runs of software systems comprises of group of instances.

### **2.5.2 <<*PAcontext*>> Stereotype**

A performance context defines one or more important scenarios to explore different dynamics situations containing special set of resources and with critical performance. Performance context comprises of a set of scenarios, their relative workload, and set of resources. Fig.2.3 presents a high level context where two software resources, specifically Client and Server components, communicate to provide Browse Cart functionality when ordered by Customers.

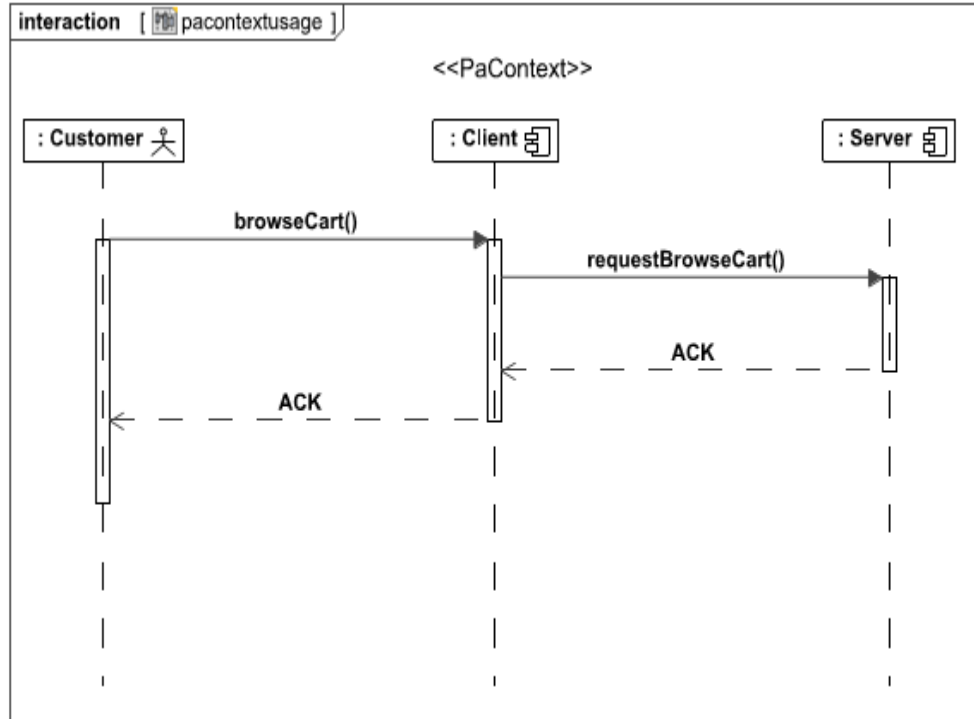


Figure 2.3: Example of <<PAcontext>> annotation (Cortellessa et al. 2005)

### 2.5.3 <<PAclosedLoad>> and <<PAopenLoad>> stereotypes

A scenario is executed by user class with workload intensity; this workload could be either open or closed. The modeling of stereotypes open and closed can be implemented at the first step in performance context.

Open workload is defined as infinite number of requests which get into the queuing system at specific rate, and population that change over the time. Customers that finished their service depart the system model. The model for an open workload is <<PAopenLoad>>. In Fig.2.4 an open workload is annotated on the Browse Cart functionality using the <<PAopenLoad>> stereotype.

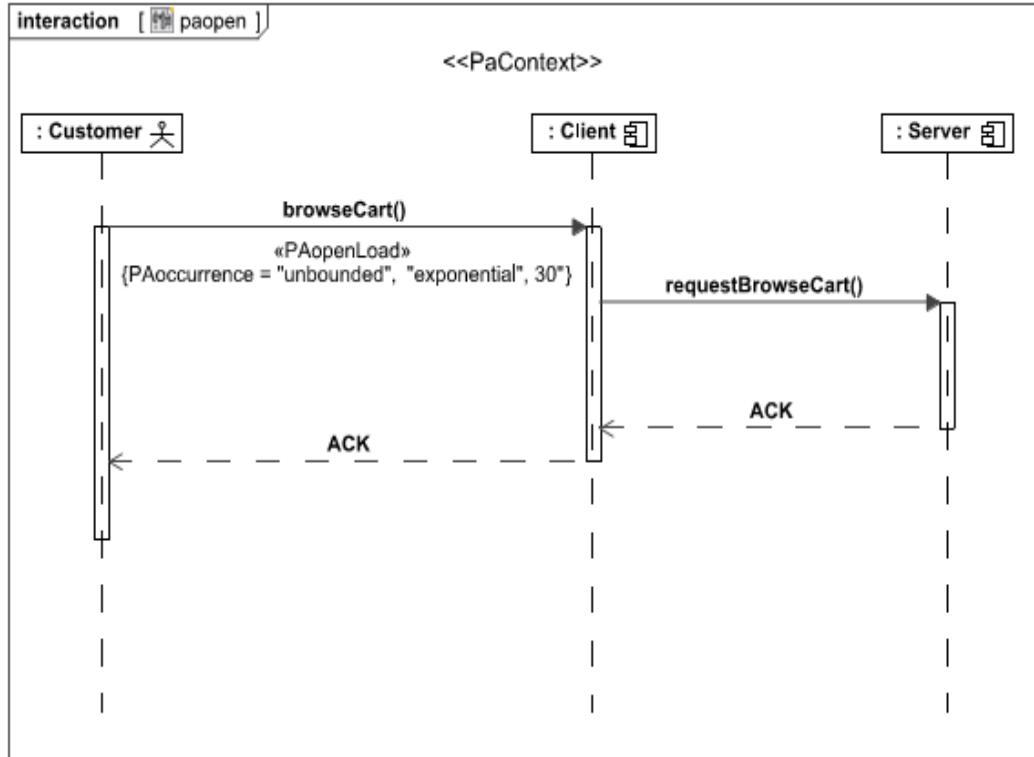


Figure 2.4: Example of *<<PAopenLoad>>* annotation (Cortellessa et al. 2005)

Closed workload has a specific number of jobs (population) which pass through executing of the scenario, time outside the system between each two request (Think Time), and time between the end of one response and the next request (externalDelay). The closed workload modeled as *<<PAClosedLoad>>*. In Fig. 2.5 a closed workload is annotated on the Browse Cart functionality using *<<PAClosedLoad>>* stereotype. It specifies a constant number of 3000 jobs, each job takes an assumed mean external delay time of 1 millisecond.

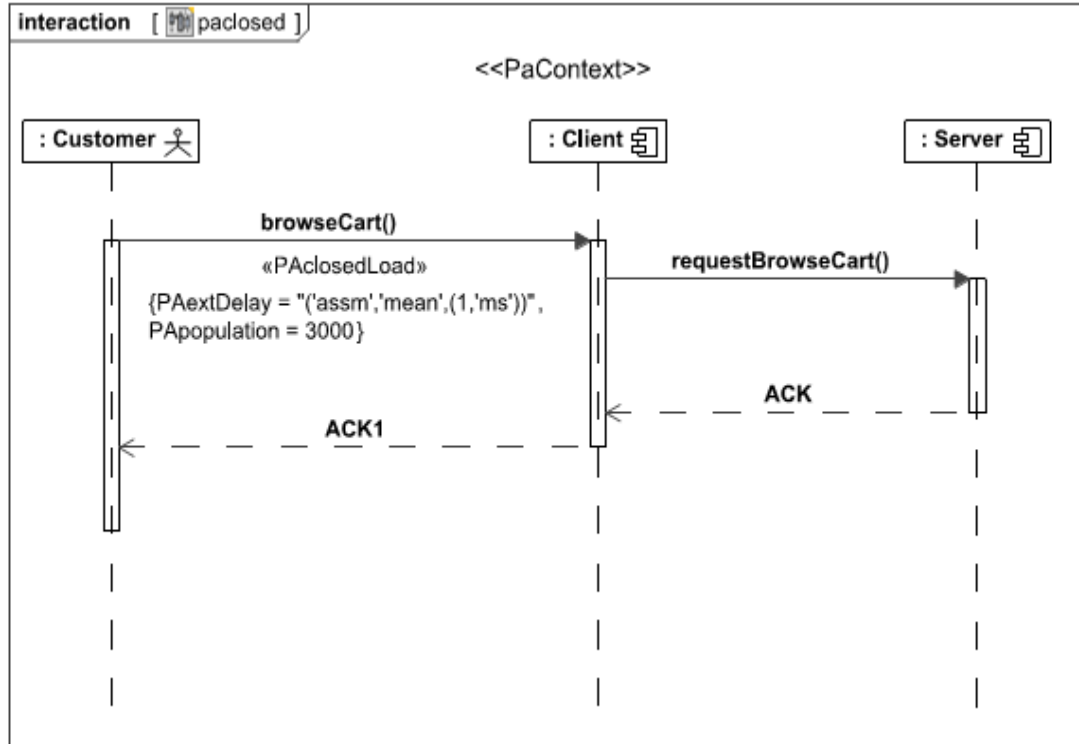


Figure 2.5: Example of *<<PAclosedLoad>>* annotation (Cortellessa et al. 2005)

### 2.5.4 *<<PAstep>>* Stereotype

Scenarios are comprised of (steps) with successor and predecessor relationships which might include fork, join, and loops. A step is defined as simple operation, or sub scenario. A scenario step represents an increment in the running of a scenario and it might be use resources to perform function of the scenario. Moreover, a step consumes a finite time to execute (executionTime). At last, a scenario step may have performance attributes and might be determine the resource needed in the accomplishment of the step. In Fig. 2.6 the request Browse Cart is annotated using the *<<PAstep>>* stereotype. The numerical value assigned to the *PAdemand* tag states a measure mean service time demand of 5 milliseconds.



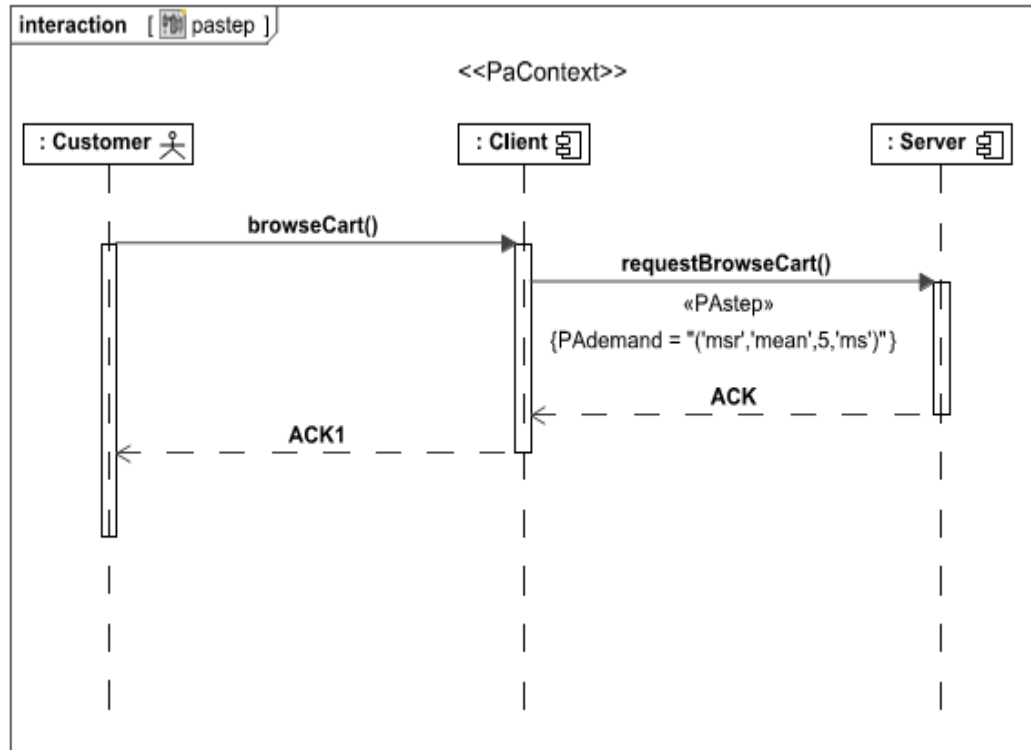


Figure 2.6: Example of `<<PAstep>>` annotation (Cortellessa et al. 2005)

### 2.5.5 `<<PAhost>>` and `<<PAresource>>` Stereotypes

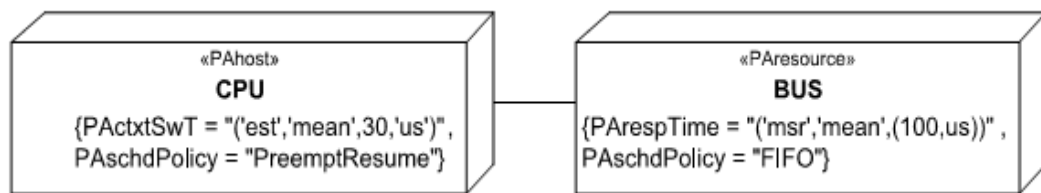
A Resource models is a general view of passive or active resource, these resources take part in one or more scenarios of the performance context. Resources are presented as servers and keep information about throughput, utilization and scheduling Policy. The active resources are the normal servers in performance models, and have service times. Furthermore, `<<PAhost>>` stereotype models a processing resource.

Passive resources are gained and freed during scenario. In addition to properties it inherits from the Resource entity, it has capacity stating the number of simultaneous users and time holding (`AccessTime` and `waitingTime`).

Performance measures for a system involve waiting times, resource utilization, execution demands and response time that is the actual time to execute scenario step. Each measure might be: a required value, coming from the system requirements or from performance budget based on them, an assumed value, based on experience, an estimated

value, calculated by a performance tool and reported back into the UML model, or a measured value.

In Fig. 2.7 a CPU node is an active resource annotated with the `<<PAhost>>` stereotype. It implements a Preemption-Resume scheduling policy to each step executed. The estimated mean context switching time required by such an active resource is 30 microseconds. The CPU communicates through a BUS that represents a passive resource that dispatches information in 100 microseconds, while applying FIFO scheduling policy.



**Figure 2.7: Example of `<<PAhost>>` and `<<PAresource>>` annotation (Cortellessa et al. 2005)**

## 2.6 Machine Learning

Before moving into formal definition of machine learning, in Information and Communication Technology (ICT) field, the two terms that comprises up machine learning; machine or computer and learning. Defining these terminologies will be guide to select the suitable machine learning definition for this research.

Computer is a machine for performing calculation; it accepts data, processes them and provides information based on a sequence of instructions on how to be processed. Moreover, learning can be defined as a process of acquiring modifications in existing skills, knowledge and habits through experience and practice (Omary & Mtenzi 2009).

Alpaydin (Omary & Mtenzi 2010) defines machine learning is the ability of the computer program to develop a new knowledge from available or non-available examples for the reason of enhancing performance criteria. Software engineers and researchers have been started using machine learning techniques in the area of quality of service assessment

and prediction. Furthermore, machine learning has proved its efficiency to assist and optimize model-based performance prediction (Moniem 2015).

Over the past 50 years, machine learning as any growing field of study has grown hugely. The growing attention in machine learning is driven by two factors as per Alpaydin:-

- (a) Removing tedious human work.
- (b) Reducing cost.

Machine learning techniques, when applied to different fields such as in medical diagnosis, bio-surveillance, speech and handwriting recognition, computer vision and detecting credit card fraud in financial institutions, have confirmed to work with huge amounts of data and provide results in a matter of seconds.

### **2.6.1 Machine Learning Categories**

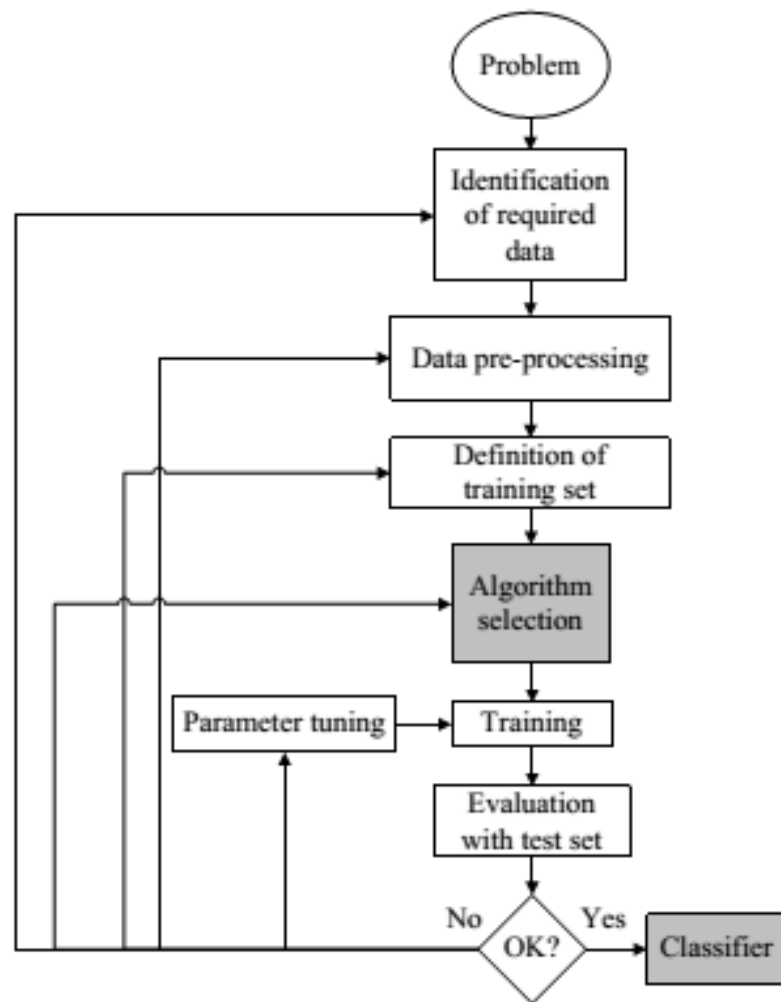
Machine learning can be categorized into two main groups, supervised learning and unsupervised learning machine learning (Anon n.d.). The two different groups are related with different machine learning algorithms that represent how the learning approach works (Kotsiantis 2007).

#### **(a) Supervised Learning**

Supervised learning comprises of algorithms that realize from externally provided cases to output general hypothesis which then make predictions about future instances (Omary & Mtenzi 2010). In general, by using supervised learning there is a presence of outcome variable to guide the learning process. There are several supervised machine learning algorithms such as decision trees, K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and random Forest (Omary & Mtenzi 2009). Fig 2.8 describes the process of applying supervised machine learning to a real world problem:

- (1) Collecting the dataset.
- (2) Data processing and data preprocessing.

- (3) Defining and providing training dataset.
- (4) Selecting the algorithm.
- (5) Training and building the model.
- (6) Evaluation and assessment with test set.
- (7) If the evaluation is ok? Yes go to step 8, else go to step 9.
- (8) Perform the classification operation. Go to step 10.
- (9) Tune the parameters and go to step 5.
- (10) End.



**Figure 2.8** the process of supervised learning

## **(b) 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 (Moniem 2015). This means no guidance is available and learning must perform heuristically by the algorithm examining different training data.

### **2.6.2 Machine Learning Techniques**

There are various machine learning techniques can be used depending on the application domain; four techniques are applied on the research that: Naïve Bayes, k-nearest neighbor, support vector machines and multivariate regression. These four techniques are used to give understanding of using machine learning techniques in the area of model based performance and resource utilization prediction (Omary & Mtenzi 2009).

#### **(a) Multivariate Regression**

Multivariate regression (MR) is the most commonly used technique for modeling the relationship between two or more independent variables and dependent variable by fitting a linear equation to observed data (Guo et al. 2012). The general form of a MR can be given be:

$$\hat{y}_i = a_0 + a_1x_{i1} + \dots + a_kx_{ik} \quad (1)$$

$$y_i = a_0 + a_1x_{i1} + \dots + a_kx_{ik} + e_i \quad (2)$$

where  $x_{i1}, \dots, x_{ik}$  are the independent variables,  $a_0, \dots, a_k$  the parameters to be estimated,  $\hat{y}_i$  the dependent variable to be predicted,  $y_i$  the actual value of the dependent variable, and  $e_i$  is the error in the prediction of the  $i^{\text{th}}$  case.

#### **(b) Naïve Bayes Classifiers**

Naïve Bayesian (NB) are very simple Bayesian networks which are consist of directed acyclic graphs with only one parent and several children with a strong assumption

of independence among child nodes in the context of their parent (Kotsiantis 2007). Moreover, the independence model of Naïve Bayes is based on estimating:

$$R = \frac{P(i|X)}{P(j|X)} = \frac{P(i)P(X|i)}{P(j)P(X|j)} = \frac{P(i) \prod P(X_r|i)}{P(j) \prod P(X_r|j)} \quad (3)$$

Comparing these two probabilities, the larger probability indicates that the class label value that is more likely to be the actual label (if  $R > 1$ : predict  $i$  else predict  $j$ )(Mohanty 2012). Since the Bayes classification algorithm uses a product operation to compute the probabilities  $P(X, i)$ , it is especially prone to being especially affected by probabilities of 0. This can be avoided by using Laplace estimator, by adding one to all numerators and adding the number of added ones to the denominator.

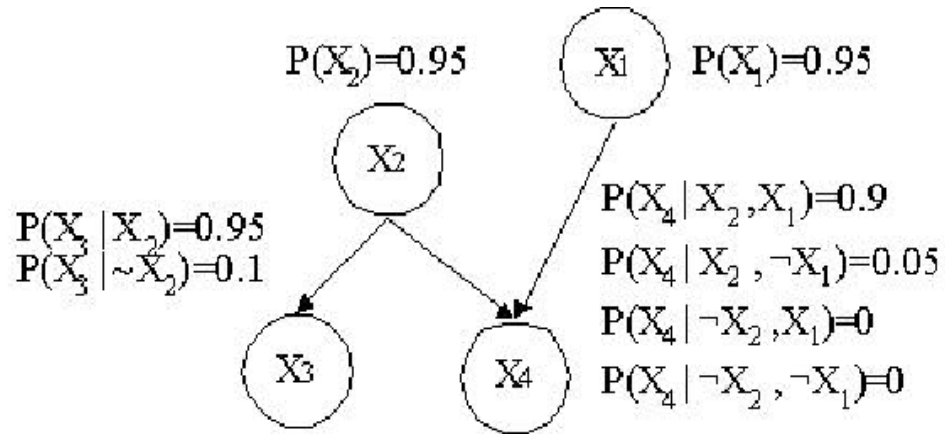
The basic independent Bayes model has been modified in various ways in various ways in attempts to improve its performance. Attempts to solve the independence assumption are mainly based on adding more edges to include some of the dependencies between the features. In this case, the network has the limitation that each feature can be related to only one other feature. Semi-naïve Bayesian classifier is another important attempt to avoid the independence assumption. In which attributes are partitioned into groups and it is assumed that  $x_i$  is conditionally independent of  $x_j$  if and only if they are in different groups.

The main important advantage of the naïve Bayes classifier is its small computational time for training. Furthermore, since the model has the form of a product, it can be converted into a sum through the use of logarithms with significant consequent computational advantages. If a feature is numerical, the normal procedure is to discretize it during data pre-processing (Kotsiantis 2007).

### (c) Bayesian Networks

A Bayesian Network (BN) is a graphical model for probability relationships among a set of variables. The Bayesian network structure  $S$  is a directed acyclic graph (DAG) and the nodes in  $S$  are in one-to-one correspondence with the features  $X$ . The arcs represent

casual influences among the features while the lack of possible arcs in  $S$  encodes conditional independences. Furthermore, a node is conditionally independent from its non-descendants given its parents ( $X_1$  is conditionally independent from  $X_2$  given  $X_3$  if  $P(X_1/X_2, X_3) = P(X_1/X_3)$  for all possible values of  $X_1, X_2, X_3$ ).



**Figure 2.9** the structure of Bayes Network

Naïve Bayes models are similar named for their “naive” assumption that variables  $X_i$  are mutually independent given “special” variable  $C$ . The joint distribution is then given compactly by:

$$P(C, X_1, \dots, X_n) = P(C) \prod_{i=1}^n P(X_i | C) \quad (4)$$

From Fig 2.9 the univariate conditional distributions  $P(X_i | C)$  can take any form. If the variable  $C$  is observed in the training data, naïve Bayes can be used for classification by assigning test example  $(X_1, \dots, X_n)$  to the class  $C$  with highest  $P(C | X_1, \dots, X_n)$  (Mohanty 2012). When  $C$  is unobserved, data points  $X_1, \dots, X_n$  can be clustered by applying the EM algorithm with  $C$  as the missing information; each value of  $C$  corresponds to a different cluster, and  $P(C | X_1, \dots, X_n)$  is the point’s probability of membership in cluster  $C$ . Naïve Bayes models can be viewed as Bayesian networks in which each  $X_i$  has  $C$  as the sole parent and  $C$  has no parents. A naïve Bayes model with Gaussian  $P(X_i | C)$  is equivalent to a mixture of Gaussian with diagonal covariance matrices. While mixtures of Gaussians are widely used for density estimation in continuous domains, naïve Bayes models have seen

very little similar use in discrete and mixed domains. However, they have some notable advantages for this purpose. In particular, they allow for very efficient inference of marginal and conditional distributions. To consider this, let  $X$  be the set of query variables,  $Z$  be the remaining variables, and  $K$  be the number of mixture components. The marginal distribution of  $X$  by adding out  $C$  and  $Z$ :

$$P(X = x) = \sum_{c=1}^k \sum_z P(C = c, X = x, Z = z) \quad (5)$$

$$\begin{aligned} &= \sum_{c=1}^k \sum_z P(c) \prod_{i=1}^{|X|} P(x_i|c) \prod_{j=1}^{|Z|} P(z_j|c) \\ &= \sum_{c=1}^k P(c) \prod_{i=1}^{|X|} P(x_i|c) \prod_{j=1}^{|Z|} \sum_{z_j} P(z_j|c) \\ &= \sum_{c=1}^k P(c) \prod_{i=1}^{|X|} P(x_i|c) \end{aligned}$$

Where the last equality holds because, for all  $j$ ,  $\sum_{z_j} P(z_j|c) = 1$  thus the non-query variables  $Z$  can simply be ignored when computing  $P(X = x)$ , and the time required to compute  $P(X = x)$  is  $O(|X|k)$ , independent of  $|Z|$ . This contrasts with Bayesian network inference, which is worst-case exponential in  $|Z|$ . Similar considerations apply to conditional probabilities, which can be computed efficiently as ratios of marginal probabilities:

$$P(X = x|Y = y) = P(X = x, Y = y)|(Y = y) \quad (5)$$

A little bit richer model than naïve Bayes which still allows for efficient inference is the mixture of trees, where, in each cluster, each variable can have one other parent in addition to  $C$ .



#### (d) K-Nearest Neighbor (KNN)

K-Nearest is one of the methods referred to as instance-based learning which considered as supervised learning category (Keung & Nguyen 2010). KNN works by simply storing the training data set, and when a new instance is used, a set of similar related instances that are neighbors is retrieved from the training dataset set and used to classify the new instance (Garcia et al. 2008). Classification is useful to take more than one neighbor into account and then referred to as k-nearest neighbor (Chen & Ma 2013).

Let  $R = \{r_1, r_2, \dots, r_m\}$  be a set of  $m$  reference points in a  $d$  dimensional space, and let  $Q = \{q_1, q_2, \dots, q_n\}$  be a set of  $n$  query points in the same space. The  $k$  nearest neighbor problem consists in searching the  $k$  nearest neighbor of each query point  $q_i \in Q$  in the reference set  $R$  given a specific distance. Fig. 2.10 illustrates the KNN problem with  $k = 3$  for a set of points in a two dimensional space.

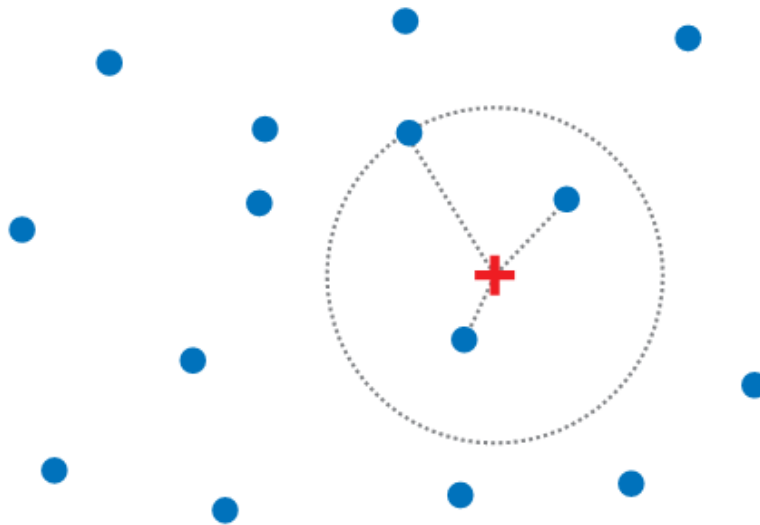


Figure 2.10 Illustration of KNN search problem for  $k = 3$

#### (E) Support Vector Machines (SVMs):

Support Vector machine (SVMs) finds separating hyper planes between training instances that maximize the margin and minimize the classification errors. Margins sometimes known as geometric margin defined as distance between the hyper planes

separating two classes and the closest data points to the hyper planes(Abe 2015). The SVM algorithm is able to work with both linearly and separable problems in classification and regression tasks(Islam 2013).

SVMs were developed 1995 for binary classification. Their approach may be roughly sketched as follows:

- **Class separation:** basically, we are looking for the optimal separating hyper plane between the two classes by maximizing the margin between the class's closet points(Garcia et al. 2008). Fig.2.11shows the points lying on the boundaries are called support vectors, and the middle of the margin is our optimal separating hyper plane

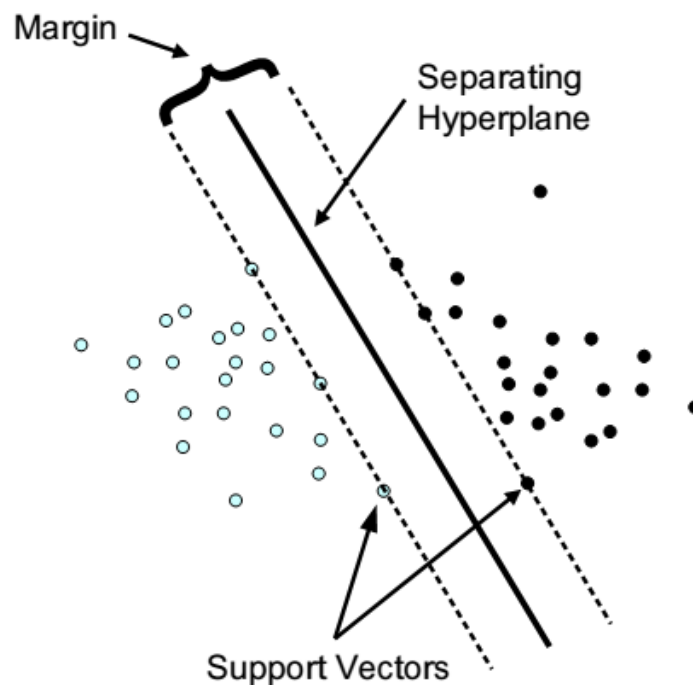


Figure 2.11 SVM Classification

- **Overlapping classes:** data points on the wrong side of the discriminate margin are weighted down to reduce their influence.

- **Nonlinearity:** when we cannot find a linear separator, data points are projected into and usually higher dimensional space where the data points effectively become linearly separable.
- **Problem solution:** the whole can be formulated as a quadratic optimization problem which can be solved by known techniques. A program able to perform all these tasks is called support vector machine.

Several extensions have been developed, the ones currently included in lib-svm are:

- **V-classification:** this model allows for more control over the number of support vectors by specifying an additional parameter  $v$  which approximates the fraction of support vectors.
- **One-class-classification:** this model tries to find the support of a distribution and thus allows for outlier detection.
- **Multi-class classification:** basically, SVMs can only solve binary classification problems. To allow for multi-class classification, lib-svm use the one-against-one technique by fitting all binary sub-classifiers and finding the correct class by a voting mechanism.
- **$\epsilon$ -regression:** the data points lie in between the two borders of the margin which is maximized under suitable conditions to avoid outlier inclusion.
- **V-regression:** with analogue modifications of the regression model as in the classification case.

## 2.7 Related Work:

From the literature, Ganapathi used machine learning on software performance prediction (Moniem 2015). She proposed a machine learning technique to predict/optimize multi components, parallel system utilization and performance. The proposed technique gathers the correlation between a workload's pre-execution characteristics configuration parameters, and post-execution performance observations. Finally, the correlation has been used for performance prediction and optimization. To prove the methodology, she presents

three cases on storage and computer-based parallel systems. The outcomes suggest the use of machine learning based performance modeling to improve the quality of system management decisions.

The above work is very useful representation of using statistical machine learning to predict the performance of software systems; however, the approaches focused on software systems that already designed and implemented not that are at the early modeling stage.

Archana has worked on the area of model based performance prediction by designing a complementary tool called Software Performance Risk Assessment (SPRA). The tool performs on scenario based performance risk assessment of a model by analyzing annotated UML diagrams (Radhakrishnan & Virginia 2007). However, she applied her case studies on case studies with small number of users which may not give a broad solution. Our approach use machine learning techniques with thousands of users.

Dubach and et al. used machine learning technique to explore the good compiler architecture design(Singh et al. 2007). He designed two performance models and applied them to increase the efficiency of searching the design space for micro architecture. Models predict performance metrics such as processor cycles, energy consumption, and the trade-off of the two characteristics.

Malhotra and et al. have employed machine learning to measure software maintainability. Number of the word “change” is observed over a period of three years on a dataset. The researchers shown that when using Naïve Bayesian algorithm the classification performs better than other machine learning techniques (Islam 2013).

Ipek and et al, used multilayer neural network, the network trained on input data collected from execution on targeted platform (Li et al. 2009). This approach is useful for predicting many aspects of performance and it takes full system complexity. The study focuses on the high performance parallel application SMG2000. The model has predicted performance within error 5% to 7% across a large, multidimensional parameter space.

However, the work of Ganapathi, Dubach, Malhotra, and Ipek applied on system that already designed and implemented, but they didn't start the performance prediction

process of software from the early modeling stage. The proposed approach proposes building and evaluating performance model, so that if the model gives reasonable performance indicators model and then we will continue constructing the system, otherwise the change will be made on the model itself not on the final product.

UML profile for performance, Schedulability, and time has been announced by Object Management Group (OMG) as standard specification mechanism (Brunnert & Krcmar 2015). Starting from model-to-model transformation we have to take annotation tags and stereotypes proposed in the profile, and the ability to add more specifications. UML annotations concepts are enough to describe performance attributes of software systems. Moreover, extending the queuing theory with the machine learning is one of a new concept we have introduced in our research.

The main contribution of the research is to combine response time, system throughput, resource utilization, and performance risk prediction from annotated UML models. The approach started at early software development stage and before implementing the software. In real life, the exact quantitative performance prediction is not enough in specific situations, such as e-business systems where the delay in response time may lead to losing thousands of customers. In addition, software engineers looking for feedback to make decisions on leveling the consequences of performance risk such as low, medium, or high performance risk.

We introduced a method to predict system response time, system throughput, resource utilization, and performance risk from the software system modeled with annotated UML diagrams based on machine learning techniques. Performance risk can be categorized as: time related or resource related and the research merged them together in new proposed approach. Moreover, we can predict resource utilization, response time, and system throughput by formulating the case as statistical problem and we use multivariate regression technique. Similarly, we can predict and categories the performance risk level as (low, medium, or high) by formulating the case as classification prediction problem. To illustrate the method clearly we presented the steps in Fig.1. In the next chapters we will illustrate the methodology by using a complete three case studies and compare the results.

## **2.8 Summary**

In this chapter, the background of performance risk prediction has been presented. The relation of UML with performance notation discussed clearly as it considered important part of our approach. Queuing network models paly crucial role in the modeling of software system. Equally important, we use machine learning to extend QNM in order to deal intelligently with performance risk prediction and assessment.

## CHAPTER III

### **Methodology for Model-based Prediction of Resource Utilization and Performance Risk**

#### **3.1 Introduction**

As a general definition, a performance measure how effective is a software system with respect to time constraints and allocation of resource. Standard performance indices stated by ISO 926 are response time, throughput and utilization. Response time defined as the end to end time that a job spends to traverse a certain path within the system (Shoaib & Das 2011). Throughput is defined as the number of jobs that can be completed per unit of time by a specific part of the software system. Moreover, resource utilization defined as the percentage of time that a certain part of the system is busy working.

#### **3.2 Challenges in Performance Prediction**

Figure 3.1 shows a simple view of our performance prediction model. We construct a model that represents our software system. Furthermore, the software system model will be run and the input workloads observed. Meanwhile, the performance measurement such as response time, system throughput, and resource utilization metrics will be recorded to build large dataset.

Our main objective is to build a model that represents the real system and generate large dataset by running the model on different workload. In order to predict the resource utilization and performance risk level for the system we decided to use machine learning as intelligent technique. We build a model based on the generated dataset and then we use this model to predict new instances base on the previous workload. However, there are several obstacles and constraints we must overcome to successfully build performance model for the system.

**Challenge 1:** Predicting software system performance requires deep knowledge of the system under construction. Consequently, one of the software engineering myths, that normal developer cannot measure and predict the performance until the program finished and run.

**Solution 1:** Use a black-box modeling approach with performance metrics datasets to predict software quality before software implemented and put on production environment.

**Challenge 2:** Performance modeling tools do not include performance risk prediction concept and most of these tools only present the results of the analysis.

**Solution 2:** Extend the performance queuing modeling tool with machine learning in order to predict and understand the performance of the software system before constructed.

**Challenge 3:** Software system performance risk prediction measured normally quantitatively, which may have less meaning for the software engineers.

**Solution 3:** Define technique that can capture both quantitative and qualitative parameters to help software engineers to decide changes if any at early design phases.

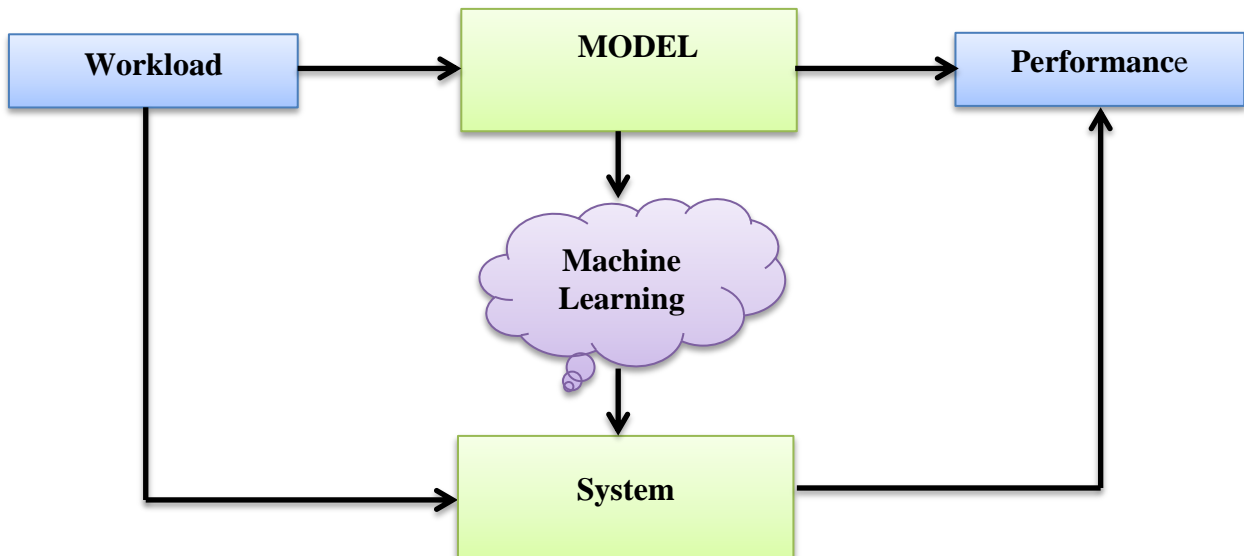


Figure 3.1 our goal is to mimic the behavior of real system by using model



Adjusting the performance modeling problems with these challenges in consideration, and given black box system, our goal is to utilize machine learning techniques to assist software engineers in performance analysis. Our dataset contains recorded performance indices measurements with different workload in order to predict response time or throughput or resource utilization. The dataset also represents the measurement parameters classified into three performance risk levels low or medium or high performance risk. Machine learning provides a variety of algorithms that can be used for performance prediction and performance risk assessment.

### **3.3 The steps of the Approach**

The proposed approach depends mainly on prediction of the performance and resource utilization of a software system at early stages of SDLC. Fig.3.2 presents the approach which follows the stages:-

**STEP 1: Annotated UML diagrams will be designed to describe the software system as follows:**

- (a) Use Case: each actor in a Use Case diagram may represent a stream of requests arriving at the system. There may be unlimited sequence of requests (open workload), or fixed population of users requiring service from the system service from the system (closed workload)
- (b) Deployment diagrams: Deployment diagrams are used to model the physical resources available in the system. Each resource is represented by a node in the deployment diagrams (Moniem 2015).
- (c) Sequence diagrams: shows interactions consisting of a set of objects and the messages sent and received by those objects. Sequence diagram address the dynamic behavior of a system with special emphasis on the chronological ordering of messages.

**STEP 2: Transform Annotated UML Diagrams into Performance Model**

In this step the annotated UML models will be mapped into performance model. We used Java Modeling Tool (JMT) as performance model. JMT is a free open

source suite implementing several algorithms for the exact, asymptotic and simulative analysis of queuing network models (Rabta et al. n.d.).

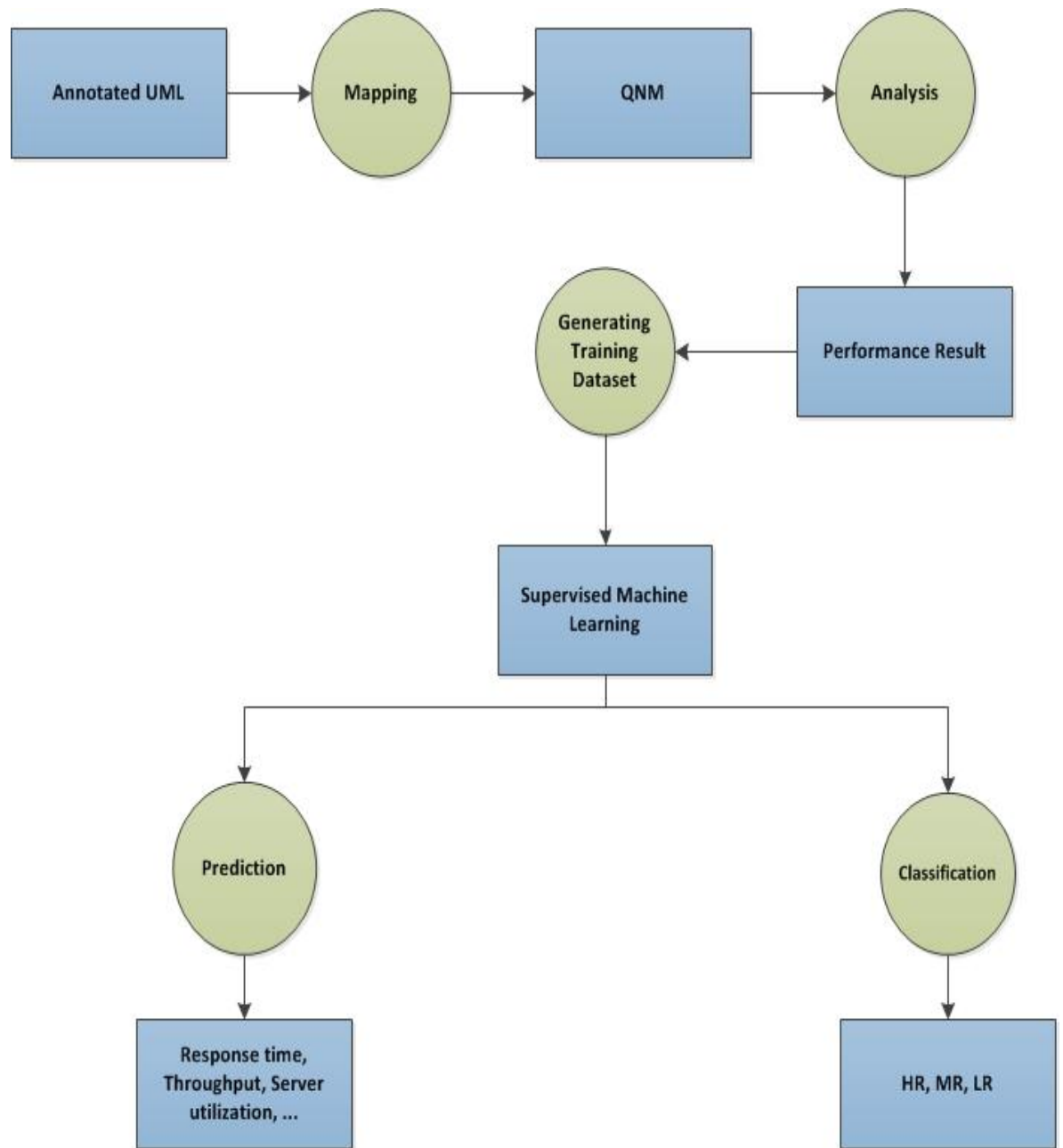


Figure 3.2 the Framework of the Approach

### **STEP 3: Running the Performance Modeling Tool and Generate Training Dataset**

In order to generate suitable size of a training dataset JMT will be run many times with different workloads. We mean by workload here the number of users that concurrently using the system and the type of operations they are doing on the system.

### **STEP 4: Use Machine Learning Techniques to:**

#### **(a) Predict Performance indices:-**

To predict the performance of a software system we use multivariate regression machine learning technique. Moreover, performance defined as per ISO 9126 are three indices of a system, which are the response time, throughput, and resource utilization (Moniem 2014).

- Response time is the measure of the time between the end of a request to a service and the beginning of the time server response.
- Throughput defined as the number of requests application can process at a given period of time.
- Resource utilization of an application is a time that application resource is busy by performing requests expressed usually as percentage.

We build the model by using machine learning – regression, and then we can use the model to predict any of the above mentioned performance indices (response time, system throughput, and resource utilization) based on a training dataset(Ram et al. 2011a).

#### **(b) Assess The Performance Risk :-**

After performance indices have been predicted, the second step is to categories level of the performance risk. However, Software Performance risk is defined as undesired event that prevents the software providing full functionality under all possible environment conditions (Radhakrishnan & Virginia 2007). Early identification of software performance metrics such as system response time, resource utilization, and system

throughput is a key step to manage performance risk of software system before implementation (Moniem 2015). Software Performance requirements can be categorized into two types:

- Time-related performance requirement, which means the completion time of a specific operation must be less than a certain threshold (ex. response time for product purchasing must be less than 8 seconds).
- Resource-related performance requirement, which means the utilization of a specific resource must fall into a certain range (ex. sever utilization must be less than 80% when 500 concurrent users logged into the system concurrently).

In our approach we categorized performance risk into three levels low, medium or high performance risk, according to the degree of the acceptance by user. Whether that software system will meet the user requirement from the user perspective or either time related performance or resource related performance requirement.

**Table 3.1 attributes of the Performance dataset**

<b>Attribute</b>	<b>Description</b>	<b>Units</b>
#user	No of users currently logged onto the software system	
resp	System response time represents the time taken to send and receive a response	ms
thro	System throughput represents number of requests for a given period of time	%
web	Web server utilization represents the percentage of the web server in operation	%
app	Application server utilization represents the percentage of server in operation	%
stor	Database server utilization represents the percentage of server in operation	%
class	Represents the category of the performance risk either it is low, medium, or high	

### **3.4 Summary**

In this chapter, we introduced a machine learning based approach for performance and risk prediction. Our approach uses statistical machine learning technique to predict performance indices such as response time, system throughput, and resource utilization. In addition, we used three other classification techniques to categories performance risk into one of three levels low or, medium or, high. The strength of this approach comes from extending of queuing network models with machine learning which gives visualization for the relation between workloads, response time, throughput, resource utilization, and performance risk level.

## CHAPTER IV

### Predicting Performance of Hospital Information System

#### 4.1 Introduction

Multi-tier application provides excellent features for designing distributed internet applications; these features include flexible configuration and easy implementation. In reality, the complexity nature of web applications requires their developers to follow one of Software Development Life Cycle (SDLC) methodologies. Normally, users and consumers expect web applications to be highly responsive, for this reason performance of the application is taken at early SDLC phases and followed throughout entire the phases (Shoaib & Das 2011).

For validation, the proposed approach will be applied on a hospital system. The system is proposed and presented by Salvaneschi (Serazzi 2008); the application has been analyzed as part of the IT system of a hospital. The tasks of the application to manage patient history, such as the arrival date in hospital, past diseases, or remarks related to his recovery as daily measured temperature.

We will mention a brief idea of the available functionalities. In Fig.4.1, Fig.4.2 and Fig.4.3 the medicines prescriptions are centrally managed: doctors insert the desired medicine quantity for a certain patient, and the pharmacy of the hospital makes orders for the whole amount of a needed medicine. All the doctors have a personal computer in their office which they use to insert data from visits and prescriptions. The nurses have a device similar to a notebook to insert values of temperature, pressure, etc., while they are next to the bed of the patient. They also get from the system the quantity of

medicine for the patient. Moreover, there are functionalities that give the ability to create a new file for a patient and retrieve a file searching for his name or his bed number (Salih & Ammar 2017).

#### **4.2 The Architecture and Model**

For the hospital system we suppose that the number of customers remains constant which leads to choose a closed queuing model to represent our system. The interaction of the customers with system is provided by a web application with a browser on client side and typical 3-tier architecture on server side. The requests are submitted to a web server, in case of a static page the application response is immediately passed back to the clients, otherwise the web server interacts with an application server that performs some queries on a backend database and processed data are passed back to the web server. Finally, clients receive the dynamically generated page.

The storage consists of three database servers in work parallel. The attribute of a job to one of the storage servers is done by a load-balancer in a random way. The distribution of the requests among the three servers is uniform. We suppose negligible the delay introduced by the load-balancer in the reasonable hypothesis of a minimal processing inside the load-balancer.

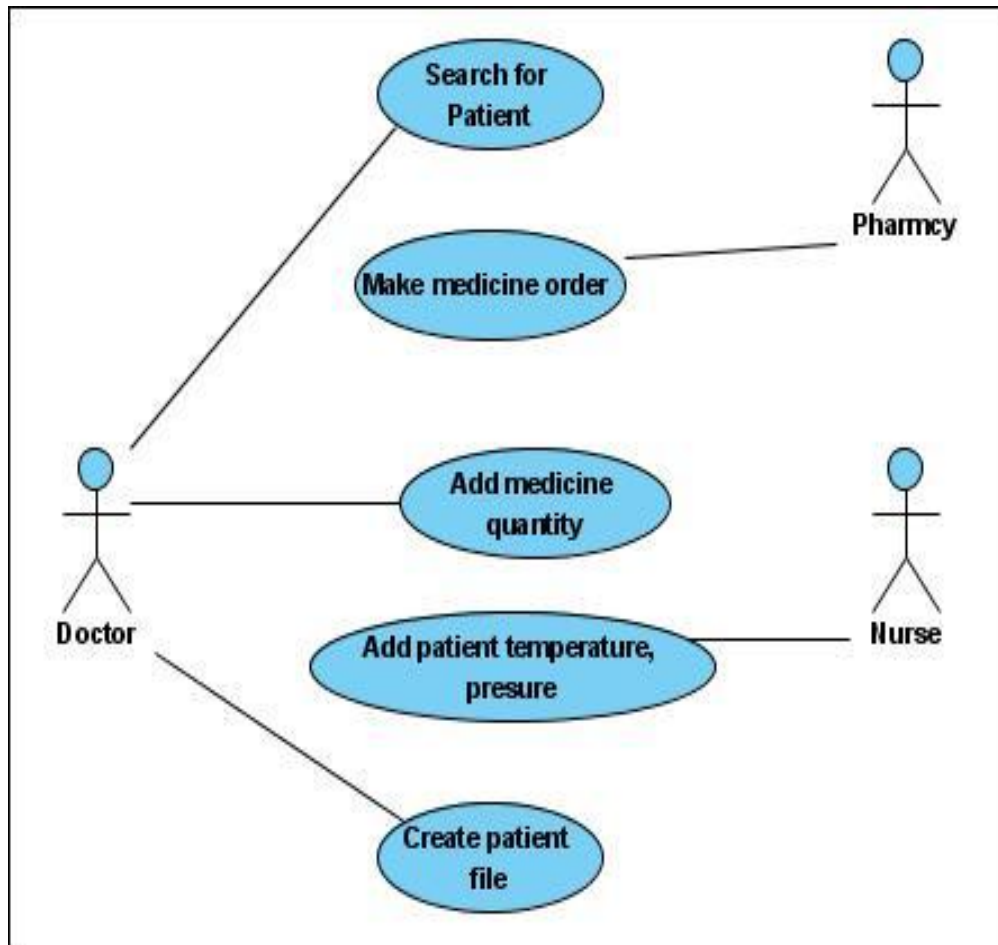
They grouped the requests arriving at the system in two groups: the requests for a database search and the requests for a database modification. The two groups are modeled by classes with different service time for each station. Considering the point of view of the database they refer to the search-class and the modification-class as the Heavy Load class and the Light Load class. In table 4.1 the service demand are reported for each station in the system and for each class. The different values of the two classes are due to the different behavior of the two types of users

**Table 4.1 Service demands time in milliseconds for each station in the system.**

	<b>Light Load</b>	<b>Heavy Load</b>
Web Server	1.40	1.10
Application Server	2.10	1.50
Storage 1	1.10	2.90
Storage 2	1.20	2.70
Storage 3	1.10	2.80

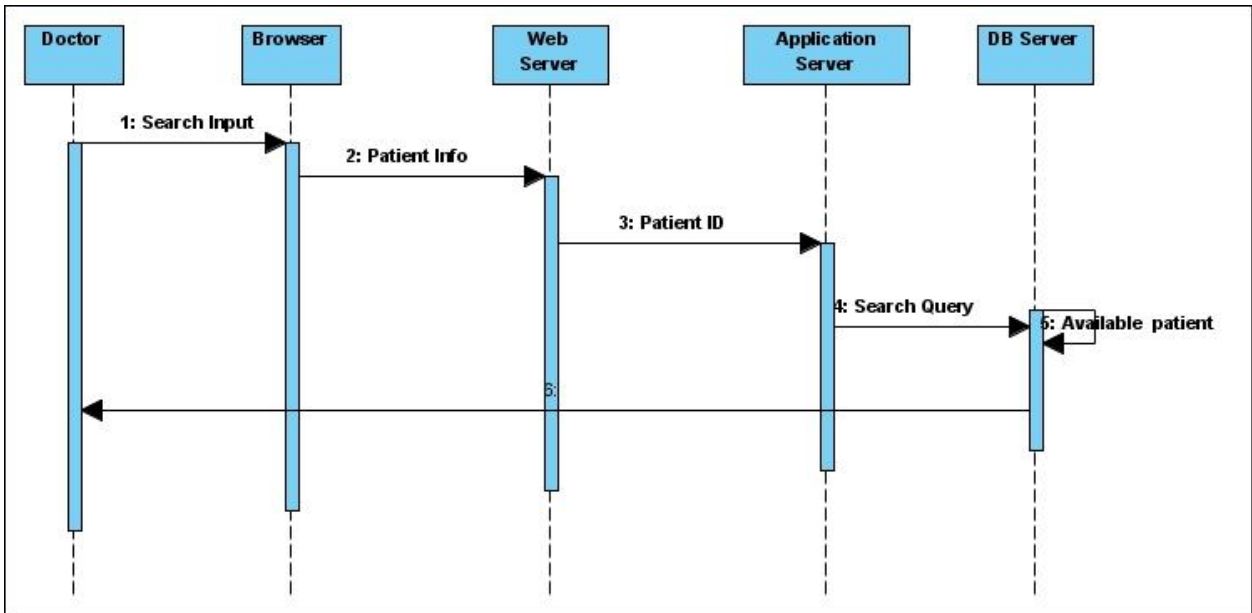
The ratio between the number of Heavy Load requests and the Light Load is about of 3/7. That means if there are 1000 customers using the system the ratio will be result in 300 jobs of light Load and 700 jobs of Heavy Load.





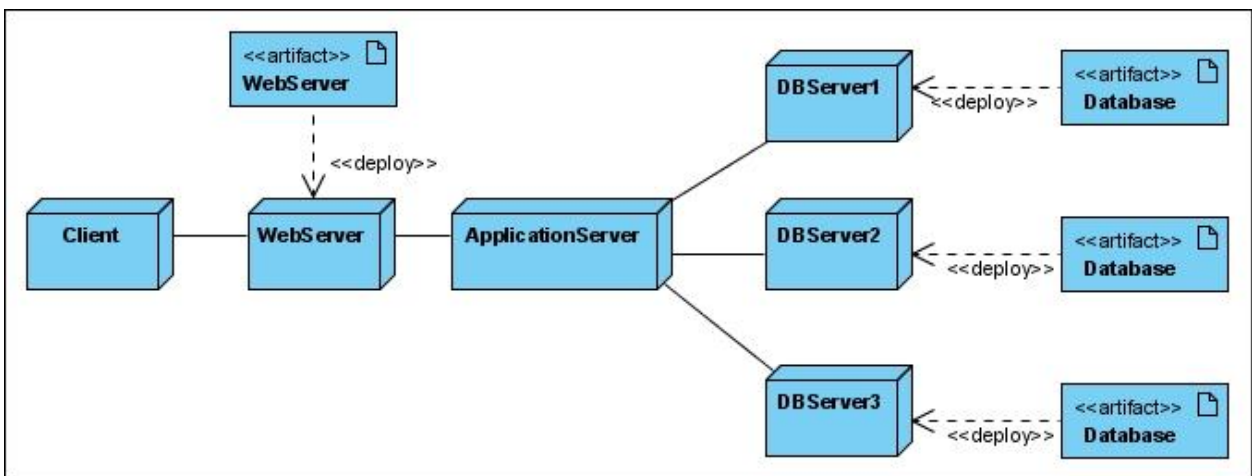
**Figure 4.1 Use Case Diagram for Hospital System**

In Fig. 4.1 we presented the use case diagram for the hospital system. The system has three actors: Doctors who login the system to search for a specific patient, add medicine quantity for patient, and create patient file. Nurse can add patient medical measurements such as temperature and blood pressure. The pharmacist can make medicine order. The use case diagram is very important for our approach as we need to know the type of the system whether it is open or close, the number of scenarios, and determine the critical scenarios.



**Figure 2.2 Sequence Diagram for Hospital System**

Fig.4.2 shows UML sequence diagram which represents the dynamic view of the system architecture. We presented the searching operation scenario for a specific patient. The request moves from client to the web server and from the web server to database server through application server.



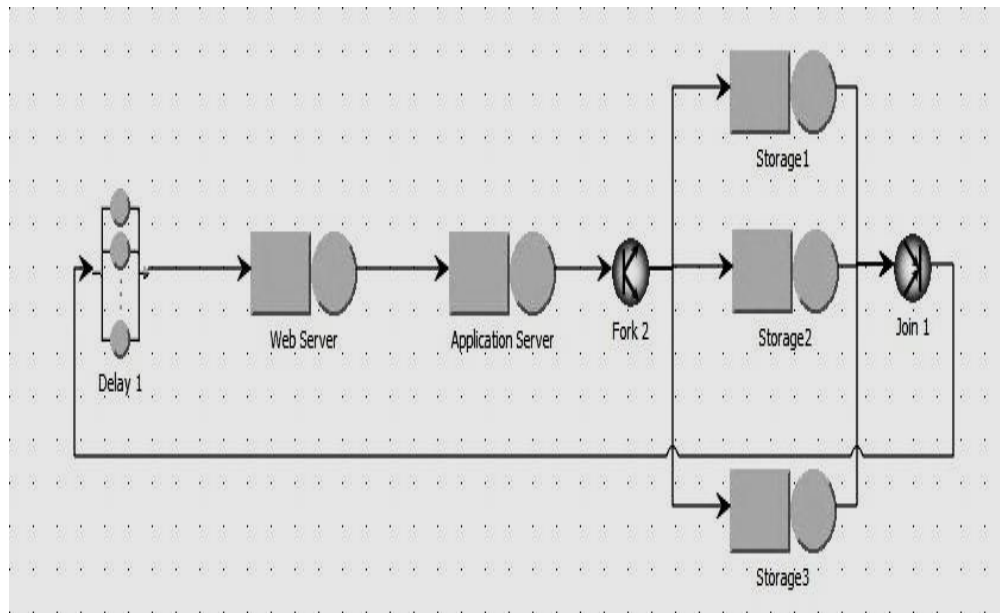
**Figure 4.3 Deployment Diagram for Hospital System**

Fig.4.3 presents the deployment diagram. The deployment diagram explains the distribution of software components on hardware resources. This diagram considered as very important diagram for transformation from UML models to performance model.

### 4.3 Transforming UML diagrams into Performance Model

Fig. 4.4 shows the Queuing Network model for the hospital system after transformation to performance model using queuing model – Java Modeling Tool. The model includes:

- (i) A set of queuing centers (web server node, application server node, and three database servers nodes) representing the hardware resources of the system, a delay center represents the number of users.
- (ii) Two classes of jobs (heavy load and light load), the ratio between the number of heavy load requests and the light load requests is about 3/7. As example if the number of user 1000, this result in 300 jobs of light load and 700 jobs of heavy load.



**Figure 4.4 Queuing Network Model for Hospital System**

#### 4.4 Running the Performance Model

After running the performance model, we present the results of Java Modeling Tool. In order to get good results we used Mean Value Analysis. Mean value analysis (MVA) is an efficient algorithm that allows us to analyze product form queuing networks and obtain mean values for queue lengths and response times, as well as throughputs. The efficiency comes because MVA does not compute the joint probability distribution for queue lengths (Marzolla 2010). However, in many cases of performance evaluation situations, the mean values are the performance metrics that required by software engineers.

Fig.4.5 represents the global response time in function of the number of customers in the system keeping constant the mix of two classes. According to the theory for high values of the number of customers, the response time grows linearly.

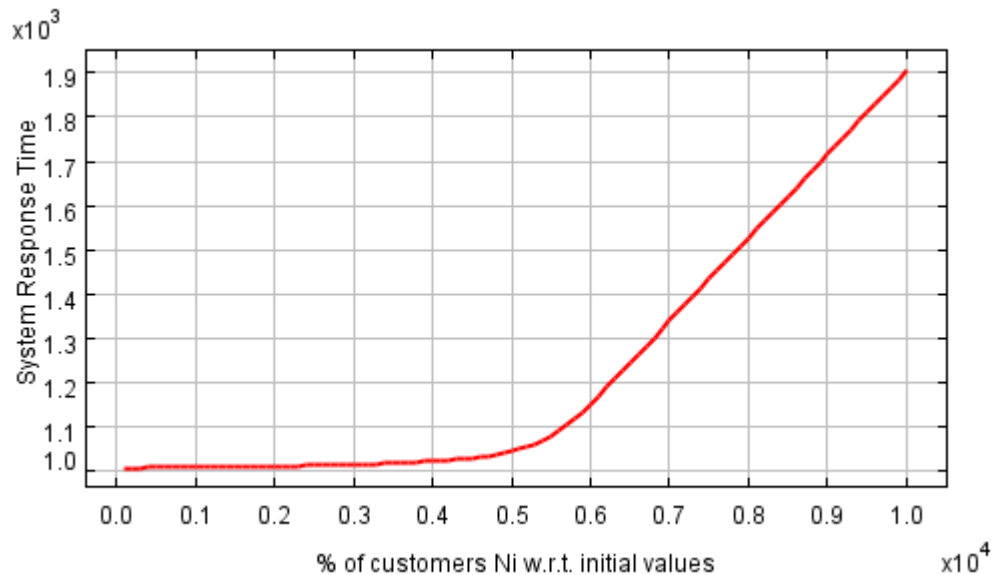
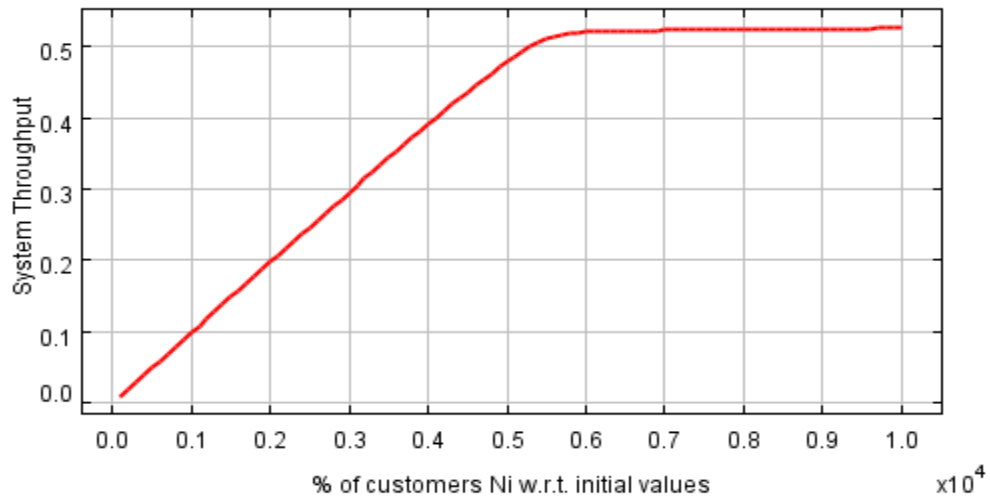
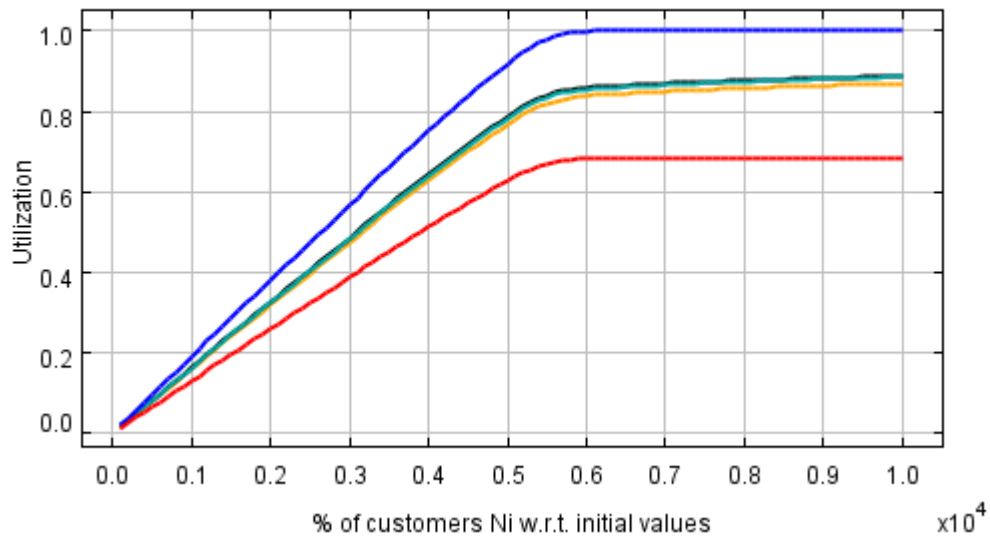


Figure 4.5 the system response time



**Figure 4.6 System Throughputs**

Fig. 4.6 represents the global system throughput, as shown in the figure system throughput will be increased linearly with the number of users rising till reaches the saturation point.



**Figure 4.7 System servers utilization**

Fig.4.6 represents utilization of each station in function of rising number of customers in the system. The upper line refers to the application server, the lower line refers to the web server and the other three lines represent the storage servers.

#### 4.5 Generating the dataset

The generation of the performance dataset accomplished by running the performance model tool JVM many times and record the number of users (workload), system response time, system throughput, application server utilization, web server utilization, and database server utilization. In our approach we generate two datasets:

- (i) Performance Prediction dataset

This dataset consist of numeric data type features such as number of current users of the application, the response time, throughput, web server utilization, application server utilization, and database server utilization(Magalhães et al. 2015). We can predict any feature either response time, or throughput, or any resource utilization.

**Table 4.1 Attributes of Performance dataset**

Attribute	Description	Units
#user	No of users currently logged onto the software system	
SysRes	System response time represents the time taken to send and receive a response	ms
SystThro	System throughput represents number of requests for a given period of time	%
webSerUtli	Web server utilization represents the percentage of the web server in operation	%
AppSerUtli	Application server utilization represents the percentage of server in operation	%
DbServUtli	Database server utilization represents the percentage of server in operation	%
class	Represents the category of the performance risk either it is low, medium, or high	

**Table 4.2 Resource utilization prediction**

#users	SysRes	SystThro	webSerUtli	AppSerUtli	DbServUtli
10	1008.25	0.01	0.01	0.02	0.02
69	1009.16	0.07	0.09	0.13	0.11
128	1010.23	0.13	0.17	0.24	0.21
200	1012.04	0.20	0.26	0.38	0.32
413	1025.03	0.40	0.53	0.77	0.66
489	1041.00	0.47	0.62	0.90	0.77
500	1045.12	0.48	0.63	0.92	0.78
572	1106.41	0.52	0.68	0.99	0.84
649	1243.86	0.52	0.68	1.00	0.86
899	1714.08	0.52	0.68	1.00	0.88
1340	2542.67	0.53	0.68	1.00	0.89
35	1008.54	0.03	0.05	0.07	0.06
2740	5169.85	0.53	0.69	?	0.91
433	1027.88	0.42	0.55	?	0.69
520	1055.05	0.49	0.65	?	0.80
600	1152.79	0.52	0.68	?	0.85
710	1358.83	0.52	0.68	?	0.86
1000	1904.06	0.53	0.68	?	0.88
5000	9408.67	0.53	0.69	?	0.92
-	-	-	-	-	-
-	-	-	-	-	-

In table .4.2 we presented sample of resource utilization dataset. The dataset contains six features: number of users who access the system, the corresponding system response time, throughput, web server utilization, application server utilization, and database server utilization. The dataset shows that we can predict the application server utilization by using machine learning statistical regression.

(ii) Performance Risk Classification dataset

This dataset consists of numeric data type features and category feature. The numeric features are the number of current users of the application, the response time, throughput, web server utilization, application server utilization, and database server utilization. While the category feature is the level of the performance risk (low, or medium, or high) that incurred by a given workload use the system on same time.

**Table 4.3 Performance Risk level classification dataset**

#users	SysRes	SystThro	webSerUtili	AppSerUtili	DbSerUtili	class
10	1008.25	0.01	0.01	0.02	0.02	LR
139	1010.50	0.14	0.18	0.26	0.22	LR
244	1013.47	0.24	0.32	0.46	0.39	LR
516	1052.77	0.49	0.64	0.94	0.80	MR
572	1106.41	0.52	0.68	0.99	0.84	MR
623	1195.13	0.52	0.68	1.00	0.85	HR
960	1828.90	0.52	0.68	1.00	0.88	HR
134	1010.37	0.13	0.17	0.25	0.22	LR
3700	6970.57	0.53	0.69	1.00	0.92	HR
4890	9408.67	0.53	0.69	1.00	0.92	HR



140	1010.51	0.14	0.18	0.27	0.23	LR
167	1011.14	0.17	0.22	0.32	0.27	LR
518	1053.88	0.49	0.64	0.94	0.80	?
5000	9408.67	0.53	0.69	1.00	0.92	?
1899	3591.77	0.53	0.69	1.00	0.91	?
710	1358.83	0.52	0.68	1.00	0.86	?
567	1099.52	0.52	0.68	0.99	0.84	?
33	1008.59	0.03	0.04	0.06	0.05	?
80	1009.33	0.08	0.10	0.15	0.13	?
-	-	-	-	-	-	-
-	-	-	-	-	-	-

In table 4.3 we presented a sample of performance risk classification dataset. The dataset contains 196 instances generated by running the JMT. We change the workload of the system and the performance indices recorded for every change. The generated dataset will work as training dataset and to build the model.

#### 4.6 Resource Utilization Prediction using Machine Learning

By using machine learning, we can accurately predict the server utilization on a given workload. Specifically, we focus here on answering questions like: *“How much resource utilization will be if a certain number of users are using the system concurrently?”*

To answer the previous question we used the technique of regression. Regression is one of the machine learning techniques used to predict numerical values. Table 4.4 shows the dataset for database server utilization(Zhang et al. 2007). The technique of regression learn from the dataset and build prediction model based on the actual dataset (Prof et al. n.d.).

## **4.7 Performance Risk Classification using Machine Learning**

By using machine learning we can predict the level of performance risk whether it is low or medium or high performance risk, if we apply the software system on specific scenario. We focus here on answering questions like: “*What is the class of performance risk if a certain number of users are using the system concurrently?*”

To answer the previous question we used three machine techniques; Naïve Bays, K-Nearest Neighbor (KNN) (Based 2004), and Support Vector Machines (SVMs). Each technique build a model in order to predict new instances based on the previous workload pattern. Furthermore, we compare between the results of the three techniques. Table 5.4 shows the training dataset for performance risk level prediction. It contains the features number of users, system response time, system throughput, web server utilization, application server utilization, database server utilization and the corresponding class of performance risk.

## **4.8 Results and Discussion**

We have presented the results from our experiments after using regression in prediction database server utilization. In addition, we used and compare between three machine learning classification techniques in order to validate of our result to classify the level of performance risks whether it is high, medium, or low performance risk.

### **4.8.1 Database Server Utilization Prediction**

We used a tool called WEKA as machine learning tool in order to predict database server utilization. Table.4.4. represents the regression technique prediction with mean square error 0.01. The value of correlation coefficient indicates that there is a high correlation between the predicted value and the actual value.

**Table 4.4 Database server utilization prediction accuracy**

=== Evaluation on test split ===	
=== Summary ===	
Correlation coefficient	0.7
Mean absolute error	3.5
Root mean square error	4.4
Total Number of Instances	194

#### 4.8.2 Performance Risk Level Classification

After prediction and using of server utilization database, next step is to classify performance risk level. We compare between three machine learning algorithms Naïve Bayes, K-Nearest Neighbor (KNN), and Support Vector Machine (SVM).

(i) **Naïve Bayes**

In table 4.5 we presented the accuracy of our approach by using Naïve Bayes technique. The table shows the mean square error is 0.23, which is considered as small error percentage.

**Table 4.5 Naïve Bayes risk classification accuracy**

=== Evaluation on test split ===		
=== Summary ===		
Correctly Classified Instances	53	91.3 %
Incorrectly Classified Instances	5	8.6 %
Mean absolute error	0.06	
Root mean squared error	0.23	
Total Number of Instances	58	

Table 4.6 stated the confusion matrix of Naïve Bayes where four instances are classified as medium performance risk while they are actually high performance risk, and one instance is classified as medium performance risk while it is actually high performance risk.

**Table 4.6 Naïve Bayes risk classification confusion matrix**

=== Confusion Matrix ===			
A	b	C	
22	4	0	a = LR
0	12	0	b = MR
0	1	19	c = HR

(ii) **K-Nearest Neighbor (KNN)**

In table 4.7 we presented the accuracy percentage of the second technique we used. KNN is one of famous machine learning techniques. KNN presents mean square error 0.21 which is considered as better than Naïve Bayes.

**Table 4.7 KNN performance risk classification accuracy**

=== Evaluation on test split ===		
=== Summary ===		
Correctly Classified Instances	54	93.1 %
Incorrectly Classified Instances	4	6.8 %
Mean absolute error	0.05	
Root mean squared error	0.21	
Total Number of Instances	58	

In table 4.8 the confusion matrix of KNN stated that three instances are classified as low performance risk while they are actually medium performance risk, and one instance is classified as medium performance risk while it is actually high performance risk.

**Table 4.8 KNN performance risk classification confusion matrix**

=== Confusion Matrix ===			
A	B	C	
23	3	0	a = LR
0	12	0	b = MR
0	1	19	c = HR

(iii) **Support Vector Machine (SVM)**

In table 4.9 we present the accuracy of SVM for prediction of performance risk classification. SVM works and produce mean square error 0.4 which considered as highest percentage of error.

**Table 4.9 SVM performance risk classification accuracy**

=== Evaluation on test split ===		
=== Summary ===		
Correctly classified Instances	40	68.9 %
Incorrectly Classified Instances	18	31.03 %
Mean absolute error	0.31	
Root mean squared error	0.40	
Total Number of Instances	58	

In table 4.10 the confusion matrix of SVM showed that two instances are classified as medium performance risk while they are actually low performance risk, also four instances are classified as high performance risk while they are actually low performance risk, and twelve instances are classified as high performance risk while they are actually medium performance risk.

**Table 4.10 SVM performance risk classification confusion matrix**

=== Confusion Matrix ===			
a	B	C	
20	2	4	a = LR
0	0	12	b = MR
0	0	20	c = HR

The results of the experiments are summarized in table 4.11. The performances of the three models were evaluated based on three criteria, the prediction accuracy, learning time and error rate.

**Table 4.11 comparisons between Classification Algorithms**

Evaluation criteria	Classifiers		
	Naïve Bayes	KNN	SVM
Timing to build model (in sec)	0.01	0.01	0.28
Correctly classified instances	53	54	40
Incorrectly classified instances	5	4	18
Prediction accuracy	91%	93.1%	68.9%

The results indicate that the K-Nearest Neighbor classifier outperforms most other Naïve Bayes and Support Vector Machine methods in prediction. Although the timing to build the model between Naïve Bayes and KNN are similar, the prediction accuracy differs significantly.

Fig. 4.8 shows the results of accuracy by performance risk class. As marked on the figure the precision of the class LR zone is very high, which indicates that the techniques works well when the relation is linear and consequently degrades when the relation change to exponential at classification of MR and HR performance risk zones.

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.795	0	1	0.795	0.886	0.974	LR
	0.049	0.105	0.111	0.049	0.068	0.472	MR
	1	0.328	0.658	1	0.794	0.836	HR
Weighted Avg.	0.716	0.149	0.68	0.716	0.677	0.815	

**Figure4.8 Detailed Accuracy by Class**

## 4.9 Summary

In this chapter we applied our approach on a case study of a hospital system. We studied the system and then the UML diagrams that describe the system were built. Furthered, we mapped UML diagrams to performance model. The performance model was generated by using JVM as standard tool based on Java. JVM run by changing workloads and record the corresponding response time, throughput, and server’s utilization into a dataset. We build the dataset in order to enable machine learning to build the classification model. To validate our approach we compare between three techniques Naïve Bayes, SVM, and KNN. KNN technique showed good and more accurate results among the mentioned techniques.

## **CHAPTER V**

### **Predicting Performance of E-commerce System**

#### **5.1 Introduction**

The performance of e-commerce sites plays a crucial role in attracting and retaining customers. Frustrated customers leave these sites and do not return, causing revenue to be lost. E-commerce system is a web-based application that manages data of the business: Customers browse catalogs and make selection of items that need to be purchased. Moreover, suppliers can upload their catalogs; change the prices and availability of products (Cortellessa et al. 2005). The scenarios we analyzed in this chapter here are: Browse Catalog and Make Purchase. The first scenario can be critical for performance attribute because it is required by a large number of customers who registered or who are not registered, while the latter can be also critical for performance attribute because it requires several database accesses that can be degrade the application performance.

#### **5.2 The Architecture and Model**

E-commerce system has been studied; the system was analyzed by Trubian, 2011. Trubian supposed that the max number of customer reach up to 5000 users. The interaction of the customers with system is provided by a web application with a browser on client side and database on server side. The requests are submitted to a web server node, then the request passed to the library node, and finally to the control node in case of a static request. Moreover, the web server node interacts with an application server that performs some quires on a backend database server and processed data are passed back to the web server node. Finally clients receive the dynamically generated result via the web browser.



They grouped the requests arriving at the system in two groups: the requests for a Browse catalog and the requests for a make purchase. The two groups are modeled by classes with different service time for each station. Considering the point of view of the database they refer to the search-class and the modification-class as the Browse catalog class and the make purchase class. In table 5.1 the service demand are reported for each station in the system and for each class. The different values of the two classes are due to the different behavior of the two types of users

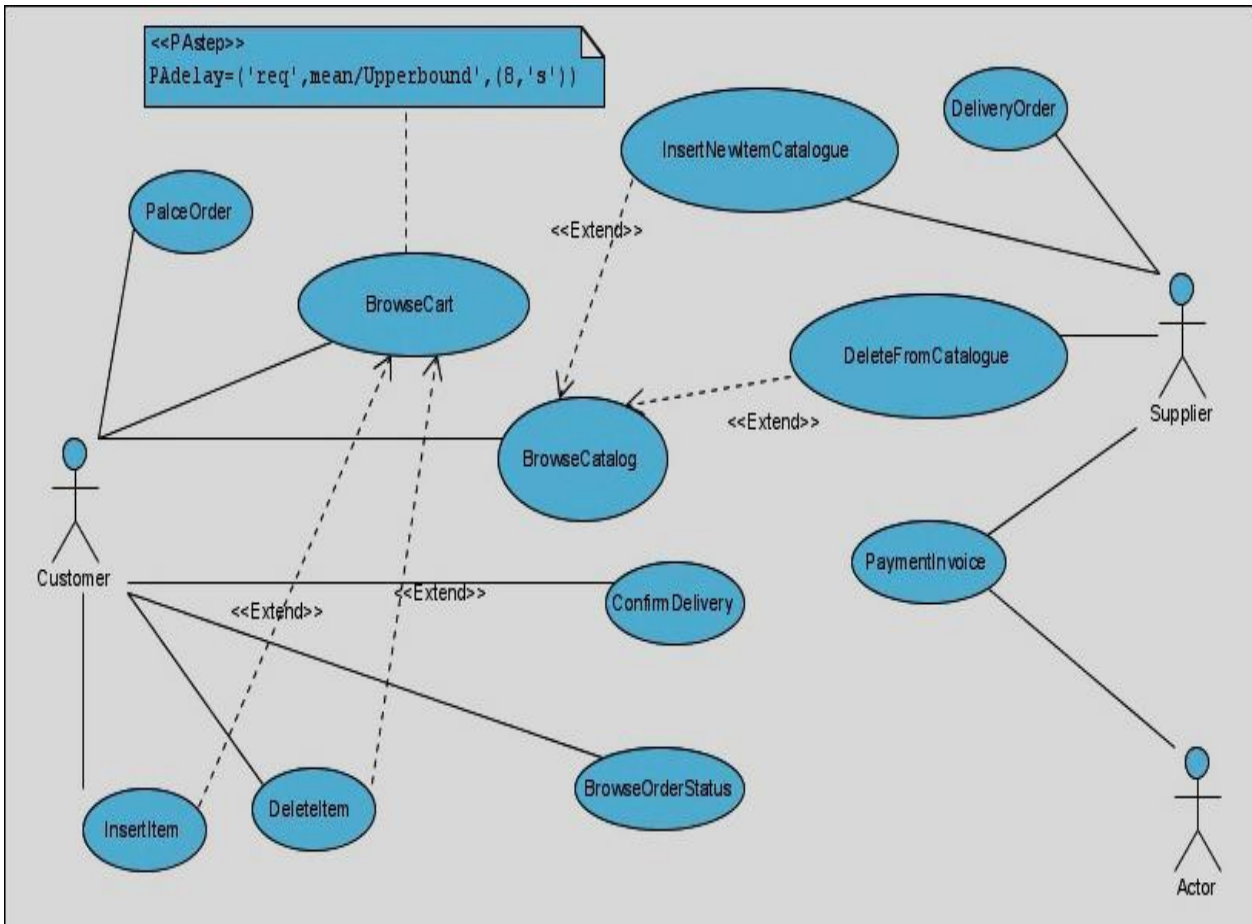
**Table 5.1 Service demands in milliseconds for each station in the system.**

	<b>Browse catalog (Class A)</b>	<b>Make purchase (Class B)</b>
Lan	44 msec	44 msec
Wan	208 msec	208 msec
webServerNode	2 msec	4 msec
librayNode	7 msec	16 msec
controlNode	3 msec	3 msec
db_cpu	15 msec	30 msec
db_Disk	30 msec	60 msec

The ratio between the number of Browse catalog requests and the Make purchase is about of 90% for Browse catalog and 10% for Make purchase. That means if there are 5000 customers using the system the ratio will be result in 500 requests of Make purchase and 4500 requests of Browse catalog.

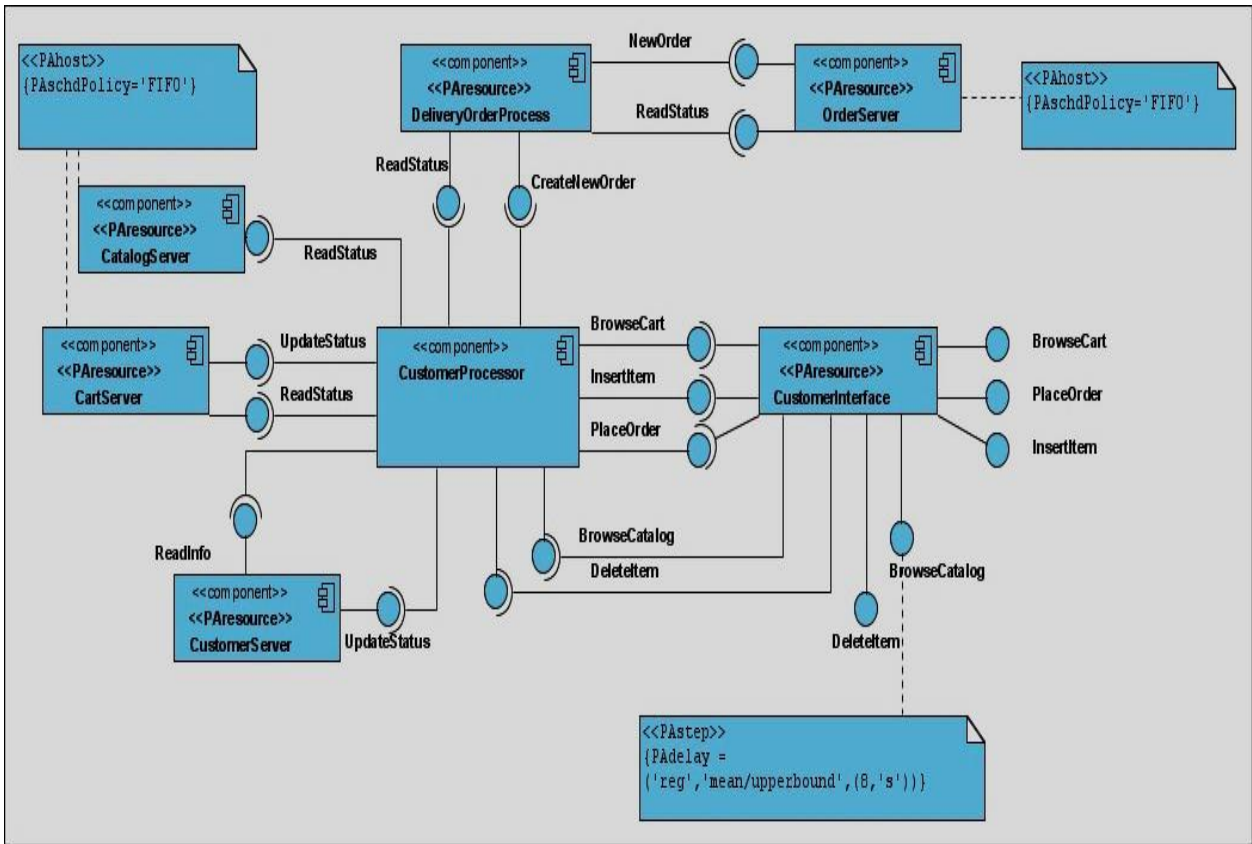
The use case diagram in Fig. 5.1 represents the artifact modeling the e-commerce system at the level of requirements specification (Cortellessa et al. 2005). The diagram annotated the link connecting the customer actor to the browse catalog functionality to express a response time requirement, that is: a customer should not wait more than eight seconds to access the catalog. From the analysis point of view we can explain this limit

either as an average or as an upper bound (Rajagopal & Thilakavalli 2017). The capability of annotating UML diagrams with additional information such as performance parameters and indices is provided from the UML profiling technique that has been described in Chapter 2.



**Figure 5.1 ECS Use Case Diagram**

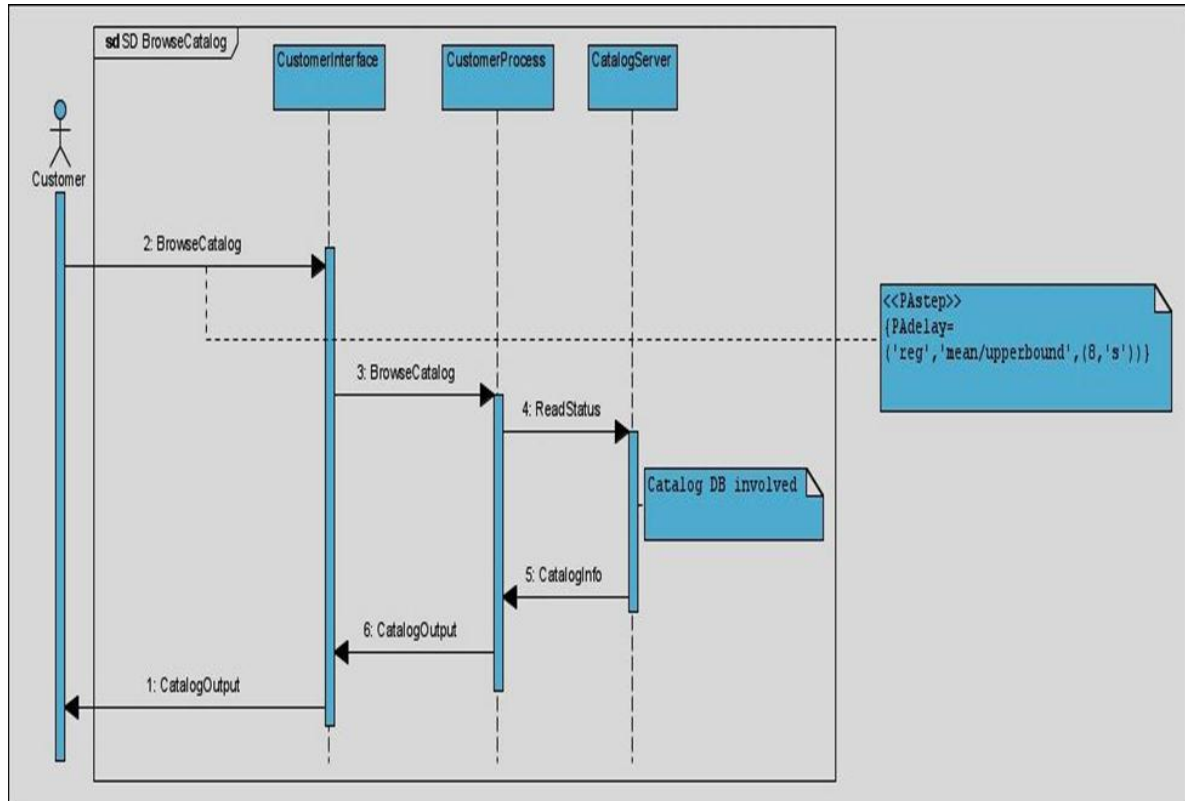
Through the proceeding in the software development process, Fig. 5.2 shows an annotated UML component diagram of the example, which represents a static view of the e-commerce system. Component diagrams are used to model physical features of a system and the relation between each object.



**Figure 5.2 ECS Components Diagram**

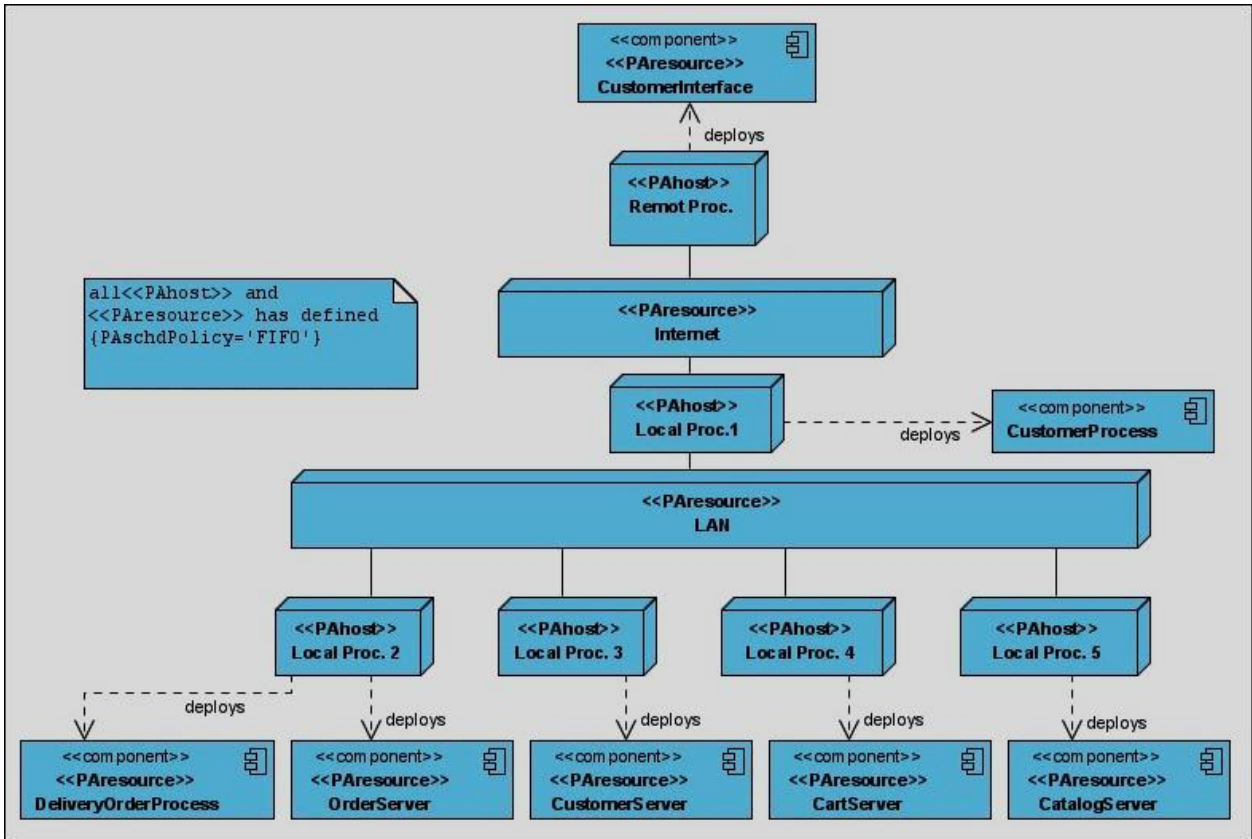
Component diagram describes the components used to make functionalities of the system. Furthermore, a single component diagram cannot represent the entire system but a collection of diagrams are used to represent the whole.

Fig. 5.3 shows an annotated UML sequence diagram which represents the dynamic view of the system architecture. A sequence diagrams indicates interactions among a set of objects temporal order, which is good for understanding timing and interplay issues.



**Figure 5.3 ECS Sequence Diagram**

Also, the diagram depicts the objects via their lifelines and shows messages they exchange in time sequence.



**Figure 5.4 ECS Deployment Diagram**

Moreover, Fig.5.4 shows the UML deployment diagram of the system that reflects the configuration of run-time processing nodes and the components hosted on them. Deployment diagrams address the static deployment view of architecture. Furthermore, they related to component diagrams with nodes hosting one or more components.

### 5.3 Transformation of UML Diagrams into Performance Model

Fig. 5.5 shows the Queuing Network model for the e-commerce case study. It includes:

- (i) A set of queuing centers (web server node, Library server node, control node, database servers nodes) representing the hardware resources of the system, a delay center represents the number of users.

- (ii) Two classes of jobs (browse catalog and make purchase), the ratio between the number of browse catalog requests and the purchase requests is about 90% for browse catalog to 10% for make purchase. As example if the number of users 1000, this result in 100 jobs of make purchase and 900 jobs of browse catalog.

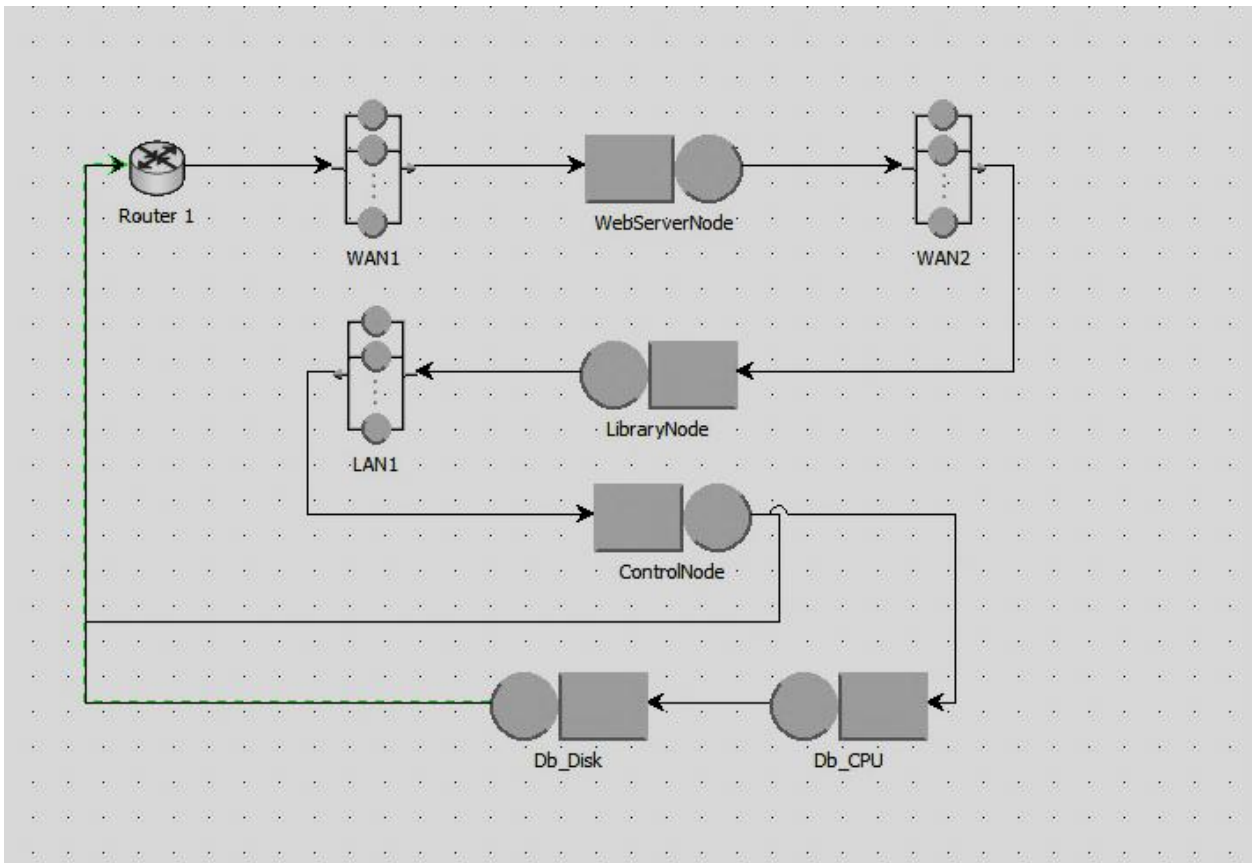
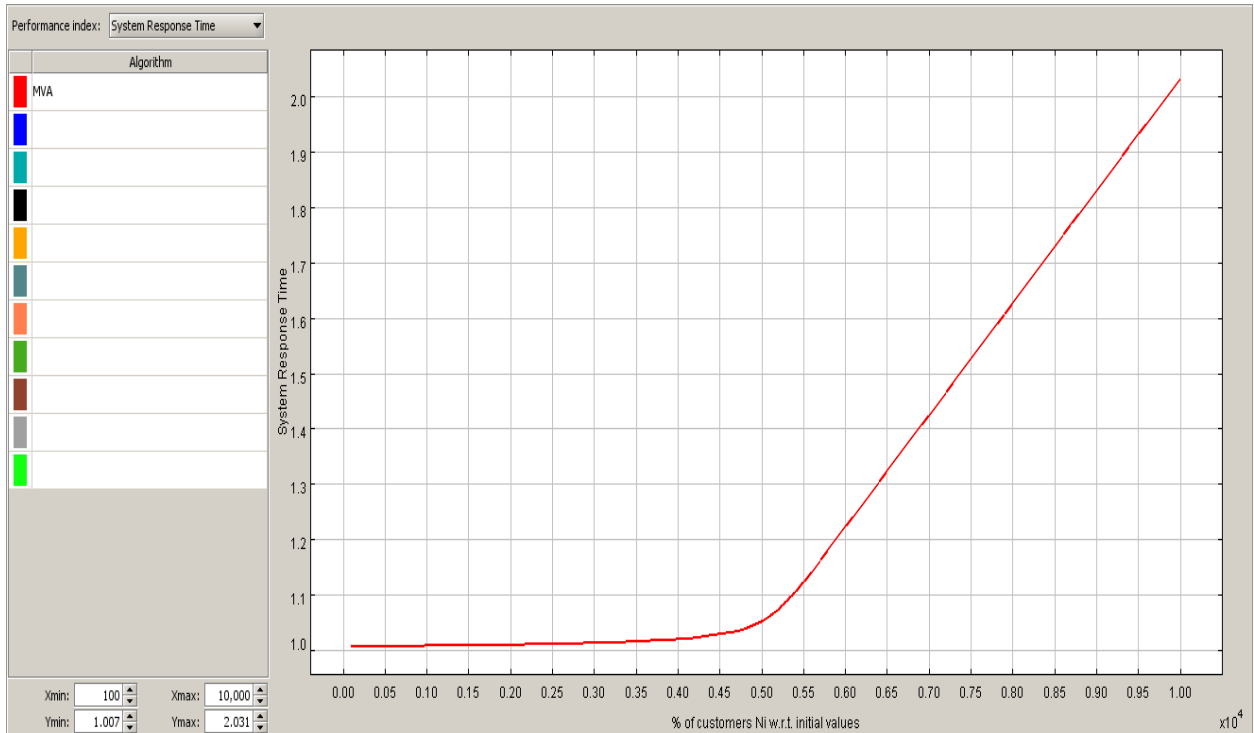


Figure 5.5 Queuing Network Model for E-commerce System

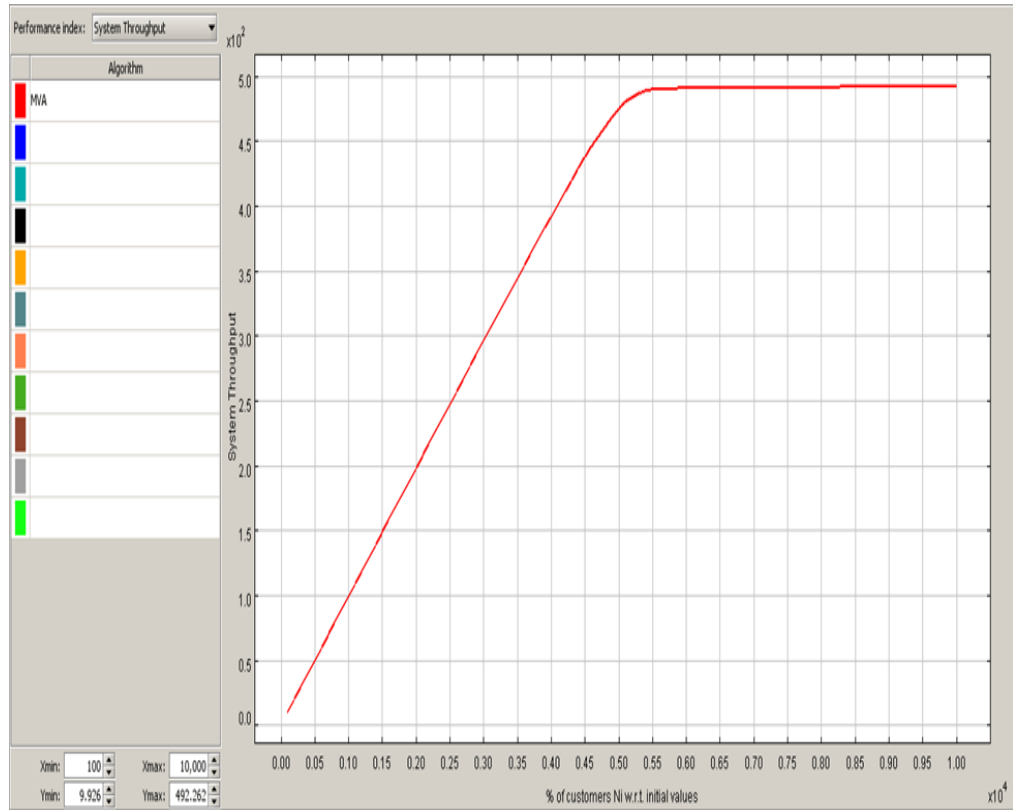
#### 5.4 Running the Performance Model:

After running the performance model, we present the results of Java Modeling Tool (JMT). In order to get good results we use Mean Value Analysis algorithm (MVA). The reasons for using MVA are mention in Chapter 4. Fig.5.6 represents the global response

time in function of the number of customers in the system keeping constant the mix of two classes. According to the theory for high values of the number of customers, the response time grows linearly.



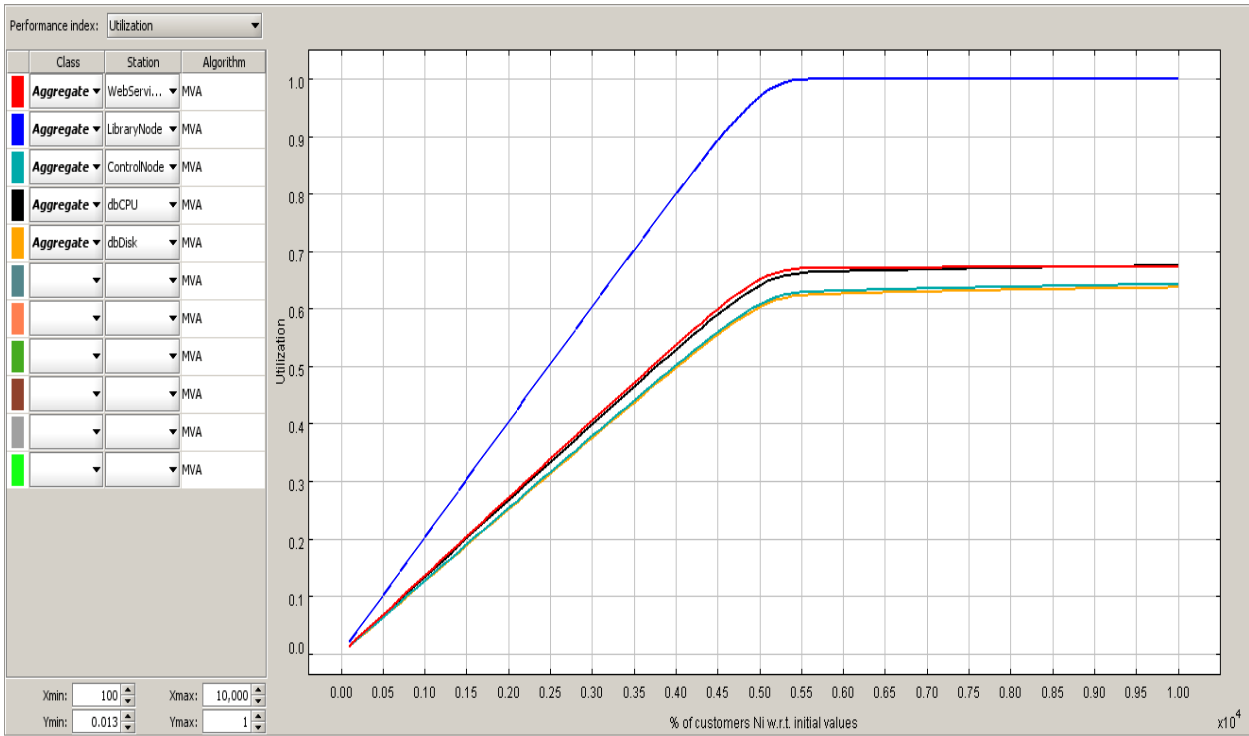
**Figure 5.6 ECS response time**



**Figure 5.7 ECS throughput**

Fig. 5.7 represents the global system throughput, as shown in the figure system throughput will be increased linearly with the number of users rising till the system reaches the saturation point.





**Figure 5.8 ECS Servers utilization**

Fig. 5.8 represents utilization of each station in function of increasing number of customers in the system. The upper line refers to the library node server; the lower line refers to the web server, control node, and database server.

## 5.5 Generating the dataset

The generation of the performance dataset accomplished by running the performance model tool JVM many times and record the number of users (workload), system response time, system throughput, library node server utilization, web server utilization, control node server, and database server utilization. In our approach we generate two datasets for performance prediction and for performance risk classification:-

### 5.5.1 Performance Prediction dataset

Table 5.2 presents the response time of the ECS system. This dataset consist of numeric data type features such as number of current users of the application, the response

time, throughput, web server utilization, application server utilization, and database server utilization. We can predict any feature either response time, or throughput, or any of the server utilization value.

**Table 5.2 ECS Response time prediction using regression model dataset**

<b>users</b>	<b>Res</b>	<b>thro</b>	<b>webutil</b>	<b>librutil</b>	<b>control</b>	<b>dbCPU</b>	<b>dbDisk</b>
10	1	9.92	0.01	0.02	0.01	0.01	0.01
15	1	14.89	0.02	0.03	0.01	0.01	0.01
19	1	18.85	0.02	0.03	0.02	0.02	0.02
22	1	21.83	0.02	0.04	0.02	0.02	0.02
26	1	25.8	0.03	0.05	0.03	0.03	0.03
28	1	27.78	0.03	0.05	0.03	0.03	0.03
30	1	29.77	0.04	0.06	0.03	0.04	0.03
34	1	33.74	0.04	0.06	0.04	0.04	0.04
39	1	38.69	0.05	0.07	0.04	0.05	0.04
41	1	40.68	0.05	0.08	0.05	0.05	0.05
46	1	45.64	0.06	0.09	0.05	0.06	0.05
49	1	48.61	0.06	0.09	0.06	0.06	0.06
53	1	52.58	0.07	0.1	0.06	0.07	0.06
56	1	55.55	0.07	0.11	0.07	0.07	0.07
59	1	58.53	0.08	0.11	0.07	0.07	0.07
63	1	62.49	0.08	0.12	0.07	0.08	0.07
67	1	66.45	0.09	0.13	0.08	0.09	0.08
69	1	68.44	0.09	0.13	0.08	0.09	0.08

....	....	....	....	....	....	....	....
....	....	....	....	....	....	....	....
1100	2.23	492.49	0.67	1	0.64	0.67	0.63
1117	2.26	492.57	0.67	1	0.64	0.67	0.63
1128	2.28	492.58	0.67	0.80	0.64	0.67	0.63
1137	?	492.61	0.67	0.77	0.64	0.67	0.63
1154	?	492.55	0.67	0.33	0.64	0.67	0.63
1162	?	492.59	0.67	0.67	0.64	0.67	0.63
1179	?	492.67	0.67	0.49	0.64	0.67	0.63
1189	?	492.68	0.67	0.99	0.64	0.67	0.64
1198	?	492.72	0.67	0.76	0.64	0.67	0.64
1200	?	492.69	0.67	0.65	0.64	0.67	0.64
1220	?	492.73	0.67	0.90	0.64	0.67	0.64
1250	?	492.78	0.67	1	0.64	0.67	0.64
1450	?	493.08	0.67	0.67	0.64	0.68	0.64
1550	?	493.21	0.67	0.98	0.64	0.68	0.64
2900	?	494.1	0.67	0.55	0.65	0.68	0.65
3000	?	494.14	0.67	0.50	0.65	0.68	0.65
3700	?	494.33	0.67	1	0.65	0.68	0.65
4500	?	494.49	0.67	0.76	0.65	0.68	0.65
4530	?	494.49	0.67	0.77	0.65	0.68	0.65
10000	?	494.89	0.67	0.77	0.66	0.69	0.65
10500	?	494.91	0.67	0.78	0.66	0.69	0.65

100000	?	495.2	0.67	0.79	0.66	0.69	0.65
--------	---	-------	------	------	------	------	------

### 5.5.2 Performance Risk Classification dataset

This dataset consists of numeric data type features and category feature. The numeric features are the number of current users of the application, the response time, throughput, library server utilization, control server utilization, database server CPU utilization, and database Disk server utilization. In addition, the category feature is the of the performance risk level which is low, or medium, or high that incurred on a given workload.

**Table 5.3 Sample of Classification dataset for performance risk level**

Users	res	thro	webutil	librutil	Control	dbCPU	dbDisk	Class
10	1	9.92	0.01	0.02	0.01	0.01	0.01	LR
15	1	14.89	0.02	0.03	0.01	0.01	0.01	LR
19	1	18.85	0.02	0.03	0.02	0.02	0.02	LR
22	1	21.83	0.02	0.04	0.02	0.02	0.02	LR
26	1	25.8	0.03	0.05	0.03	0.03	0.03	LR
28	1	27.78	0.03	0.05	0.03	0.03	0.03	LR
30	1	29.77	0.04	0.06	0.03	0.04	0.03	LR
524	1.07	485.34	0.66	0.99	0.62	0.65	0.61	MR
526	1.08	486.06	0.66	0.99	0.62	0.65	0.61	MR
529	1.08	486.79	0.66	0.99	0.62	0.65	0.61	MR
531	1.08	487.21	0.66	0.99	0.62	0.65	0.61	MR
533	1.09	487.59	0.66	0.99	0.62	0.65	0.61	MR

535	1.09	487.92	0.66	0.99	0.62	0.65	0.61	MR
538	1.1	488.57	0.66	0.99	0.62	0.66	0.62	MR
540	1.1	488.81	0.66	0.99	0.62	0.66	0.62	MR
543	1.11	489.1	0.66	0.99	0.62	0.66	0.62	MR
545	1.11	489.26	0.6	0.99	0.62	0.66	0.62	MR
547	1.11	489.63	0.67	0.99	0.62	0.66	0.62	MR
549	1.12	489.74	0.67	0.99	0.62	0.66	0.62	MR
552	1.12	489.88	0.67	0.99	0.62	0.66	0.62	MR
555	1.13	489.94	0.67	0.99	0.62	0.66	0.62	MR
945	1.92	492.03	0.67	1	0.64	0.67	0.63	HR
951	1.93	492.11	0.67	1	0.64	0.67	0.63	HR
957	1.9	492.19	0.67	1	0.64	0.67	0.63	HR
964	1.95	492.1	0.67	1	0.64	0.67	0.63	HR
969	1.96	492.19	0.67	0.80	0.64	0.67	0.63	HR
974	1.97	492.12	0.67	0.77	0.64	0.67	0.63	HR
983	1.99	492.16	0.67	0.33	0.64	0.67	0.63	HR
989	2	492.25	0.67	0.67	0.64	0.67	0.63	HR
990	2.01	492.23	0.67	0.49	0.64	0.67	0.63	HR
1000	2.03	492.26	0.67	0.99	0.64	0.67	0.63	HR
1006	2.04	492.34	0.67	0.76	0.64	0.67	0.63	?
1015	2.06	492.21	0.67	0.65	0.64	0.67	0.63	?
1027	2.08	492.37	0.67	0.90	0.64	0.67	0.63	?
1033	2.09	492.29	0.67	1	0.64	0.67	0.63	?

1046	2.12	492.43	0.67	0.67	0.64	0.67	0.63	?
1052	2.13	492.35	0.67	0.98	0.67	0.67	0.63	?
1059	2.15	492.42	0.67	0.55	0.67	0.67	0.63	?
1065	2.16	492.34	0.67	0.50	0.64	0.67	0.63	?
1072	2.17	492.4	0.67	1	0.64	0.67	0.63	?
380	1.01	373.06	0.51	0.76	0.47	0.5	0.47	?
385	1.01	377.8	0.51	0.77	0.48	0.5	0.47	?
389	1.01	381.58	0.52	0.77	0.48	0.51	0.48	?
392	1.01	384.41	0.52	0.78	0.49	0.51	0.48	?
400	1.02	391.92	0.53	0.79	0.5	0.52	0.49	?
403	1.02	394.73	0.54	0.8	0.5	0.53	0.5	?
412	1.02	403.1	0.55	0.82	0.51	0.54	0.51	?
419	1.02	409.55	0.56	0.83	0.52	0.55	0.51	?
425	1.02	415.03	0.56	0.84	0.52	0.55	0.52	?
428	1.02	417.76	0.57	0.85	0.53	0.56	0.53	?
430	1.02	419.56	0.57	0.85	0.53	0.56	0.53	?
437	1.02	425.84	0.58	0.86	0.54	0.57	0.54	?
442	1.02	430.27	0.58	0.87	0.54	0.58	0.54	?

## 5.6 Performance Prediction using Multivariate Regression

By using machine learning, we can accurately predict the system response time on a given workload. Specifically, we focus here on answering questions like: *“How much the system response time will be if a certain number of users using the system concurrently?”*

To answer the previous question we used the technique of regression. Regression is one of the machine learning prediction technique used to predict numerical values.

### **5.7 Performance Risk Assessment using Machine Learning Techniques**

By using machine learning we can predict the level of performance risk whether it is low or medium or high performance risk, if we apply the software system on specific configuration. We focus here on answering questions like: *“What is the class of performance risk if a certain number of users are using the system concurrently?”*

To answer the previous question we compare between three machine learning algorithms; Naïve Bays, K-Nearest Neighbor (KNN), and Support Vector Machines (SVMs). We apply the classifiers on the training dataset contains 250 instances, and then we compare between the results. Table 5.3 shows a sample of training dataset for performance risk level prediction. It contains number of users, system response time, system throughput, library server utilization, control server utilization, database CPU server utilization, database Disk server utilization and the class of performance risk.

### **5.8 Results and Discussion**

We have presented the results from our experiments after using multivariate regression in prediction system response time. In addition, we used and compare between three classification techniques in order to classify the level of performance risks whether it is high, medium, or low performance risk.

#### **5.8.1 Prediction of System Response Time**

We used WEKA as machine learning tool in order to predict system response time. Figure 5.4 represents the multivariate regression technique prediction with mean square error 2.12. The high value of the correlation coefficient indicates the small differences between the actual value and the predicted value of system response time.

**Table 5.4 ECS Response Time Prediction Accuracy**

==== Evaluation on test split ====	
==== Summary ====	
Correlation coefficient	0.8
Mean absolute error	2.9
Root mean squared error	3.7
Total Number of Instances	250

## **5.8.2 Performance Risk Level Classification**

After prediction of system response time, the next step is to classify performance risk level. We compare between three machine learning algorithms Naïve Bayes, K-Nearest Neighbor (KNN), and Support Vector Machine (SVM) same as the previous example of hospital system presented in chapter four.

### **5.8.2.1 Naïve Bayes**

In Fig.5.5 we presented the accuracy of our approach by using Naïve Bayes algorithm. From the figure the mean square error is 0.22.



**Table 5.5 Naïve Bayes performance risk classification accuracy**

=== Evaluation on test split ===		
=== Summary ===		
Correctly Classified Instances	78	91.7 %
Incorrectly Classified Instances	7	8.2 %
Mean absolute error	0.05	
Root mean squared error	0.22	
Total Number of Instances	58	

Fig. 5.6 stated the confusion matrix of Naïve Bayes where two instances are classified as medium performance risk while actually they are low performance risk. Moreover, four instances is classified as medium performance risk while correctly it is high performance risk, and one instance is classified as low performance risk while actually it is medium performance risk.

**Table 5.6 Naïve Bayes performance risk classification confusion matrix**

=== Confusion Matrix ===			
a	b	C	
50	2	0	a = LR
1	14	0	b = MR
0	4	14	c = HR

### 5.8.2.2 K-Nearest Neighbor (KNN)

In table 5.7 we presented the accuracy percentage of KNN algorithm with mean square error 0.12. KNN gives a good percentage for correctly classified instances.

**Table 5.7 KNN performance risk classification accuracy**

<b>==== Evaluation on test split ====</b>		
<b>==== Summary ====</b>		
Correctly Classified Instances	83	97.64 %
Incorrectly Classified Instances	2	2.3 %
Mean absolute error	0.02	
Root mean squared error	0.12	
Total Number of Instances	85	

In table 5.8 the confusion matrix of KNN stated that two instances incorrectly classified. One instance is classified as medium performance risk while actually it is low performance risk, while one instance classified as medium risk while it is actually high performance risk.

**Table 5.8 KNN performance risk classification confusion matrix**

<b>==== Confusion Matrix ====</b>			
a	b	C	
51	1	0	a = LR
0	14	1	b = MR
0	0	18	c = HR

### 5.8.2.3 Support Vector Machine (SVM)

In table 5.9 we present the accuracy of SVM for prediction of performance risk classification with mean square error is 0.4. SVM shows low percentage of correctly classified instance.

**Table 5.9 SVM performance risk classification accuracy**

=== Evaluation on test split ===		
=== Summary ===		
Correctly Classified Instances	60	70.5 %
Incorrectly Classified Instances	25	29.4 %
Mean absolute error	0.19	
Root mean squared error	0.44	
Total Number of Instances	85	

In table 5.10 the confusion matrix of SVM showed that eight instances are classified as low performance risk while they are actually medium performance risk, and seventeen instances are classified as low performance risk while they are actually high performance risk.

**Table 5.10 SVM performance risk classification confusion matrix**

=== Confusion Matrix ===			
a	b	C	
52	0	0	a = LR
8	7	0	b = MR
17	0	1	c = HR

The results of the experiments are summarized in table 5.11. The performances of the three models were evaluated and compared based on three criteria, the prediction accuracy, learning time and error rate.

**Table 5.11 Comparisons between Classification Techniques**

Evaluation criteria	Classifiers		
	Naïve Bayes	KNN	SVM
Timing to build model (in sec)	0.01	0.01	0.55
Correctly classified instances	78	83	60
Incorrectly classified instances	7	2	25
Prediction accuracy	91.7 %	97.6 %	70.5 %

The results indicate that the K-Nearest Neighbor classifier outperforms in prediction than Naïve Bayes and Support Vector Machine methods. Although the timing to build the model between Naïve Bayes and KNN are similar, the prediction accuracy differs significantly.

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.859	0	1	0.859	0.924	0.962	LR
	0	0	0	0	0	0.665	MR
	1	0.31	0.612	1	0.759	0.845	HR
Weighted Avg.	0.792	0.102	0.741	0.792	0.748	0.885	

**Figure 5.9 the detailed accuracy by class**

As marked on the figure the accuracy of prediction looks to be very high at the zone of LR which is linear, while the accuracy degrades at MR and HR as the relation changed to exponential relation.

## **5.9 Summary**

In this chapter we applied our approach on a case study of e-commerce system. We studied the system and UML diagrams were designed. After that, we mapped UML diagrams to performance model. The performance model was designed by using JVM as standard tool based on Java code. Moreover, the JVM runs by changing workload and record corresponding response time, throughput, and server's utilization. We use the dataset in order to predict performance of system response time. In addition, we apply three machine learning techniques to classify the performance risk of a specific workload. The comparison states that KNN gives a good result among the other techniques.

## **CHAPTER VI**

### **Prediction Performance of Online Time Table (OTT)**

#### **6.1 Introduction**

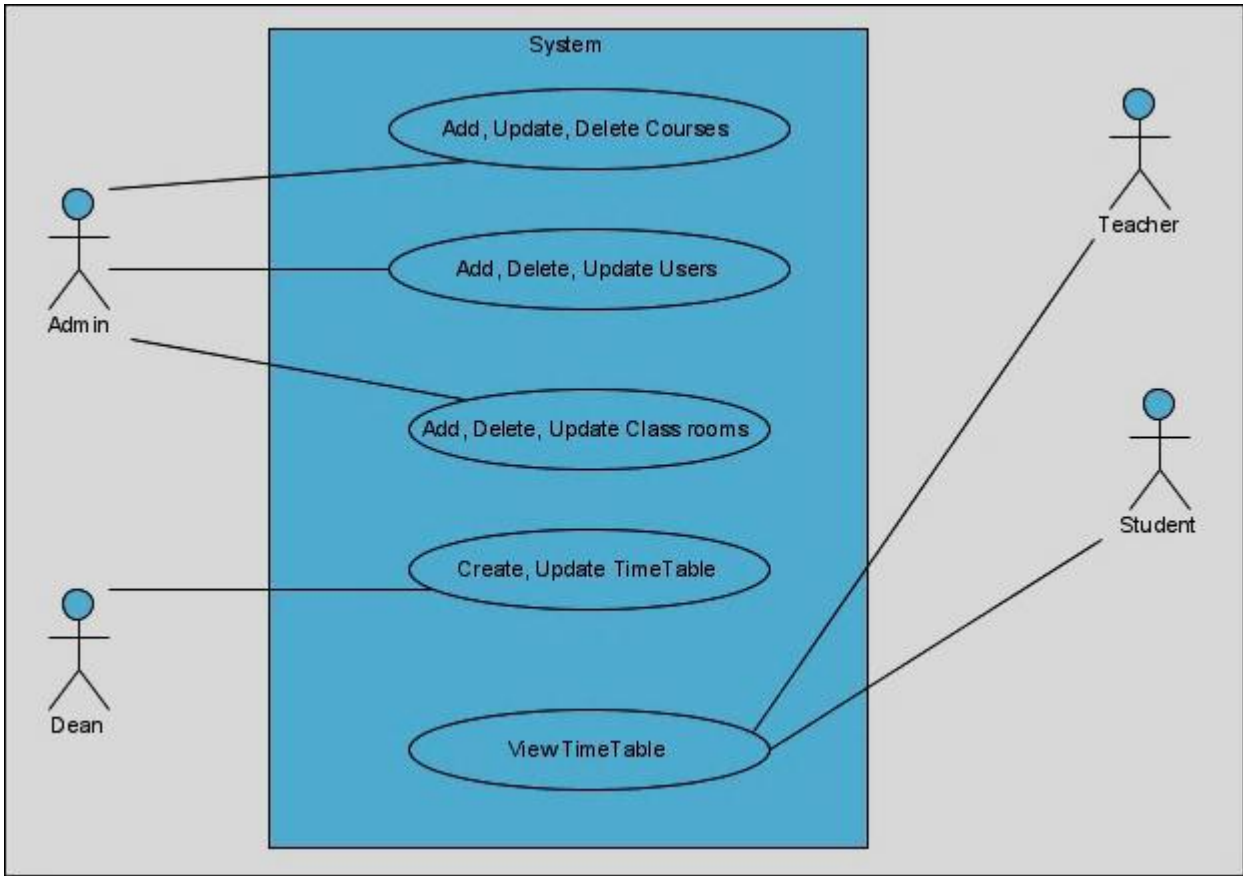
In this chapter the methodology with already running system has been applied. OTT is an online time table system that serves university teachers and students. The performance of the system is crucial as both teachers and students are login to the system in order to know their classes with associated timing.

#### **6.2 The System Architecture**

The system contains three modules: Administrator module, lecturer module, and student module. The role of the administrator is to handle entire administration task. Administrator has to handle additional, editing and deleting classes, subjects, and timing.

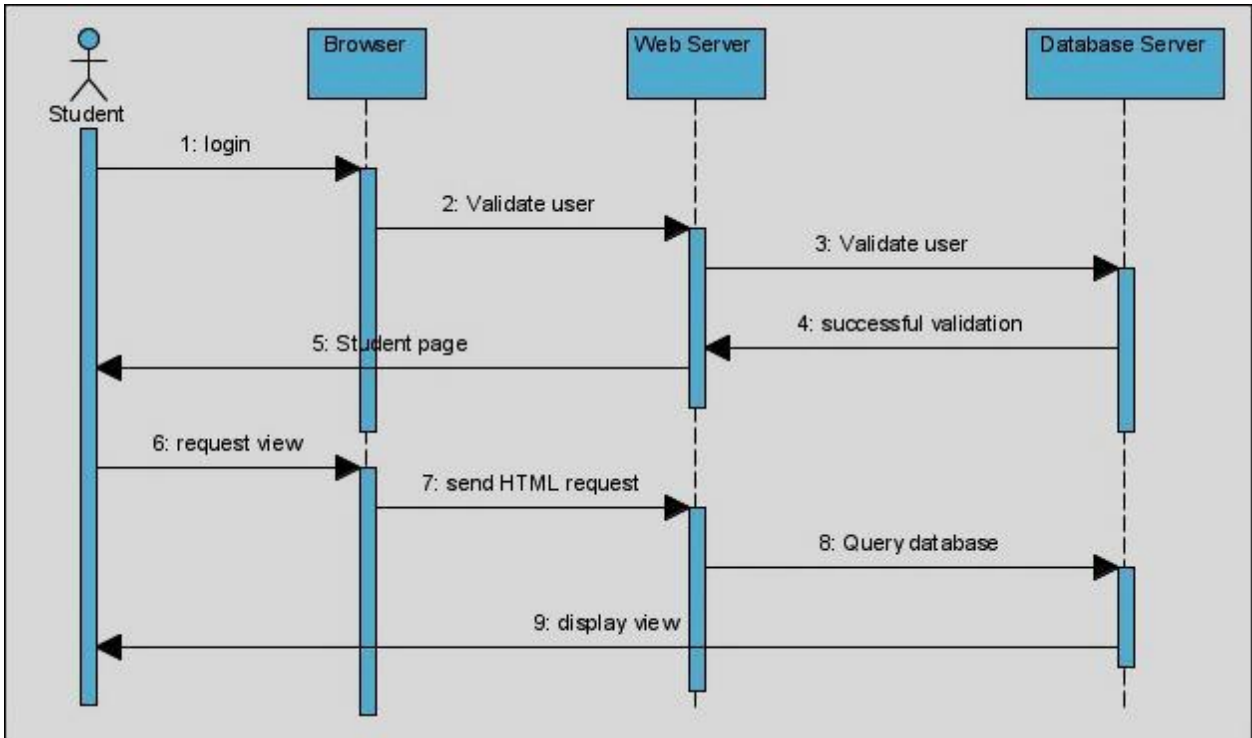
For the lecture module it contains the function to view timetable for the specific lecturer and the timetable for all the semester. Students can view timetable and print. OTT also contains a database, which stores the lectures and class rooms. Only the administrator can view, add and delete the data in the timetable.

In Fig. 6.1 the admin of the system can login, add, update, and delete classes according to specific time. However, teachers and students login the system to view their classes timing and the assigned room.



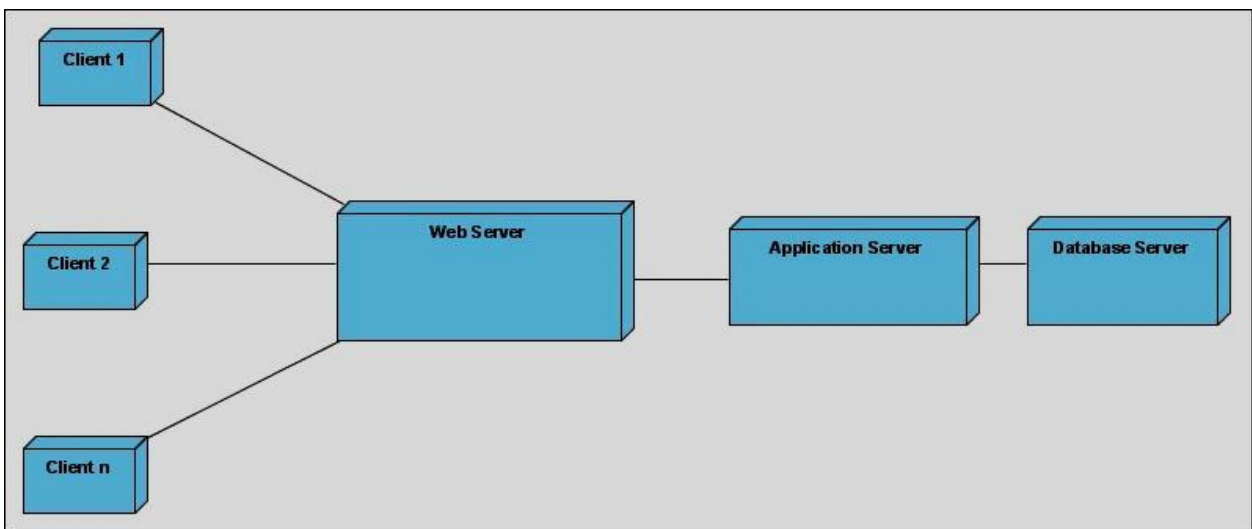
**Figure 6.1 OTT Use Case Diagram**

Fig. 6.2 shows the sequence diagram for the students trying to inquiry for semester timetable. At the beginning the system validates the student user name and password. After that student input his batch year, then the application server sends the request to the database and comes with answer to be displayed as HTML on the browser of the student.



**Figure 6.2 OTT Sequence Diagram for Student**

Fig. 6.3 shows the distribution of the software on the hardware. The OTT system consists of three servers: web server, application server, and database server. Users of the system use their smart phones, laptops, and PCs to get system services.



**Figure 6.3 OTT System Deployment Diagram**



### 6.3 Collecting Performance Dataset using Apache JMeter

The Apache JMeter application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions.

Apache JMeter may be used to test performance both on static and dynamic resources, Web dynamic applications (Singh & Kumar 2011). It can be used to simulate a heavy load on a server, group of servers, network or object to test its strength or to analyze overall performance under different load types.

Apache JMeter features include:

- Ability to load and performance test many different applications/server/protocol types:
  - Web - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, ...)
  - SOAP / REST Webservices
  - FTP
  - Database via JDBC
  - LDAP
  - Message-oriented middleware (MOM) via JMS
  - Mail - SMTP(S), POP3(S) and IMAP(S)
  - Native commands or shell scripts
  - TCP
  - Java Objects

We used JMeter in order to gather and collect dataset of OTT system. The performance dataset is collected by running the OTT and JMeter which creates virtual users to mimic the system on production environment. We collected a performance dataset that consists of 205 instances containing the features #users, system response time, system throughput, application server utilization, and database server utilization.

## 6.4 Using Statistical Machine Learning Multivariate Regression:

Table 6.1 shows a sample of OTT performance dataset for prediction system response time. To predict the performance of the system, we train the machine learning by dividing the dataset into 70% of the dataset for training and 30% for testing.

**Table 6.1 Sample of OTT response time dataset**

<b>users</b>	<b>respons_time</b>	<b>throughput</b>	<b>app_UTLI</b>	<b>DB_UTLI</b>
1	1082	20.8	0	0.01
2	1086	1.3	0.02	0.01
3	1087	1.7	0.03	0.02
4	1086	2.2	0.03	0.02
10	1094	5	0.08	0.04
18	1085	8.7	0.1	0.05
19	1082	9.3	0.12	0.05
22	1080	10.9	0.14	0.05
23	1080	11.2	0.15	0.04
27	1078	13.4	0.13	0.03
29	1077	14.1	0.16	0.04
35	1074	17	0.17	0.12
39	1074	18.8	0.19	0.08
41	1075	20.1	0.25	0.11
42	1077	20.4	0.25	0.13
46	1072	22.4	0.31	0.12
49	1078	23.8	0.3	0.15

53	1072	25.8	0.32	0.18
61	1073	29.6	0.37	0.28
63	1073	30.7	0.42	0.19
64	1070	31.2	0.5	0.25
31	?	46.5	2.25	1.65
315	?	49.2	2.33	1.64
321	?	47.9	2.18	1.48
322	?	46.7	2.96	1.89
325	?	44.9	2.91	1.83
331	?	47.3	2.77	1.85
332	?	48.1	2.75	1.77
333	?	46.7	2.9	1.96
334	?	47.6	2.86	1.84
335	?	44	2.68	2
336	?	45.3	2.79	1.88
337	?	48.7	2.8	1.7
341	?	48.4	2.76	1.81
344	?	48.2	2.85	1.8
348	?	47.3	2.83	1.79
350	?	50	2.95	1.87
351	?	47.3	3.19	2.07
355	?	49.7	2.93	1.87
356	?	46.4	3.12	1.94

....	....	....	....	....
....	....	....	....	....
501	?	46.3	6.76	4.42
505	?	49.6	5.67	3.79
508	?	47.4	9.26	6.03
512	?	49.6	6.17	4.12
515	?	50.9	5	3.19
516	?	42.1	5.43	3.5
519	?	51.3	4.79	3.36
545	?	84.9	5.4	3.45
549	?	82.3	6.11	4.27
550	?	83	10.22	7.03
555	?	86.2	5.2	3.7
559	?	83.8	5.12	3.53
561	?	83.9	8.39	5.72
569	?	81.1	5.04	3.58
575	?	84.5	12.31	8
578	?	85.7	6.36	4.39
579	?	84.9	5.48	3.5
580	?	82	10.15	6.86
584	?	81.6	5.99	4
587	?	85.5	5.23	3.55
590	?	84.6	4.85	3.4

593	?	81.3	13.3	8.62
597	?	83.2	5.73	3.74
600	?	86.1	10.23	7.09
-	-	-	-	-

### 6.5 Using Machine Learning for Performance Risk Assessment:

At the first part of our methodology we are become able to predict any of performance indices from dataset such as system response time, system throughput, or each of server utilization by using multivariate regression. At the second part of our methodology we will become able to classify the performance risk for each instance by using either Naïve Bays, or KNN, or SVM techniques. Fig. 6.2 shows a sample of dataset to predict performance risk level.

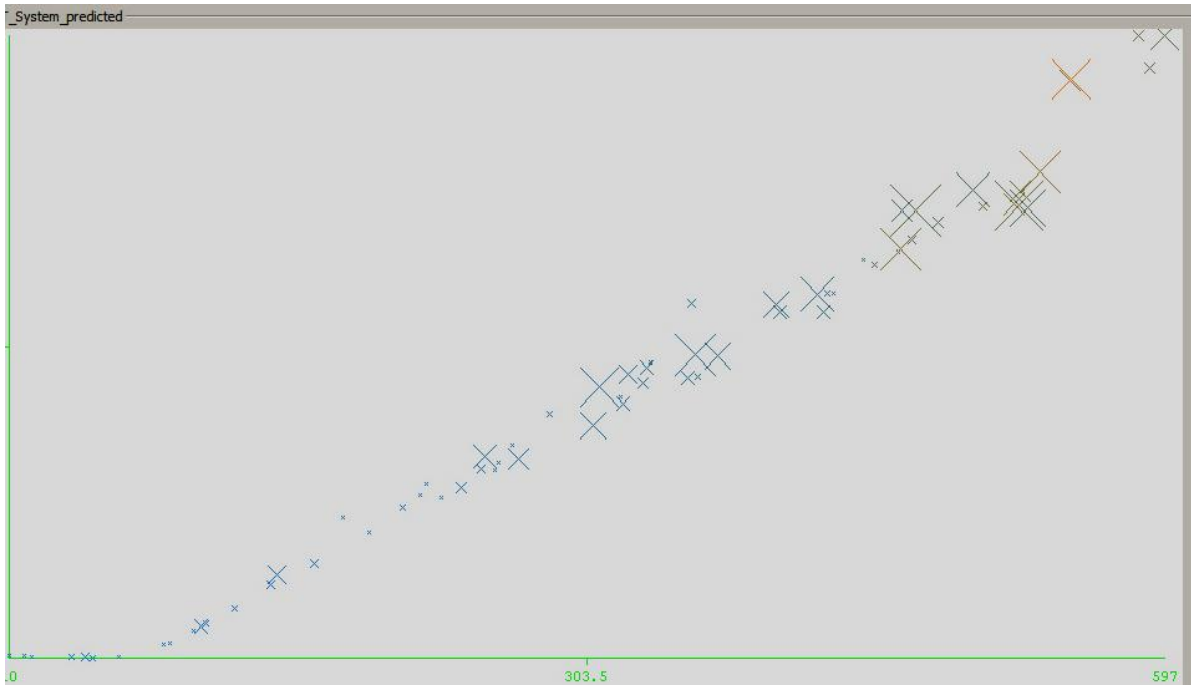
**Table 6.2 Sample dataset of OTT system for performance risk classification**

Users	Resp	Throu	Apputi	dbuti	class
555	5751	86.20	5.20	3.70	HR
600	6386	86.10	10.23	7.09	HR
584	6639	81.60	5.99	4.00	HR
593	6623	81.30	13.30	8.62	MR
569	6657	81.10	5.04	3.58	MR
523	5116	51.80	5.32	3.79	HR
355	3573	49.70	2.93	1.87	LR
528	5094	51.60	4.87	2.99	HR
521	5129	51.30	6.41	4.34	HR

519	5143	51.30	4.79	3.36	?
515	5058	50.90	5.00	3.19	?
525	5246	50.70	5.87	3.86	?
461	4630	50.60	4.34	3.04	?
524	5245	50.60	6.44	4.11	?
411	4092	50.40	3.10	2.08	?
424	4158	50.40	3.04	2.10	?
530	5235	50.30	5.02	3.38	?
469	4812	50.10	4.37	2.95	?
538	5417	50.10	5.41	3.47	?
350	3551	50.00	2.95	1.87	?
534	5421	50.00	7.91	4.94	?
360	3591	49.90	2.77	1.89	?
454	4606	49.90	4.41	2.88	?

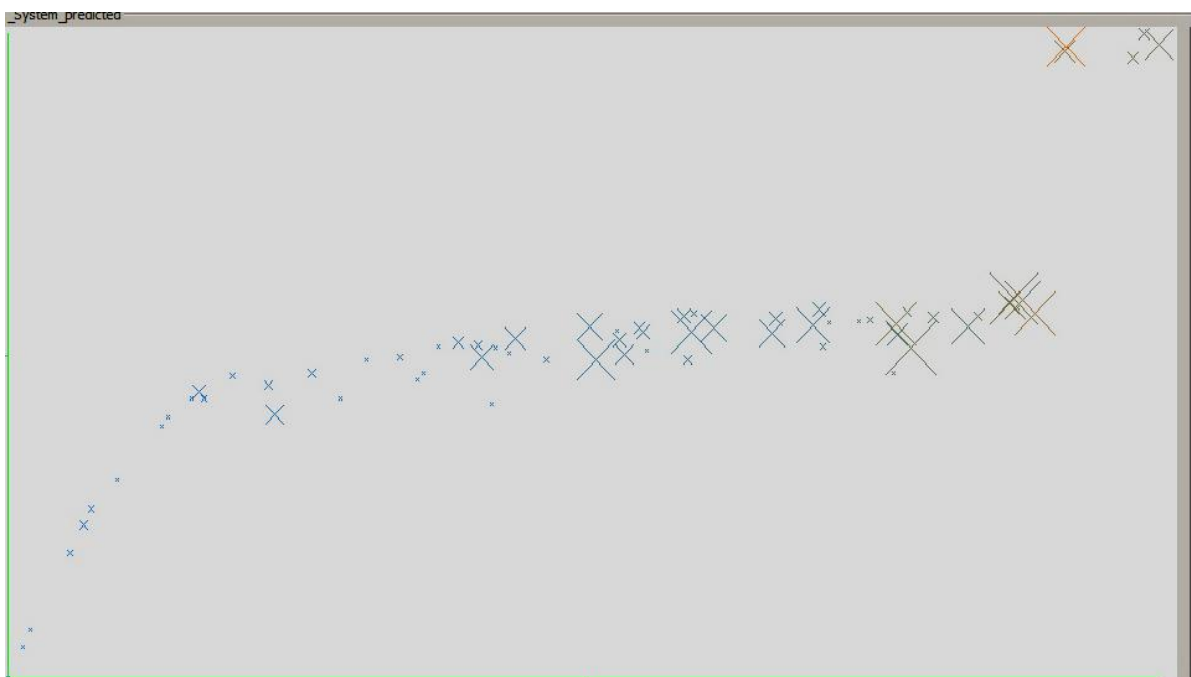
## 6.6 Using WEKA to Visualize the Dataset

We used WEKA to visualize and validate the dataset. Fig 6.4 shows the response time diagram, the diagram represents the relationship between the numbers of users and response time values. It looks clear that the response time increases with number of users linearly and then exponentially after saturation point.



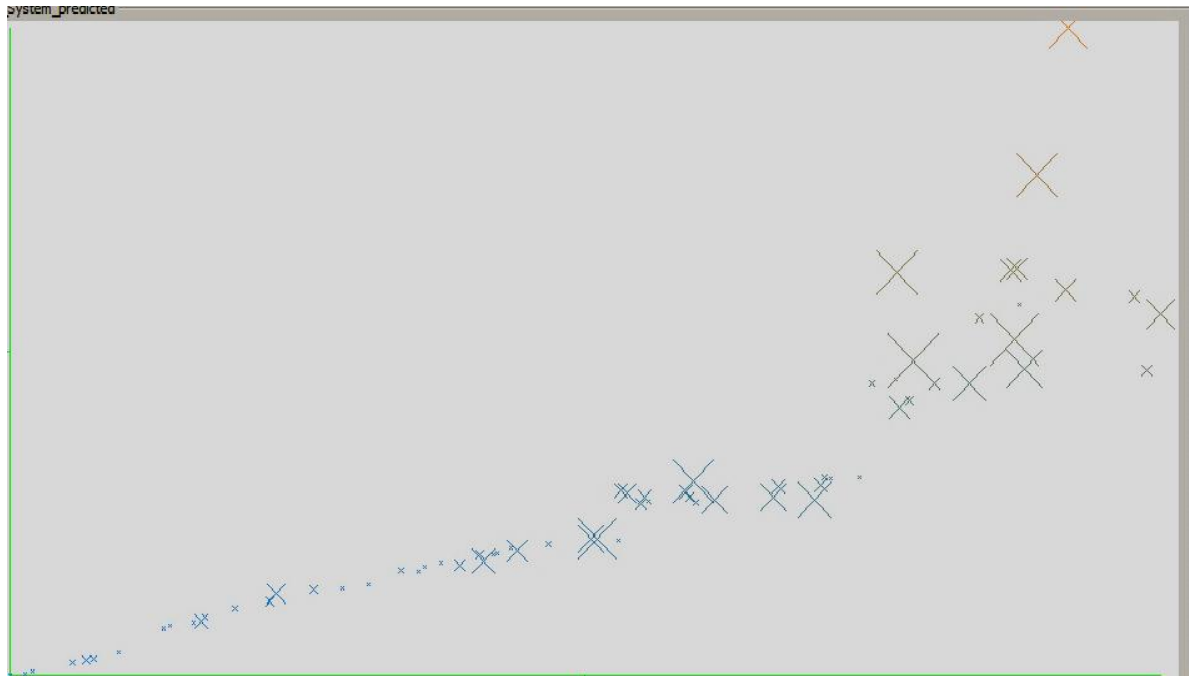
**Figure 6.4 OTT System Response Time - Performance Dataset Visualized by WEKA**

Fig. 6.5 shows the system throughput. The throughput increased as the number of users increase till it reached stabilized point when the numbers of users reach certain.



**Figure 6.5 OTT System Throughput – Performance Dataset Visualized by WEKA**

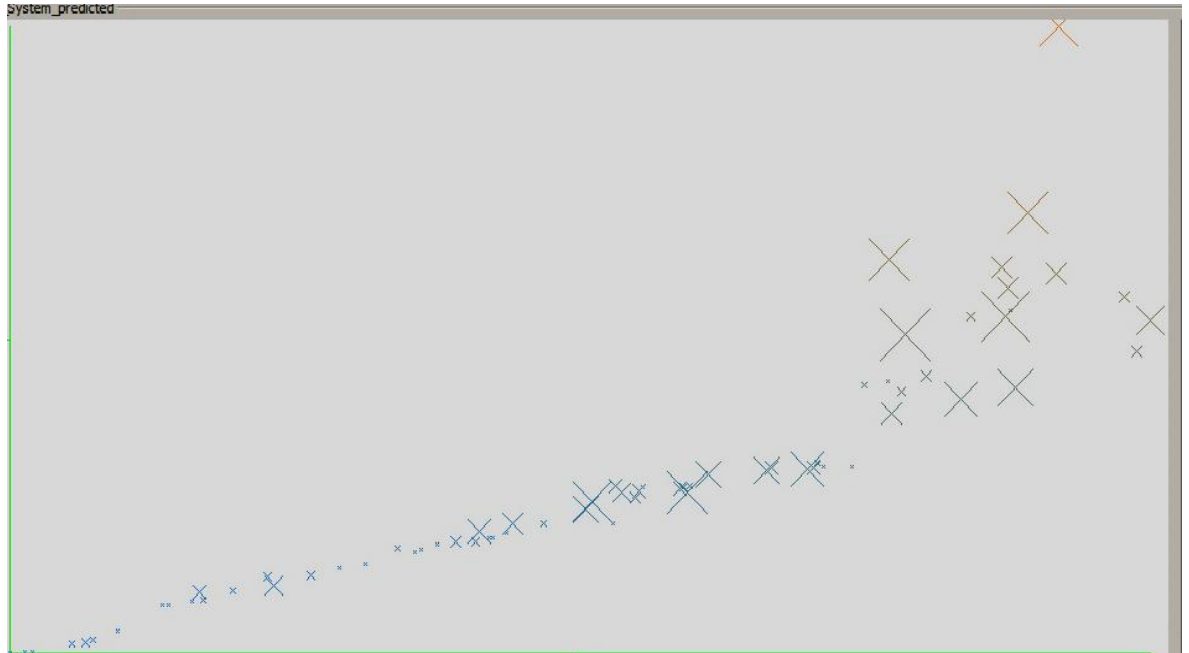
Figure 6.6 represents the application server utilization. It is obvious that system reaches saturation point when a specific number of users access the system concurrently.



**Figure 6.6 Application server utilization – performance dataset visualized by WEKA**

Fig. 6.7 represents the database server utilization. The utilization of server increased when the number of customer grows till it reaches saturation point.





**Figure 6.7 Using multivariate regressions to predict OTT system response time**

## 6.7 Using Multivariate Regression

Table 6.3 shows the correlation coefficient between actual response time and predicted response time. The high value of the correlation coefficient indicates the small difference between the actual response time and the predicted response time.

**Table 6.3 Using multivariate linear regression to predict OTT system response time**

=== Evaluation on test split ===	
=== Summary ===	
Correlation coefficient	0.7
Mean absolute error	4.1
Root mean squared error	5.1
Total Number of Instances	205

## 6.8 Prediction Performance Risk Class

### 6.8.1 Using Naïve Bayes Technique

Table 6.4 states the usage of Naïve Bayes technique as simple technique to assess performance risk. The result shows the percentage of the error as 6.5% which is good result, but due to instability of the algorithm we make validation using more techniques.

**Table 6.4 Using Naïve Bayes to classify Performance risk level**

=== Evaluation on test split ===		
=== Summary ===		
Correctly classified Instances	57	93.4 %
Incorrectly Classified Instances	4	6.5 %
Mean absolute error	0.04	
Root mean squared error	0.18	
Total Number of Instances	61	

Table 6.5 shows the confusion matrix of Naïve Bayes technique. As listed on the table there are two instances are classified as low performance risk but in reality they are medium performance risk. Moreover, there are two states are classified as medium performance risk while actually they are high performance risk.

**Table 6.5 Confusion Matrix for Naïve Bayes Algorithm**

=== Confusion Matrix ===			
A	b	C	
10	2	0	a = LR
0	9	2	b = MR
0	0	38	c = HR

## 6.8.2 Using KNN Technique

Table 6.6 presents the results of usage KNN technique. The algorithm shows error 3.27% which looks better than Naïve Bayes.

**Table 6.6 Using KNN technique to classify performance risk level**

=== Evaluation on test split ===		
=== Summary ===		
Correctly Classified Instances	59	96.72 %
Incorrectly Classified Instances	2	3.27 %
Mean absolute error	0.03	
Root mean squared error	0.14	
Total Number of Instances	61	

Table 6.7 shows the confusion matrix of KNN. As stated on the table there is one instance is classified as low performance risk while actually it must be medium performance risk. In addition, one instance is classified as medium performance risk while in reality it must be high performance risk.

**Table 6.7 Confusion Matrix for KNN Technique**

=== Confusion Matrix ===			
A	b	C	
11	1	0	a = LR
0	10	1	b = MR
0	0	38	c = HR

### 6.8.3 Using SVM Technique

Table 6.8 shows the result of performance risk classification using SVM technique. The high percentage of error 37.70% explains the inability of the technique to give accurate result.

**Table 6.8 Using SVM technique to classify performance risk level**

=== Evaluation on test split ===		
=== Summary ===		
Correctly classified Instances	38	62.29 %
Incorrectly Classified Instances	23	37.70 %
Mean absolute error	0.25	
Root mean squared error	0.50	
Total Number of Instances	61	

Table 6.9 presents the confusion matrix of the SVM technique. It is obvious that SVM technique failed to classify the instances as it classifies all instances as high performance risk.

**Table 6.9 Confusion Matrix for SVM technique**

=== Confusion Matrix ===			
A	b	C	
0	0	12	a = LR
0	0	11	b = MR
0	0	38	c = HR

Table 6.10 presents the comparison between the three machine learning techniques. Naïve Bayes works probably and gives a good result, but it considered as instable machine learning technique and simple. We focus on KNN as stable technique that give good results with all three case studies. Moreover, KNN gives prediction accuracy 96.72% which considered as high result among the three techniques.

**Table 6.10 Comparison of the classifiers techniques**

Evaluation criteria	Classifiers		
	Naïve Bayes	KNN	SVM
Timing to build model (in sec)	0.01	0.01	0.27
Correctly classified instances	57	59	38
Incorrectly classified instances	4	2	23
Prediction accuracy	93.4%	96.72%	62.29%

Fig. 6.8 presents the detailed accuracy by class. As per the figure the precision of the accuracy is very high at LR zone as the relation is linear, while the precision degrades at MR and HR as the relation moved to exponential relation.

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.859	0	1	0.859	0.924	0.962	LR
	0	0	0	0	0	0.665	MR
	1	0.31	0.612	1	0.759	0.845	HR
Weighted Avg.	0.792	0.102	0.741	0.792	0.748	0.885	

Figure 6.8 Detailed accuracy by class

## 6.9 Summary

This chapter considered as third validation work for our approach, we apply the approach on a running OTT system. The OTT is already developed and working system serving university to inform teachers, students, administration, and parents about class times. We have generated a dataset from the system by using Apache JMeter. JMeter is a Java based tool to mimic virtual users accessing the system at same time. Furthermore, we used machine learning techniques to visualize, analyze, predict, and assess performance risk. After we performed our experiments we made comparison between the three mentioned above machine learning techniques. KNN gives low percentage of error about 3.27 % which considered a promising result.

## CHAPTER VII

### DISCUSSION AND RESULTS

#### 7.1 Introduction

Performance risk prediction and assessment is very often ignored when designing the software system, this due to invisibility of most units of software system. Bad performance of software system is not immediately obvious and tangible, moreover correcting the performance problems afterwards can be just as costly and difficulty as stated by Brebner et al. (2009). Furthermore, the complexity of modern software systems makes it hard to understand how the system will perform under a changed load, or even after changes on the software or hardware. Existing approaches, methods, and tools are either tailored for a very special type of application, or they come with a variety of configurations, resulting in the need of expert knowledge to operate them.

Not knowing the performance behavior of software system thought is a disaster and big risk. In order to contribute to these challenges, we develop an approach to predict performance risk of application software at early stages of SDLC and before software implementation. The approach applied on three case studies: a hospital system, an e-commerce system, and already running online timetable system. The produced results are good and give promising results, moreover this contribution can open a new area for researchers to design and build a tool that combine queuing network model and machine learning to easy prediction of software performance at the planning stage.

In this chapter we will discuss our findings from the thesis and explain what does it means. Furthermore, this chapter highlights the stages of preparing the dataset that used for performance risk prediction. In addition, the chapter compares the results of the approach with the related work.

## **7.2 Dataset Preparation**

The process of using machine learning in research may not be exactly the same, but there are certain standard and necessary steps:

- a) Define the problem
- b) Prepare the dataset
- c) Evaluate machine learning techniques
- d) Improve the results
- e) Present the results.

In order to achieve reliable predictions from a machine learning model, it is very important to handle and organize the data precisely. There are three steps involved in preparing a dataset for use by a machine learning algorithm. First, selecting the data and then process and transform the data.

### **7.2.1 Selecting Data**

Data which are relevant to the problem chosen should be selected. Sometimes it is good to collect all the relevant data that are available for the problem domain because it helps to train the model well. The preparation of dataset started when building the performance model using JMT. JMT is a queuing tool to model software performance by implementing several algorithms for the exact, asymptotic and simulative analysis of queuing models. By changing workloads the performance of the software recorded on dataset contains more than 250 instances. Moreover, there are five numeric attributes such as number of users, system response time, system throughput, application server utilization, database server utilization, and the sixth attribute is classification attribute represents the class of performance that ranking risk into three levels {low, medium, high}.



### 7.2.2 Processing Data

After data collection, the next step focuses on the utilization of data by the learning algorithms. Data processing helps to build and create a framework for the collected data in order for the algorithm to work efficiently on the data. There are steps for processing data:

**(a) Formatting:** The raw data which has been collected from JMT was not in suitable format for use by Weka, therefore formatting the data according to the needs of Weka.

**(b) Cleaning:** There may be some incomplete or missing data in the raw data. A process has been done to fix or remove the missing data in order to make the dataset consistent and useful for the performance risk prediction and assessment.

For the sake of dataset validation we use statistical method to estimate the accuracy of the models that we created on unseen data. To be sure about the accuracy of the best model, we evaluated it on actual unseen dataset. To do this, we took some dataset that algorithms will not see and use this data later to get a second and independent idea of how accurate the best model really is. Dataset will be split into two, 80% - 70% of which used to train our models and 20% - 30% of which used to hold back as a validation and testing dataset.

### 7.3 Comparison with Existing Studies

Several studies were carried out in the past few years to predict software performance risk. In this part of the research we will conduct a detailed comparison between our work and work done by Ganapathi in 2011 and Archana in 2007. There are partly differences in their research questions and scope from our work, but are anyway useful benchmarks to test the validity of our results. Important differences are the following:

- **Time of usage:**

This feature indicates the phase of SDLC at which the approach is applied. “Early” means the application of approach at the planning phase and before the software

being constructed. In contrast “Late” means application of the approach after software constructed.

- **Number of concurrent users:**

This feature indicates the capacity of the approach and state the number of users who can concurrently use the software application.

- **Cost:**

Cost means how much the process of applying software performance evaluation method costs in terms of consuming time and resources.

- **Techniques:**

This feature states the tools and methods used in the approach. Techniques are varying such as simulation, execution graph, Markov chain, queuing network model. Modern techniques of software quality attributes measurements have been started using artificial intelligence techniques such as neural network, multi-agent system, and machine learning in order to predict many software quality attributes such as maintainability, performance, security.

- **Easy of Configuration:**

Easy of configuration state the effort needed to setup and use the approach. This criterion is very important as many of these approaches were ignored because of it is complexity and difficulty.

- **Result:**

Performances risk assessments has different scope and domain, in this comparisons we tried to find the most related approaches that have similar concern.

- **Modeling Language:**

Using modeling language such as UML means the planning for performance assessments starts at early phases of SDLC. Our approach focus on performance prediction based at modeling stage.

- **Number of Activity:**

Software performance evaluation requires a number of activities in order to be achieved. Each step has input, process, and output. Some of the methods perform these steps sequentially or others iteratively or in parallel way.

**Table 7.1 Comparison with Existing Studies**

<b>Criteria</b>	<b>Ganapathi</b>	<b>Archana</b>	<b>Our Approach</b>
Time of usage	Late	Early	Early
Number of concurrent users	Large	Small	Large
Cost	Expensive	Not Expensive	Not Expensive
Techniques	Statistical Machine Learning	Execution Graph and Simulation	QNM and Machine Learning
Easy of configuration	Complex	Easy	Easy
Results	Performance Prediction	Risk Assessment	Performance prediction and risk estimation
Usage of Modeling Language	No	Yes	Yes
Number of Activities	4 steps	5 steps	5 steps

In contrast, there are major differences between other related works in the literature we didn't mention them. Firstly, the work of Dubach and et al. used machine learning technique to explore the good compiler architecture design. Architecture design of compiler is far away from our scope as we focused on n-tier application architecture. Moreover, their dataset features are completely different that contains features such as processor cycles, energy consumption, and the trade of the two characteristics. Secondly, the work of Malhotra and et al. applied on prediction of different non-functional requirement that is maintainability using machine learning techniques. Thirdly, the work of Ipek and et al. used

multilayer neural network, the network trained on input data collected from execution on targeted platform. However, the work used to predict performance of parallel application SMG2000 after the application run on working environment.

#### 7.4 Result Discussion

In this section, the researcher analyzes and discusses the result of three case studies chapter four the hospital system, chapter five the e-commerce system, and chapter six the OTT system. The comparison conducted using KNN technique as it gives good and stable results among the techniques that have been used. We used criterion such as number of dataset instances, timing to build the model, correctly classified instances, incorrectly classified instances, and the accuracy percentage to show major differences.

**Table 7.2 Comparison between case studies using KNN**

<b>Criteria</b>	<b>Case Study 1 Hospital System</b>	<b>Case Study 2 Ecommerce System</b>	<b>Case Study 3 OTT System</b>
<b>Number of dataset instances</b>	196	250	205
<b>Timing to build the model</b>	0.01	0.01	0.01
<b>Number of correctly classified instances</b>	54	83	95
<b>Number of incorrectly classified instances</b>	4	2	2
<b>Accuracy Percentage</b>	93.1 %	97.6 %	96.7 %

The table 7.2 shows that case study No.2 on ecommerce system gives high accuracy prediction percentage reaches up to 97.6 % with dataset splitting 70 % as training set and 30 % as test set. This indicates that the relation between accuracy percentage of the result and the training dataset size is proportionally.

### **7.5 Summary**

In this chapter a deep comparison has been conducted between our approach and two previous approaches. The results stated that our approach has more functions than the previous approaches; firstly, it can predict resource utilization, response time, throughput of the software application. Secondly, it can predict and rank performance risk into three levels low, medium, and high risk. Moreover, regarding to the capacity and number of concurrent users our approach can handle large number of users compared to the work of Ganapathi and Archana.

## **CHAPTER VII**

### **CONCLUSION AND FUTURE WORK**

This research proposed a new approach for model based resource utilization prediction and performance risk assessment using machine learning techniques. We build models for three case studies using Java Modeling Tool (JMT) as performance modeling tool. Furthermore, we generated dataset from JMT by recording the performance indices such as system response time, system throughput, and resources utilization corresponding to their workloads. Based on the generated datasets we can build machine learning model to predict the new instances of workload from previous recorded workloads.

To validate our work, we applied the approach on three different systems: hospital system as small closed queuing system, ecommerce system as large opened queuing system, and OTT as already implemented and running system. Moreover, we predicted the resource utilization, system response time, system throughput, and performance risk with accuracy ranged from 93.1 % up to 97.6 %. Based on the three case studies, the research shown that machine learning (ML) based approach provides a high degree of automation, not require too much ML expertise, and work better than many alternatives.

The steps of the approach started by designing annotated UML diagrams to analyze the system. Correspondingly, these diagrams transformed to performance models in order to produce a large dataset that contains the related performance indices corresponding to the workloads.

In chapter four we applied our methodology on a hospital system. Using information available prior to system software put on production environment; we were able to predict multiple performance metrics such as response time, system throughput, and resource utilization. In this chapter we achieved prediction accuracy with error percentage 0.21%.

The second validation for our approach is in chapter five which we used the steps of our approach for predicting performance of e-commerce system. As performance considered as important attribute for e-commerce website and frustrated customer can leave website of the company and move to other company website. In this case, we achieved very high prediction accuracy with error percentage 0.12 %.

In chapter six, we performed the third case study for our methodology on OTT system. The OTT is a system that already running, designed and implemented to serve a university students, staff, and parents. By using our approach we achieved prediction accuracy with error percentage 0.14%.

Chapter seven we conducted comparisons between our work and two different related work. We found that our approach has many advantages over the current work, as it focuses on prediction of performance risk at early stages of the SDLC. Furthermore, we extended the approach by adding ML to increase the accuracy and capacity of the approach.

In the future researches, we suggest that our machine learning based approach can provide powerful tools for managing and analyzing state-of-art systems. We expect our approach to have scope beyond the three success cases presented in the research. Moreover, we believe that queuing theory should join machine learning in essential toolkit that system researches and practitioners should use. The produced tool can be intelligent tool to predict performance risk and resource utilization of software application at design time.

In like manner, with growing work on the internet of things technology and the emergence of critical timing applications, our approach can be used in wide range of this area. Examples of these applications are autopilot autonomous cars; autopilot car is a

vehicle that is capable of sensing its environment and navigating without human input. Our approach can be used to predict the response time of application to outside events. Moreover, in Tele-health & remote patient monitoring application, this type of application connects any wearable or portable device to the cloud, pulls and analyzes collected patient data in real time, and also it monitors patients at home using live video and audio streaming.



## REFERENCES

- Abe, S., 2015. Fuzzy support vector machines for multilabel classification. *Pattern Recognition*, 48(6), pp.2110–2117. *International Journal of Computer Applications (0975 – 8887). Volume 131 – No.3* Available at: <http://dx.doi.org/10.1016/j.patcog.2015.01.009>.
- Adhianto, L. et al., 2010. HPCTOOLKIT: Tools for performance analysis of optimized parallel programs. *Journal Concurrency Computation Practice and Experience*, 22(6), pp.685–701.
- Balsamo, S. et al., 2004. Model-based performance prediction in software development: a survey. *IEEE Transactions on Software Engineering*, 30(5), pp.295–310.
- Based, N., 2004. Neighbor Based Algorithm for Multi-label Classification. , pp.718–721. *Granular Computing, 2005 IEEE International Conference*
- Brunnert, A. & Krcmar, H., 2015. Continuous performance evaluation and capacity planning using resource profiles for enterprise applications. *Journal of Systems and Software*. Available at: <http://dx.doi.org/10.1016/j.jss.2015.08.030>.
- Canessane, R.A. & Srinivasan, S., 2013. Performance Optimization of Software Design using Queuing Networks. , 2(12), pp.3390–3395. *International Journal of Engineering Research & Technology (IJERT) · Vol. 2 Issue 12*
- Canevet, C. et al., 2003. Performance modelling with UML and stochastic process algebras. *IEE Proceedings: Computers and Digital Techniques*, pp.1–14. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.65.5340&rep=rep1&type=pdf%5Cnpapers://c142b844-46f9-47f1-9c59-24141d995c3f/Paper/p5190>.
- Chen, M. & Ma, Y., 2013. A Hybrid Approach to Web Service Recommendation Based on QoS-Aware Rating and Ranking. *Services Computing (SCC), 2015 IEEE International Conference* , pp.1–23.

- Cortellessa, V. et al., 2005. Model-Based Performance Risk Analysis. *Journal IEEE Transactions on Software Engineering archive*, 31(1), pp.3–20.
- Franks, G. et al., 2013. Layered Queueing Network Solver and Simulator User Manual. *Journal of Systems and Software*, v.80 n.4, p.510-527
- Garcia, V., Debreuve, E. & Barlaud, M., 2008. Fast k Nearest Neighbor Search using GPU. *Journal of the ACM*, vol. 45, pp. 891–923, 1998
- Guo, J. et al., 2012. Variability-Aware Performance Modeling: A Statistical Learning Approach. , (August). *Automated Software Engineering Journal*, 2017. Volume 8800, 2014
- Ionita, M.T. & Hammer, D.K., 2002. Scenario-based software architecture evaluation methods: An overview. *International Journal of Pharmacy and Technology* 8(4):25720-25733 Sara 2002 @Icse. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.8382&rep=rep1&type=pdf>.
- Islam, B., 2013. Predict Software Reliability by Support Vector Machine. *The International Journal of Advanced Manufacturing Technology* February 2006, Volume 28, *Issue 1–2*, pp 154–161, 2(4), pp.46–52.
- Izzeldin, R. & Osman, M., 2010. Performance Modelling of Database Designs using a Queueing Networks Approach Performance Modelling of Database Designs using a Queueing Networks Approach. *Journal of Systems and Software*, v.82 n.3
- Kattepur, A., Nambiar, M. & Kattepur, A., 2015. Performance Modeling of Multi-tiered Web Applications with Varying Service Demands Performance Modeling of Multi-tiered Web Applications with Varying Service Demands. *International Journal of Networking and Computing*(6)(2016)
- Keung, J.W. & Nguyen, T., 2010. Quantitative Analysis for Non-linear System Performance Data Using Case-Based Reasoning. *Software Engineering Conference (APSEC), 2010 17th Asia Pacific*, pp.346–355.

- Kotsiantis, S.B., 2007. Supervised Machine Learning: A Review of Classification Techniques. *Proceedings of the 2007 conference on Emerging Artificial Intelligence*, 31, pp.249–268.
- Li, J. et al., 2009. Machine learning based online performance prediction for runtime parallelization and task scheduling. *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, 1(c), pp.89–100.
- Magalhães, D. et al., 2015. Workload modeling for resource usage analysis and simulation in cloud computing. *IEEE Transactions on Cloud Computing ( Volume: 2, Issue: 2, April-June 1 2014 )*, 47, pp.69–81.
- Marzolla, M., 2010. The qnetworks Toolbox : A Software Package for Queueing Networks Analysis Moreno Marzolla Technical Report UBLCS-2010-04 Department of Computer Science. , (February). *International Conference on Analytical and Stochastic Modeling Techniques and Applications\_ ASMTA 2010: Analytical and Stochastic Modeling Techniques and Applications* pp 102-116
- Mohanty, R., 2012. Classification of Web Services Using Bayesian Network. *Journal of Software Engineering and Applications*, 5(4), pp.291–296. Available at: <http://www.scirp.org/journal/PaperDownload.aspx?DOI=10.4236/jsea.2012.54034>.
- Moniem, H.A., 2015. A framework for Performance Prediction of Service-Oriented Architecture. *International Journal of Computer Applications Technology and Research Volume 4– Issue 11, 865 - 870, 2015, , 4(11)*, pp.865–870.
- Moniem, H.A., 2014. Performance Prediction of Service-Oriented Architecture - A survey. , 3(12), pp.831–835. *INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION, VOL 1 (2017) NO 3*
- Omary, Z. & Mtenzi, F., 2009. Dataset threshold for the performance estimators in supervised machine learning experiments. *2009 International Conference for Internet Technology and Secured Transactions, (ICITST)*, 3(3), pp.314–325.

- Omary, Z. & Mtenzi, F., 2010. Machine Learning Approach to Identifying the Dataset Threshold for the Performance Estimators in Supervised Learning. *International Journal*, 3(3), pp.314–325.
- Prof, E. et al., Proactive Prediction Models for Web Application Resource Provisioning in the Cloud., *Service Oriented System Engineering (SOSE), 2013 IEEE 7<sup>th</sup> Conference*
- Rabta, B., Alp, A. & Reiner, G., Queueing Networks Modeling Software for Manufacturing. *International journal of production research* 24(6):1485–1503.
- Radhakrishnan, A. & Virginia, W., 2007. Tool Support for Software Performance Risk Assessment Tool Support for Software Performance Risk Assessment. *IEEE Access ( Volume:5) Journal*
- Rajagopal, D. & Thilakavalli, K., 2017. A Study : UML for OOA and OOD. , 7(2), pp.5–20. *International Journal of Knowledge Content Development and Technology*, 2017;7(2)
- Ram, C. et al., 2011a. E Arly Performance Prediction of Web Services. , 2(3), pp.31–41., *International Journal on Web Service Computing (IJWSC)*, Vol.2, No.3, September 2011
- Ram, C. et al., 2011b. P REDICTING P ERFORMANCE OF W EB S ERVICES USING SMTQA. *The Journal of Supercomputing archive* Volume 71 Issue 2, February 2015Pages 673-696 , pp.58–66.
- Salih, H.A.M. & Ammar, H.H., 2017. Model-Based Resource Utilization and Performance Risk Prediction using Machine Learning Techniques. , 1(3), pp.101–109., *INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION, VOL 1 (2017) NO 3*
- Serazzi, G., 2008. Performance Evaluation Modelling with JMT : learning by examples.
- Shoaib, Y. & Das, O., 2011. Web Application Performance Modeling Using Layered Queueing Networks. *Electronic Notes in Theoretical Computer Science*, 275(1),

pp.123–142. Available at: <http://dx.doi.org/10.1016/j.entcs.2011.09.009>.

Singh, H. & Kumar, S., 2011. Dispatcher Based Dynamic Load Balancing on Web Server System. , 4(3), pp.89–106.

Singh, K. et al., 2007. Predicting parallel application performance via machine learning approaches: Research Articles. *Concurr. Comput. : Pract. Exper.*, 19(17), pp.2219–2235. Available at: <http://dx.doi.org/10.1002/cpe.v19:17>.

Smith, C., 1981. Software Performance Engineering. *Computerworld*. Available at: <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:SOFTWARE+PERFORMANCE+ENGINEERING#0%5Cnhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Software+Performance+Engineering.#0>.

Tertilt, D. & Krmar, H., 2011. Generic Performance Prediction for Erp and Soa Applications. *Proceedings of the 19th European Conference on Information Systems ECIS 2011 ICT and Sustainable Service Development*, p.197. Available at: <http://aisel.aisnet.org/ecis2011/197/>.

Tribastone, M., Mayer, P. & Wirsing, M., 2010. Performance prediction of service-oriented systems with layered queueing networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6416 LNCS(PART 2), pp.51–65.

Zhang, Q., Cherkasova, L. & Smirni, E., 2007. A Regression-Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications, ICAC '07 Proceedings of the Fourth International Conference on Autonomic Computing, June 11 - 15, 2007