ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE
International Journal of Advanced Research in Computer and Communication Engineering
ICRITCSA
M S Ramaiah Institute of Technology, Bangalore
Vol. 5, Special Issue 2, October 2016

Early Performance Evaluation of Data Warehouse Systems: From UML to LQN models

Dr. Madhu Bhan[1], Dr. K. Rajanikanth[2], Dr. T.V. Suresh Kumar[3]

Assistant Professor, Department of Computer Applications, M.S. Ramaiah Institute of Technology, Bangalore, India[1]

Professor, Department of Information Science, M.S. Ramaiah Institute of Technology, Bangalore, India[2]

Professor, Department of Computer Applications, M.S. Ramaiah Institute of Technology Bangalore, India[3]

Abstract: If the performance of a Data Warehouse System is determined to be unacceptable, at the time of "acceptance testing" it can result in very expensive redesign and consequent delayed delivery or, in the worst case, complete non-use of the system! There is clearly a need for tools and techniques that enable performance analysis of designs to be done easily and reliably throughout the development process of Data warehouse systems. In this paper we demonstrate the derivation of Layered Queuing Network (LQN) Performance Models from a set of UML diagrams and an algorithm for deriving LQN model. LQN model is a very useful tool to analyse the performance of a system from abstract model so that the developer of Data warehouse systems is able to understand performance effects of various design decisions starting at early stages when changes are easy and less expensive.

Keywords: Data warehouse Systems; Software Performance Prediction; UML; Queuing models.

I. INTRODUCTION

The most popular definition of Data Warehouse comes from Bill Inmon who says, "A Data Warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process" [1]. Such systems are primarily used for business intelligence tasks such as analysing statistics related to business facts like sales, profits, customer choices along different dimensions like time, region etc. Data to support such tasks is extracted from "heterogeneous operational systems and other legacy systems" and is stored in Data Warehouse. The process of extracting data from the source systems and transforming it into an appropriate format and finally loading it into a Data Warehouse is called as ETL (Extract, Transform, Load). While operational systems maintain current information, a Data Warehouse is a very large database containing historical data. While operational systems maintain current information, a Data Warehouse is a very large database containing historical data. Data Warehouse systems are highly complex systems with their performance depending on many interacting factors such as the nature of the query workload, the views materialized, the index structures built on the base tables. Thus assessing the performance of these systems is also a complex problem. Measuring the performance of a built system a-posteriori using suitable instrumentation is feasible. However, should the results of such measurement indicate mismatch between the performance characteristics of the system and expectations of the designers and / or users, it would be highly expensive to alter the implementation of the system.

Thus, after making initial architectural and design decisions it is desirable to assess the performance of the proposed system before proceeding to the implementation stage [2]. Such an early performance assessment would allow economical exploration of alternative choices of architectures, designs, hardware and software components. While this is indeed an extremely difficult problem, this paper discusses the basic operations for converting a UML model to a performance model in order to conduct a quantitative performance analysis. A case study of Data warehouse system is used to illustrate the transformation procedure.

II. SOFTWARE PERFORMANCE ENGINEERING

Software Performance Engineering (SPE), an approach introduced by C.U. Smith, proposes to use quantitative methods and performance models in order to assess the performance effects of different architectures, designs and implementation alternatives during the development of a software system. SPE supports the idea that integrating performance analysis into the software development process, from the initial stages to the end, can ensure the system to meet its performance goals. This would eliminate the need for "late-fixing" of performance problems, a frequent practical approach that postpones any performance concerns until the system is completely implemented. Late fixes tend to be very expensive and inefficient, and the product may never reach its original performance requirements. SPE is a software oriented approach that focuses on architecture, design and

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

implementation choices. It uses model predictions to evaluate trade-offs in software functions, hardware size, quality of results and resource requirements. The models assist developers by enabling them to select architecture and design alternatives with acceptable performance characteristics. These models avoid in tracking performance throughout the development process and prevent problems from surfacing late in the life cycle when they are difficult and expensive to correct. The process of building a system's performance model before the system is completely implemented starts with identifying a small set of key performance scenarios representative of the way in which the system will be used [3],[4]. The performance analysts must understand first the system behavior for each scenario by talking with the system developers and/or by using design specifications.

The analyst, helped by the developers, will follow the execution path through the software for each scenario, from component to component, identifying the quantitative demands for resources made by each component (such as CPU execution time and I/O operations), as well as the various reasons for queuing delays (such as competition for hardware and software resources). The scenario descriptions thus obtained are mapped onto a performance model. By solving the model, the analyst will obtain performance results such as response times, throughput, utilization of different resources by different software components, etc. Trouble spots can be thus identified, and alternative solutions for eliminating them assessed in a similar way.

III. LQN MODELS

In general, a performance model can be classified either as an analytic or as a simulation model. While an analytical model captures the essence of modeled system as a set of mathematical equations, a simulation model mimics the structure and behavior of the real system, Some well-known examples of analytic performance models are Queuing Network Models (QN) and their extensions, timed Petrinets and Stochastic Process Algebra. Although QN models have been successfully used in the context of traditional time sharing computers they often fail to capture complex interactions among various hardware and software components in client/server distributed processing systems. Layered Queuing Network (LQN) was developed as an extension of this well-known Queuing Network model for handling such complex interactions [5], [6]. LQN Systems are particularly well suited to analyzing software performance because they model layered resources and logical resources in a natural way and they scale up well for large systems. An LQN model is an acyclic graph, with nodes (named tasks) that represent software entities and hardware devices, and arcs denote service requests. The LQN tasks are classified into three categories: pure clients, pure servers and active servers. Each server has an implicit message queue called the request queue where the incoming requests are waiting their turn to be served. A software or hardware server node can be either a single-server or a multi-server. A multi server is composed of more than one identical clones and work in parallel and share the same request queue. The tasks are represented by parallelograms and the processors by circles. LQN task can denote more than one kind of service, each modeled by a smaller parallelogram nested inside a task. An entry is like a port or an address of a particular service offered by a task. Each entry has its own execution time and demands for other services. An entry can be further decomposed into activities if more details are required to describe its execution. Arcs in LQN denote requests from one entry to other. Requests for service from one server to other can be made via three kinds of messages in LQN models: synchronous, asynchronous and forwarding.

The main difference between QN and LQN is that LQN can easily represent nested services. Also a server can become in turn a client to other servers from which it requires nested services, while serving its own clients. The word "layered" in the LQN name does not imply a strict layering of tasks (for example, tasks in a layer may call each other or skip over layers). The LQN model structure is generated from the high-level software architecture that shows the high-level architectural components and their relationships, and from deployment diagrams that indicates the allocation of software components to hardware devices. The LQN model parameters are obtained from annotated UML models of key performance scenarios.

IV. UML PERFORMANCE PROFILE

The Unified Modeling Language (UML) is the most widely used design notation for software at this time, unifying a number of popular approaches to specifying structure and behavior. To enable users to capture time and performance requirements and to evaluate those properties from early specifications , a language extension called the UML profile for Schedulability, Performance and Time (SPT) has been defined and adopted.
The "UML Profile for SPT" defines a general resource model, time modeling, general concurrency, schedulability and performance modeling. The SPT profile allows UML diagrams to be annotated with performance information. Particularly, the Performance Profile provides mechanisms for capturing performance requirements and for associating performance related QoS characteristics with the UML model. The Performance Profile facilitates the following

- capturing performance requirements within the design context,
- associating performance-related QoS characteristics with selected elements of the UML model,

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

- specifying execution parameters which can be used by modeling tools to compute predicted performance characteristics,
- presenting performance results computed by modeling tools or found by measurement

The performance profile describes a model that contains the basic values used in performance analysis, including resources used, key scenarios and user workloads. The performance profile maps classes of the domain model to stereotypes which are then applied to various UML elements. The class attributes are mapped to tagged values [7]. For instance the main stereotypes include <<PAclosedLoad>> to represent a closed workload. It has the following tags PArespTime, PApriority, PApopulation, PAextDelay.

Through these stereotypes and tags performance annotations can be attached to a UML model. For performance analysis the UML model should capture important system features such as high level software architecture, the allocation of software components to hardware resources and the key performance scenarios in the system .The importance of UML profile lies in the fact that it provides a standard way of attaching quantitative performance attributes to UML models. These attributes may represent values that are required, assumed, measured values or computed from a model. A set of rules has been defined for mapping a UML model annotated with performance information to a queuing-based performance model named Layered Queuing Network (LQN). The SPT profile allows UML diagrams to be annotated with performance information [8].

V. CASE STUDY: PERFORMANCE MODEL FOR DATA WAREHOUSE SYSTEMS

In this section we introduce a case study of Data warehouse system used to illustrate the generation of LQN models from annotated UML models. SPE approach can be used to predict the performance of a Data warehouse system before the system is actually implemented. We introduce the data warehouse architecture based on web which has four levels as given in [9]. The first is client, which provides users functions and convenient browsing of data stored in the data warehouse. At this level there is only a need to install the web browser connected to the internet; no need to install special client applications. Web Server is the second level, which is the interface between the client and the OLAP server and involves input and output of information between them. The third level is OLAP server which creates the data cube and builds multi dimensional models. The fourth level is that of a Data warehouse server. The OLAP structure based on Web is shown as below in Figure 1. The various OLAP operations that can be performed are rollup, drilldown, slice dice and pivot [10], [11]. However only one scenario namely roll-up is presented here.

The basic procedure for deriving an LQN model from UML model has the following steps.

a) The High-level software architecture of a software system is represented by one or more collaboration diagrams. These diagrams shows the concurrent or distributed components represented as active objects and the architectural patterns they participate in.

b) The components of high-level software architecture need to be allocated to hardware devices, represented as a deployment diagram.

c) A set of key performance scenarios annotated with performance information as per the UML Performance Profile. Each scenario can be represented as either as a sequence or as an activity diagram.

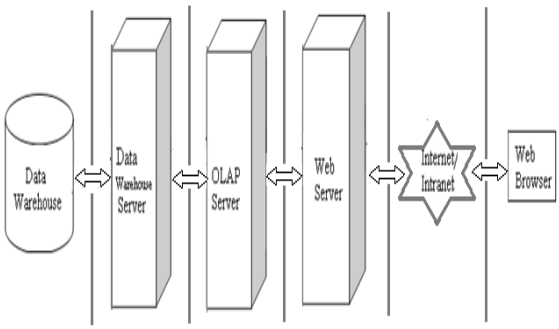

Figure 1.The OLAP structure based on Web

The output of the transformation algorithm is an LQN model that can be read and solved by the existing LQN solvers.

This section presents the algorithm for transformation of the UML to LQN notation level. The main steps of the algorithm are as follows:

Generate the LQN model structure

1.1. Identify the LQN software tasks from the high-level architecture

1.2. Identify the hardware devices from deployment diagram

2. Produce LQN details on entries, phases, activities from scenarios

2.1. For each scenario process the corresponding activity diagram

2.1.1. Match the communication pattern from the architectural pattern with the messages between components given in the activity diagram

2.1.2. Identify the activity diagram elements corresponding to different LQN entries, phases, and activities, and create the LQN elements

3. Traverse the LQN elements, compute their parameters and write out the model file.

In this section we present the UML models of a Data warehouse system used to generate the LQN model. Step 1 develops the LQN structure (i.e, the software and hardware tasks and their connecting arcs) from the high

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

level architecture of the UML model given in Figure 2 and from the deployment of software parts to hardware devices given in Figure 3.

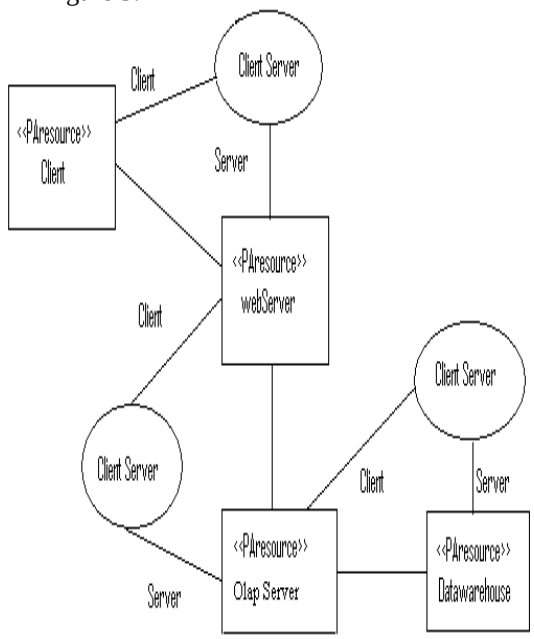

Figure 2. High-level architecture

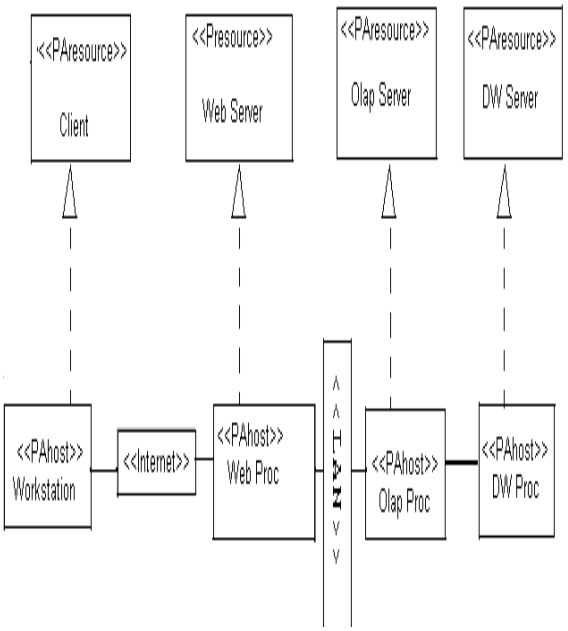

Figure 3. Deployment diagram

Figure 2 shows the high level architecture of a Data warehouse model as a collaboration diagram. The different processes are associated as shown, according to client server pattern. All the processes are treated as logical resources as indicated by the stereotype <<PAresource>> defined in the UML Performance Profile. Figure 3 shows a deployment diagram where the hardware components are linked according to the design blueprint. The stereotype <<PAhost>> is used to represent the processor nodes. The processes are running on processors as indicated by the deployment relationships. The network resources, both Internet and local LAN, are also shown, since they will have to be represented in the performance model. The Internet will be modeled as a "delay server" since the user data will suffer a delay, but it would hardly affect the contention level of the overall traffic carried in the Internet. On the other hand, the local LAN will be modeled as a finite server with queue, since the user traffic may change the congestion level of the local traffic.

The activity diagram in Figure 4 represents the only scenario considered in this example, "roll up query ". The activities performed by each concurrent process are represented inside a “swimlane”. Inter-process messages are indicated by transitions that cross the swimlane boundaries. (The type of the call, either synchronous or asynchronous, is denoted in the collaboration diagram and matches the interaction from the architectural pattern). The synchronization bar is used to represent fork or join calls between concurrent components. In the given scenario 1) Clients submit analysis request through the web browser 2) Web server receives the users analysis request and submits them to analysis server(OLAP server) 3) Analysis server calls the data from the data warehouse, finishes analysis operations and returns the results back to web server, which in turn forwards the results to the clients. 4) Client supports various OLAP operations in order to analyze the data. Annotations are added to activity diagram with performance information as per the UML Performance Profile [7],[12]. A scenario is composed of steps that can be shown in sequence, loops, branches, fork/joins, etc. In Figure 4, the stereotypes and the tagged values are given in notes attached to different activity diagram state. The scenario contains four synchronous calls representing client-server interactions (i.e., a request followed by a reply). The starting step of the scenario carries an additional note characterizing the workload: closed workload with population represented by the variable $N and an external delay (think time) of 15s. In step 1 we take into account only the structural aspect of the architectural patterns; Their behavioural are considered in step 2. Step 2.1 processes the activity diagram for each roll up scenario and generates the LQN elements; entries, phases and activities. Step 2.1.1 starts by identifying the messages between concurrent components (i.e messages crossing the swim lane boundaries). The intent is to overlay the behavioural aspect of the architectural pattern over the activity diagram in order to verify whether the scenario is consistent with the patterns. The activity diagram is then divided into sub graphs and further mapped to different LQN elements. The corresponding LQN elements (entries, phases, activities) are generated as nodes in step 2.1.2. The CPU demand of a LQN phase (activity) is obtained by summing up the CPU demands of all the states (i.e scenario steps) contained in the corresponding sub graph.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

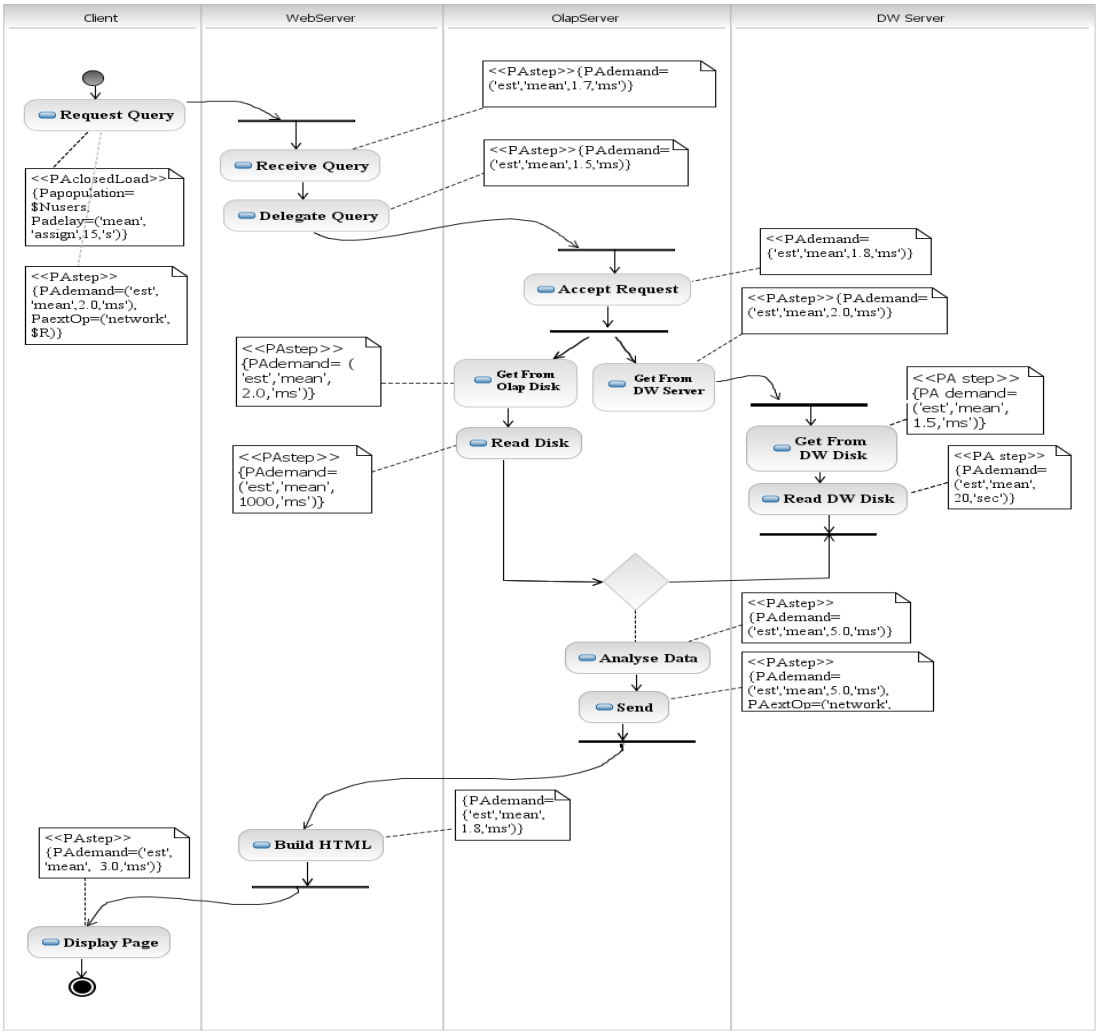

Figure 4.ActivityDiagram

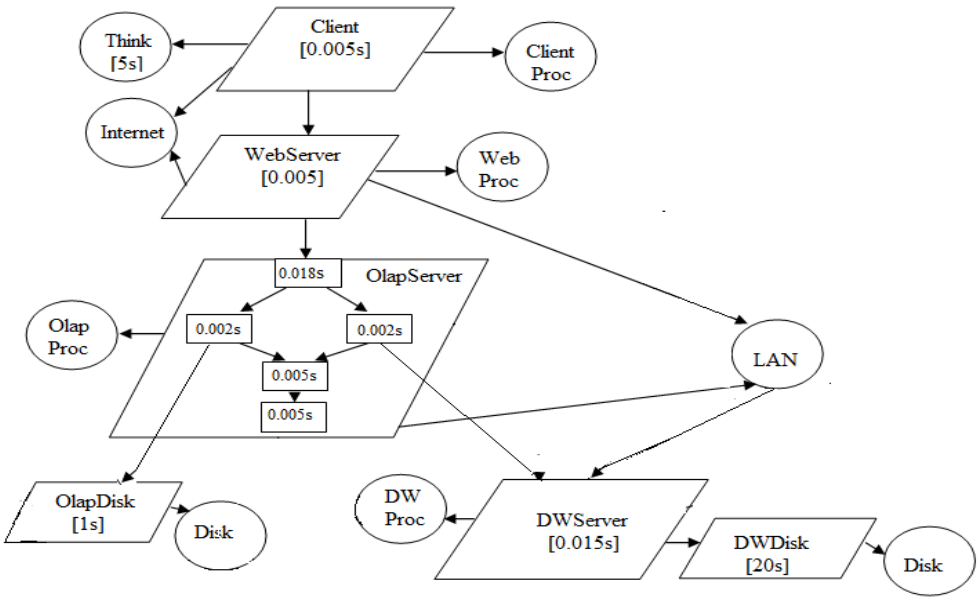

Figure 5. LQN Sub model generated for scenario “rollup”

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

The LQN model obtained by applying the above algorithm is shown in Figure 5. A LQN task was generated for each of the four software components from Figure 3.

An additional task is generated for network component. The OLAP server has many entries, one for each type of requests it accepts.

Only the roll up query is determined from this scenario. The rollup entry with internal branching is represented as a LQN activity graph that mirrors the scenario steps from the activity diagram of Figure 4. The purpose of this paper is to present the proposed UML to LQN transformation, so no performance analysis results are presented here.

VI. CONCLUSION

Our experience with the UML Performance Profile shows that it is relatively easy to understand, and that it provides enough performance annotations for generating working LQN models for a Data warehouse system.

This is the first step towards a methodology for performance evaluation of Data warehouse systems based on UML and Layered modeling.

This paper focuses on transformation process only and does not use performance model to improve the original system. Therefore our future work will include validation of the performance model against real measurements and identification of bottlenecks in the data warehouse design.

REFRENCES

[1]. S. Chaudhuri and U. Dayal. "An Overview of Data Warehousing and OLAP Technologies". ACM SIGMOD Record 26(1), Marc 1997.

[2]. Andrew Holdsworth. " Data Warehouse Performance Management Techniques". White paper , Oracle Services Ad vanced Technologies Data Warehouse,1997.

[3]. C. U. Smith, Performance Engineering of Software Systems, Reading, MA, Addison-Wesley, 1990

[4]. C.U.Smith, L.G.Williams, Performance Solutions: A practical guide to to creating responsive ,Scalable Software, Addison Wesley, 2001.

[5]. Gordon Gu, D.C. Petriu, "XSLT Transformation from UML Models to LQN Performance Models", Proc. of 3rd Int. Workshop on Software and Performance WOSP'2002, pp.227-234, Rome, Italy, 2002.

[6]. V.Cortellessa, R.Mirandola, "J. Huang "Deriving a Queuing Network based performance model from UML Diagrams ", in Proc of WOSP 2000,pp58-70,2000.

[7]. Object Management Group, " UML Profile for Schedulability, Performance, and Time Specification",OMG Adopted Specification ptc/2005.

[8]. Simonetta Balsamo, Moreno Marzolla "Performance evaluation of UML software architectures with mul ticlass Queueing Network models", WOSP 2005: 37-42

[9]. Chunhua Ju,Minghua Han,"Effectiveness of OLAP-based Sales Analysis in Retail Enterprises" ISECS International Colloquium on computing, Communication, Control and Management 2008.

[10]. E.F.Codd,S.B.Codd,C.T.Bally."Providing OLAP to user - Analysis":An IT Mandate, 1993. Codd and Date Inc.1993.

[11]. H .Hassan,P.Hyland,"Using OLAP and Multidimensional data for decision making": Faculty commerce papers , 2001, University of Wollongong. At http://ro.uow.edu.au/commpapers/4

[12]. Gordon Ping, Gu,Dorina .C.Petriu, "Early evaluation of software performance based on UML performance Profile" Proc. CASCON2003:pp 66-79