Performance Evaluation using Various Models in Distributed Component based Systems

Upinder Kaur
Computer Science Department
Chandigarh University, India

Sumit Sharma
Computer Science Department
Chandigarh University, India

ABSTRACT
Over the years, several models were proposed to analyze the performance of distributed component based system with the view of improving system's performance. Traditional methods of prediction such as Petri nets and queuing networks exploit the benefits of component engineering paradigm, such as division of work and reuse. We have surveyed different approaches of performance prediction which have attained widespread industrial use. Each approach has different goals and context, at which phase of life cycle process which approach is appropriate is described, there benefits and drawbacks are also given. Based on the analysis we have found that the models are machine centric e.g. throughput, responsiveness, no. of processes CPU has to execute within limited processor speed. None of the model draws parameters from the contributions of the client organization and end users. We have proposed a solution in this paper for taking into consideration end users perspective. Software is developed in order to satisfy requirements of the end users. Therefore, involving users in evaluating performance should not be underestimated.

General Terms
Distributed component based system

Keywords
CB-SPE, distributed systems, middleware, Palladio, PCM, performance, performance evaluation, prediction models

1. INTRODUCTION
Change is a badge of modern corporate. No corporate can do business without software. We are witnessing a change in technology. Earlier the development of software was done in traditional manner using different models (waterfall model, spiral model, prototype model etc).But, there are some pitfalls in traditional approach of software development such as architecture is monolithic. Nowadays, there is increase in demand of applications such as electronic commerce. Traditional methodologies have not achieved drastic gain in productivity and quality yet. Development time and cost is comparatively more in traditional approach of development. Concept of reuse seems to be difficult in traditional approaches. So, it becomes the dream of software engineering paradigm to remove all these pitfalls and introduce the concept of reuse .As we know that for building large and complex software, reuse of components is a smart means of development. So, CBSE was introduced in 1990's.

"The software components are binary units of independent production, acquisition, and deployment that interact to form a functioning system"[1].It is an established approach in many domains such as distributed systems, embedded systems, web based services and many other. The aim of CBSE is to achieve multiple quality objectives such as reusability, interoperability, implementation transparency. Component based software often consists of a set of self contained and loosely coupled components allowing plug and play. Component based software are developed using different programming languages and are implemented on different operating platforms. These are often produced either in-house or can be third party off-the-shelf components (COTS) in which source code is not available [2]. CBSE emphasizes modular architecture so that we can develop a system and incrementally enhance the functions by modifying the components. To make such design possible software architecture is required. One such example of software architecture for component based systems is CORBA (common object request broker architecture). Software architecture is provided in the form of frameworks. Frameworks are workable reference of underlying software architecture; they can be hierarchical up from domain independent to domain specific. Diagram below shows the development of component based systems, how components are integrated and deployed in the system.

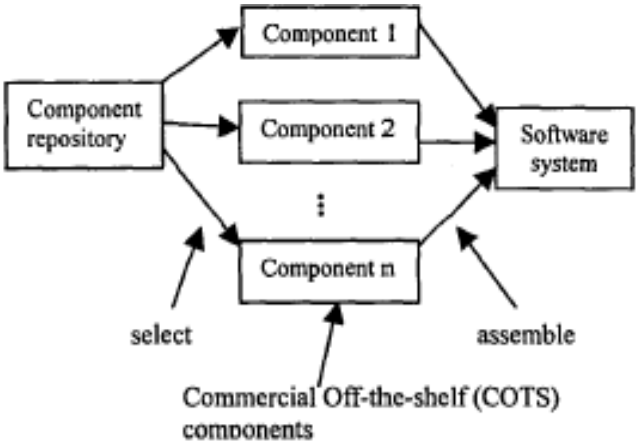

Figure 1

1.1 How CBSE differentiates from traditional approach of development?
CBSE differentiates from traditional approach of software development in following manner:

Table 1

Characteristics	Traditional approach	CBSE
Architecture	Monolithic	Modular
Methodology	Build from scratch	Composition
Process	Waterfall	Evolutionary
Development time	Takes more time	Takes less time
Reusability	Reusability is not there	It is reusable
Quality	Quality cannot be improved further	Better quality products

1.2 Distributed systems

Various commercial trends have led to an increasing demand of distributed systems. Today, Distributed component based systems are one of the complex artifacts that are used by organizations to deplore services simultaneously to many people online and real time. Firstly, no. of merges between companies is increasing rapidly. These merges let to an IT companies, which have to provide unified services to the demands of the customer. Secondly, time available for providing new services is very less. This can only be achieved from COTS and then integrate it into the required system rather than building it from scratch. Finally, it is used by people in real life applications such as electronic commerce, online payment, and e-commerce and lots others these services centralized servers and client server systems cannot provide.

Due to heterogeneity of component technologies and different network protocols used distributed applications are difficult to manage and develop. Quality of service (QOS) attributes such as reliability, security, performance, maintainability plays a critical role in real time distributed applications. Distributed systems should not only work properly in terms of functionality but also meet the requirements of the customers. In this paper we will focus on QOS attribute i.e. performance of distributed systems.

Performance of the system must be predicted at design level in order to avoid pitfalls of poor QOS during system implementation.

Software architecture (SA) describes how the components are interconnected, how they will communicate and interact with each other. This part is the major source of errors. Hence, performance evaluation at this level is useful for checking whether it fulfills clients requirements or not, potential risks and quality requirements.

We divide (SA) into two parts:-

(1)Macro-architecture-It covers external environment of the software systems. Examples are culture and belief of the people, government policies and regulations.

(2)Micro-architecture-It focuses on internal structure of the system like execution architecture, code architecture etc [4].

Performance of the software is a quality attribute which can be measured in terms of responsiveness, latency time, throughput, resource utilization, fault tolerance etc. Accessing the performance of the system is important for smooth and efficient operation of the system. There is no. of approaches used for predicting the performance of the software. We will discuss in next section.

2. PERFORMANCE EVALUATION METHODS

Performance prediction for component based software system helps software architects to evaluate their systems based on component performance specifications. Classical performance models such as queuing networks, stochastic Petri nets or stochastic process algebra can be used to analyze component based systems but these models exploit the benefits of component paradigm such as reuse and division of work. The challenge of the component performance model is that the performance of a software component depends on the context it is deployed into and its usage profile.

In this section, brief summary of performance models used in distributed component based system. This paper presents a survey and evaluation of the proposed approaches to help selecting an appropriate approach for a given survey.

This survey is more detailed and up-to-date as compared to the existing survey papers. The various approaches are categorized as you can see below:-

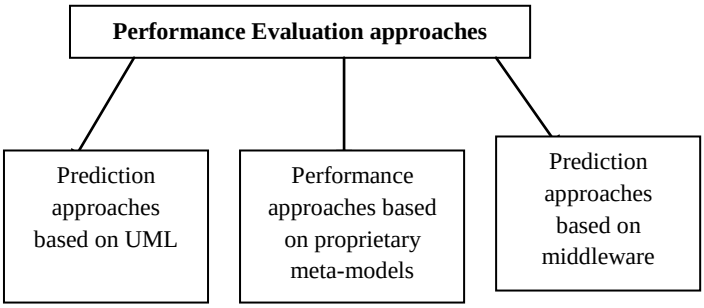

Figure 2

2.1 Prediction approaches based on UML

The Unified Modeling Language (UML) is a graphically-based object-oriented notation developed by the Object Management Group as a standard means of describing software designs which is gaining widespread acceptance in the software industry [5].UML based prediction approaches target on design phase prediction. It describes the component behavior by sequence diagrams, collaboration diagram and activity diagrams. Sequence or activity diagrams can be used to express those scenarios that have performance requirements. State chart diagrams describe the behavior of objects, and the time required to respond to stimuli. Deployment diagrams define how objects are mapped on to processing resources.[6] Component allocation is described by deployment diagram .UML only supports functional specifications, its extensions (profiles, constraints, tagged values) have been used by OMG(object management group) to allow modeling of performance parameters such as usage profile and workload. Earlier we use UML SPT profile which focuses on schedulability, performance and time. Presently, UML MARTE profile is in use. It adds capabilities to UML for model-driven development of Real Time and Embedded Systems (RTES). This extension, called the *UML profile for MARTE* (abbreviated as MARTE), provides support for specification, design, and verification/validation stages. This new profile is intended to replace the existing UML Profile

for Schedulability, Performance and Time. The benefits of using this profile are: Firstly, it Provides a common way of modeling both hardware and software aspects of a RTES in order to improve communication between developers. Secondly, it enables interoperability between development tools used for specification, design, verification, code generation, etc. Thirdly, it fosters the construction of models that may be used to make quantitative predictions regarding real-time and embedded features of systems taking into account both hardware and software characteristics[7].
Based on UML approach there are models. We describe the model which is mostly used nowadays in industry and has good tool support. There are many other models. But we are excluding those models from our survey papers which are outdated. Model for UML is CB-SPE

CB-SPE
The Component-Based Software Performance Engineering (CB-SPE) approach by Bertolino and Mirandola uses UML extended with the SPT profile as design model and queuing networks as analysis model.
CB-SPE adapts to CB framework and uses the standard RT-UML profile. The CB-SPE framework includes freely available modeling tools (Argo UML) and performance solvers (RAQS).The modeling approach is divided into a component layer and an application layer. In the component layer the component developer checks the performance of individual component in isolation. In application layer the system assembler predicts the performance of the integrated components on the actual platform. The results of the performance are analyzed by system assembler. Results can be contention based or best-worst case. If the results obtained are not meeting the desired result then the system assembler either changes the parameters to obtain the desired result or after a while, declare the infeasibility of the requirements. The CB-SPE framework includes freely available modeling tools (Argo UML) and performance solvers (RAQS)[8]. Future work includes the validation of the methodology by its application to case studies coming from industrial world.

2.2 Prediction approaches based on meta-models

The approaches in this group aim at design time performance prediction. Instead of using modeling language for prediction, these approaches use meta-models.

PCM (Palladio component model)
The PCM is a meta-model for the description of component based software architectures. The model is designed with a special focus on the prediction of QOS attributes, especially performance and reliability. In this paper, we focus on the performance related parts of the PCM. In the following, we give some details on our envisioned CBSE development process and the participating roles.
Four types of developer roles are involved in producing artifacts of a software system:

Business domain Experts, who are familiar with the customers or users of the system, provide different usage scenarios.
Parameters to be undertaken –Domain experts can specify user behavior with control flow constructs such as sequence, iterative and loop. Domain experts also specify workload. Workload can be open or closed. Open workload is that in which no. of users are not fixed and closed workload is that in which no. of users are fixed.

Software deployers model the resource environment and afterwards the allocation of components from the assembly model to different resources of the resource environment [9]
Parameters to be undertaken–System deployers allocate the resources to the components .Resources can be active and passive. Active resources are those which can execute request (hard disk, CPU etc) and passive resources are those which cannot execute request (semaphores, threads etc).

Component developers Component developers model the performance properties of component services with annotated control flow graphs, which include resource demands and required service calls (so-called resource demanding service effect specifications, RDSEFF).
Parameters to be undertaken-They can parameterize RDSEFFs for input and output parameter values as well as for the deployment platform .Resource demands can be specified using general distribution functions.

Software architects assemble components to build application.
Parameters to be undertaken-It takes care of control flow, resource demands.
The PCM reduces modeling complexity by providing different models for different CBSE developer roles.

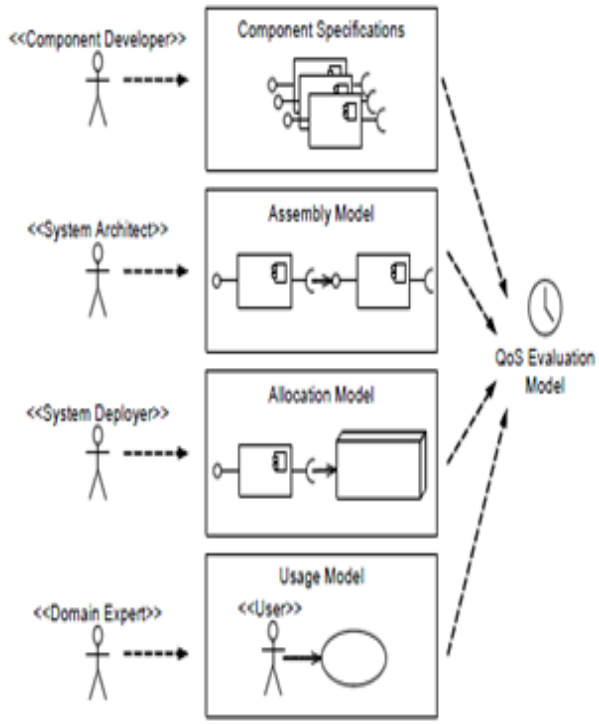

Figure 3

Diagram shows the individual performance roles [10].
Parametric dependency-The performance of software component is influenced by its usage. The resource demand may vary depending on input parameters (e.g., uploading larger files with a component service produces a higher demand on hard disk and network).Different required services can be called as a result of different inputs [10].

Limitations of PCM
There are certain limitations in PCM we will describe it briefly:
Static architecture: This means neither the connectors change nor the components move like agents.

Abstraction from the state: It is assumed that no run time environments or internal state of components determine the behavior of the system.
Limited support for concurrency: Quality-of-service properties in concurrent system are hard to predict.
Mathematical assumptions: To reduce the complexity of the model, mathematical assumptions are required.

Tool support
An Eclipse-based open-source tool called "PCM-Bench" is implemented, which enables software developers to create instances of the PCM metamodel and run performance analyses. The tool offers a different view perspective for each of the four developer roles and provides graphical model editors. Models of the different developer roles reference each other in the editor, which enables the creation of a full PCM instance. The PCM-Bench is an Eclipse RCP application and its editors have been partially generated from the PCM Ecore metamodel with support of the Graphical Modeling Framework (GMF)[11].Industrial Case study of PCM is present[9].

2.3 Performance approaches based on Middleware

Some distributed systems are built with middleware technologies such as J2EE (Java 2 enterprise edition) or CORBA (common object request broker architecture).These provide services and facilities whose implementations are available when architectures are defined. Middle tier software that provides facilities and services to simplify distributed assembly of components, e.g., communication, synchronization, threading and load balancing facilities and transaction and security management services The same middleware behaves differently in different context of applications. Medvidovic, Dashofy and Taylor state the idea of coupling the modeling power of software architectures with the implementation support provided by middleware. They notice that "architectures and middleware address similar problems, that is large-scale component-based development, but at different stages of the development life cycle."[11]

J2EE Test Suite Design

Central to the J2EE specification is the Enterprise JavaBeans (EJB) framework. EJBs are server-side components, written in Java, that typically execute the application business logic in an N-tier application. An EJB container is required to execute EJB components. The container provides EJBs with a set of ready to use services including security, transactions and object persistence. Importantly, EJBs call on these services declaratively by specifying the level of service they require in an associated XML file known as a deployment descriptor. This means that EJBs do not need to contain explicit code to handle infrastructure issues such as transactions and security.
An EJB container also provides internal mechanisms for managing the concurrent execution of multiple EJBs in an efficient manner. EJBs themselves are not allowed to explicitly manage concurrency, and hence must rely on the container for efficient threading and resource usage, including memory and thread usage for application components (EJBs) and database connections.

The Foresight Approach
The performance prediction methodology has three aims.
The first is to create a COTS product-specific performance profile that describes how the various components of the middleware product affect performance. This profile is aimed at analyzing the behavior and performance of a middleware product in a generic manner that is not related to any particular application requirements. Using this profile, it should be possible to use a set of generic mathematical models to predict the behavior of the middleware infrastructure under various configurations. The second aim is to construct a reasoning framework for understanding architectural trade-offs and their relationships to specific technology features. This reasoning framework provides the architect with insights into how the different quality attributes of the application interact with each other. It aims to help the architect reason about the effects of their architectural decisions and the effects of these on application performance and scalability. The third and final aim is to create an application-specific configuration that takes in to account the behavioral characteristics of the application at hand. The application architect describes the application behavior in terms of client loads, business logic complexity, transaction mix, database requirements, and so on. By inputting these parameters in to the generic performance models, it is possible to predict the application configuration settings required to achieve high performance.

Performance parameters [11]

Table 2

Workload	No. of clients Client request frequency Client request arrival rate Duration of the test
Physical resources	Number and speed of CPU(s) Speed of disks Network bandwidth
Middleware configuration	Thread pool size Database contention pool size Application component cache size Message queue buffer size
Application specific	Interactions with the middleware -Use of transaction management -Use of security service -Component replication -Component migration Interaction among components -remote method calls -asynchronous message deliveries

Middleware selection- The possibility of evaluating and selecting the best middleware for the performance of a specific application is important. Based on the abstract

architecture designs, it allows measuring and comparing the performance of a specific application for different middleware and middleware technologies.

NITCA-This performance prediction model targets server side component technologies such as EJB, .NET and CORBA. It doesn't distinguish roles between component developer and system architect and assumes that middleware has higher impact than single components. To determine the parameters of QN model authors analyze different architectural patterns for EJB applications. The activities in these diagrams refer to generic container services. The activity diagrams are further augmented with use case specific information such as the number of times a certain pattern is executed and the frequency of each transaction type. The resulting model contains placeholders for services provided by the component container, which are used as platform independent resource demands.

Tool - There are plans to build a capacity planning tool suite called Revel8or based on the approach.

Case study-Industrial case study is there on stock Online test application based on two middleware technologies i.e. CORBA and EJB.

3. COMPARITIVE ANALYSIS

Name	**CB-SPE**	**PCM**	**NICTA**
Approach	Based on performance approach of UML	Based on performance approach of meta-models	Based on performance of middleware
Lifecycle phase	Design phase	Design phase	Early phases of lifecycle
Case study	Software retrieval system	Web audio store	Online stock test application
Tool support	Modeling analysis (CB-SPE) tool support	PCM-bench	Revel8or
Division of work	It doesn't divide the roles.	Easy to implement as it divides the roles	It doesn't divide the roles
Applicability in industry	Medium(Case studies exist-but tools are outdated)	High(case studies exist and tools are also up-to date)	Medium(Case study exist but tools not prepared yet)
Behavior model	UML -SPT	PCM	EJB 2.0 + UML activity diagrams
Performance model	Execution graph + QN	EQN,LQN	QN(Queuing networks)
User centric/machine centric	Machine centric	Machine centric	Machine centric

Figure 4

4. CRITICAL REFLECTIONS

4.1 Prediction approaches based on UML

The main benefit of UML based approaches is their compliance to OMG standards. Hiding the complexity of performance models by using model transformations and results feedback could increase the acceptance of performance analysis techniques. Furthermore, UML based techniques enables developers to reuse existing UML models, thereby lowering the cost and effort.

There are several drawbacks that have limited UML based approaches for performance prediction. The concept of reuse is only introduced in UML 2.0 and is still subject to discussion. Component developer and software architect have to work dependent on each other using the same UML. Existing UML performance profile doesn't include parameterized component specifications for different usage profile and scenarios. Tools have not much supported. Many companies use UML only as a documentation tool.

4.2 Prediction approaches based on meta-model

This approach is based on their own meta-models and doesn't rely on UML.The benefit of these approaches is good tool support with graphical editors, model transformations to known models. Meta-models allow researchers to create new level of abstraction for component system which could be more accurate than UML.These models are stricter than UML. The modeling languages used are easier to learn for developers.

As drawbacks, these methods are not standard conforming like UML so they have to undertake several challenges. Developers first have to learn a new language then re-formulate the existing UML models into proprietary language, which might now be a straight forward approach due to different levels of abstraction. Moreover, they support only specific UML version.

4.3 Prediction approaches based on Middleware

Benefits of these approaches are high accuracy and easy applicability in industry. These approaches have a high influence where there is focus on middleware implementations.

Drawbacks of these approaches are there running, setup cost is high. Portability is low as it is restricted for specific middleware version. It becomes easily outdated if a new version appears. Tool support for these approaches is limited.

5. PROPOSED SOLUTION

A solution is proposed for user centric approach in evaluating the performance of the system. In this approach our main focus is on the level of experience of individual resource (human resource) which is usually underestimated in machine centric approach. The performance of the system doesn't contribute individual resource in achieving the required output. I will make you understand by depicting an example. Let's suppose there are 10 employees working in a cycle factory their output for one day is 50 productions of cycles. We are assuming equal contribution and experience of all employees in achieving the required output, which will always not be possible. Our approach is to assign weights to each resource according to the level of experience. A person (human resource) having more experience is assigned large no. of weights and vice versa. In this way level of experience is not ignored in evaluating the performance of the system and hence performance of users is taken into consideration.

6. CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

In this paper latest performance approaches are reviewed, there underlying principles of design and implementation and there benefits and limitations. We were able to establish that the parameters of performance models in this period were machine centered/driven and we propose that future models should have as input parameters, based on organization variables." How to measure the performance from user perspective" is hoped that this question will be addressed for future work.

6.2 Future scope

This survey has revealed many open issues and recommendations for future work in performance evaluation of distributed component based systems:-

Hybrid solution techniques: Hybrid solution combines numerical analysis method with simulation approaches. None of the method has used hybrid approach. Hybrid approaches could be the solution for efficient and accurate performance analysis.

User-centric model: User centric model should be taken into consideration. As we have seen above all the models are machine centric, take input parameters as CPU utilization, resource demand, throughput etc. These parameters don't satisfy requirements of end users. Organization decision variables (organization goals and tasks, level of users experience in IT, information requirements of users and format) should not be underestimated. Question-Answer set should be designed for performance evaluation. For each scenario, question set is given and users have to answer it. Based on this evaluation can be done in future. Designing of user specific performance models or hybrid model which takes into consideration both user variables and machine variables is foreseen in future.

7. REFERENCES

[1] Aoyama, M. (1998, April). New age of software development: How component-based software engineering changes the way of software development. In 1998 International Workshop on CBSE.

[2] Brown, A. W., & Wallnau, K. C. (1998). The current state of CBSE. IEEE software, 15(5), 37-46.

[3] Akinnuwesi, B. A., Uzoka, F. M. E., Olabiyisi, S. O., & Omidiora, E. O. (2012). A framework for user-centric model for evaluating the performance of distributed software system architecture. Expert Systems with Applications, 39(10), 9323-9339.

[4] Olabiyisi, S. O., Omidiora, E. O., Uzoka, F. M. E., Akinnuwesi, B. A., Mbarika, V. W., & Kourouma, M. K. (2011). Exploratory Study of Performance Evaluation Models for Distributed Software Architecture. Journal of Computer Resource Management (International Journal of Computer Measurement Group Inc), Autumn, 130, 47-57.

[5] Gomaa, H., & Menascé, D. A. (2001). Performance engineering of component-based distributed software systems. In Performance Engineering (pp. 40-55). Springer Berlin Heidelberg.

[6] Bennett, A. J., & Field, A. J. (2004, October). Performance engineering with the UML profile for schedulability, performance and time: a case study. In Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004.(MASCOTS 2004). Proceedings. The IEEE Computer Society's 12th Annual International Symposium on (pp. 67-75). IEEE.

[7] Object Management Group (OMG). UML Profile for MARTE, Beta 1. http://www.omg.org/cgi-bin/doc?ptc/2007-08-04, August 2007. last retrieved 2008-01-13.

[8] Bertolino, A., & Mirandola, R. (2004). CB-SPE Tool: Putting component-based performance engineering into practice. In Component-Based Software Engineering (pp. 233-248). Springer Berlin Heidelberg.

[9] Becker, S., Koziolek, H., & Reussner, R. (2009). The Palladio component model for model-driven performance prediction. Journal of Systems and Software, 82(1), 3-22.

[10] Becker, S., Koziolek, H., & Reussner, R. (2007, February). Model-based performance prediction with the Palladio component model. In Proceedings of the 6th international workshop on Software and performance (pp. 54-65). ACM.

[11] Denary, G., Polini, A., & Emmerich, W. (2004, January). Early performance testing of distributed software applications. In ACM SIGSOFT Software Engineering Notes (Vol. 29, No. 1, pp. 94-103). ACM.

[12] Koziolek, H., Becker, S., Happe, J., & Reussner, R. (2008). Evaluating performance of software architecture models with the Palladio component model. Model-Driven Software Development: Integrating Quality Assurance, IDEA Group Inc, 95-118.