Proc. of the Intl. Conf. on Advances In Engineering And Technology - ICAET-2014
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-1-63248-028-6 doi: 10.15224/ 978-1-63248-028-6-01-01

Performance Prediction Approaches for Component-based Systems

Monika Kalotra, Kuljit Kaur

Abstract: **Performance predictions of component assemblies and obtaining performance properties from these predictions are a crucial success factor for component based systems. The number of methods and tools has been developed that analyze the performance of software systems. These methods and tools aim at helping software engineers by providing them with the capability to understand design trade-offs and optimize their design by identifying performance or predict a systems performance within a specified deployment environment. In this paper, we establish a basis to select an appropriate prediction method and to provide recommendations for future research, which could improve the performance prediction of component-based systems.**

Keywords: **Component-based system, quantitative approach, Performance, Prediction.**

I. INTRODUCTION

Component-Based System is an approach to build applications from deployed components. Developing software applications using CBS has many advantages like the efficiency, reliability. Performance is an important factor that must be considered [1]. Performance is referring to how extend the component has satisfied the predefined requirements of specific factors. The failure of performance means increased expenses of hardware and software development. So, the best solution is to avoid the late evaluation of performance. The German police has been developed a system called "Impol-Neu”, that proves the importance of performance evaluation before deployment, which was published in mass media [2]. After development the performance of this system was evaluated. So, the resulted performance did not satisfied performance or user requirements. For that reason, they failed to implement the system in spring 2001 as it planned; instead the system was implemented in 18th August 2003. Consequently, performance is a key success factor in software production. To develop predictable software system, the performance should be addressed early at development stage with minimal performance problem otherwise it will impact the cost, schedule, and quality of the software [15]. Therefore, this paper presents a survey of the proposed approaches to help selecting an appropriate approach for a given software system.

Monika Kalotra, Guru Nanak Dev University

Amritsar, India

Kuljit Kaur, Assistant Professor, GNDU,

Amritsar, Punjab

A. Factors Influencing Component Performance

a) Component implementation*:* Component developers can implement the functionality in different ways that specified by an interface.
b) Required services: The total execution time of a component service depends on the execution time of required services.
c) Deployment platform: Different software architects deploy a software component to different platforms.
d) Usage profile: Clients can invoke component services with different input parameters. Depending on the values of the input parameters the execution time of a service can change.
e) Resource contention: A software component on a given platform does not execute as a single process in isolation. The induced waiting times for accessing limited resources add up to the execution time of a software component.

III. MAIN APPROACHES TO PERFORMANCE PREDICTION

The main objective of software performance prediction is to improve the performance of software. We are using the quantitative approach in this paper. Three types of approaches are resulted from the proposed study are:

A. Measurement Approach

Measurement approach refers to a software performance engineering aims to evaluate software

Proc. of the Intl. Conf. on Advances In Engineering And Technology - ICAET-2014
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-1-63248-028-6 doi: 10.15224/ 978-1-63248-028-6-01-01

application focusing on the quality attributes of performance such as response time and throughput. These features are examined using special analysis tools which enable the monitoring of execution. The approach could be efficiently used for implemented application or application with known features. The approach uses existing systems to measure performance properties and adjust performance models with the results. Performance analysts may use the models to analyze the results of changed workloads or the use of faster hardware with low effort which uses test-cases to adjust measurement from reused components [5].

In [4] a discussion is given about how the properties of component-based system may influence the selection of methods and tools used to obtain and analyze performance measures. Then, a method is proposed for the measurement of performance distinguishing between application-specific metrics and platform-specific metrics. The automation of the process of gathering and analyzing data for these performance metrics is also discussed.

Recently Jiang [14] has proposed measurement approaches based on testing validation to ensure the quality of system that composed from black-box components. The approach uses the previous testing information of reused component to help in reducing the effort of testing.

The drawbacks of this approach, is only suitable for already implemented systems and there is need to introduce the application, to enable the analysis of changed workloads.

B. Model Based Approach

Generally, model-based approaches depend on the Model Driven Development (MDD) technique which enables developer to efficiently evaluate and assess the system requirements and execution by using a set of models. In addition having good input models and accurate analysis models, performance prediction for component-based systems adds an additional level of complexity by the introduction of the development roles. The information needed for conducting a performance evaluation is spread among the developer roles [8]. The component developer knows for example how the component is realized while the software architect knows how to assemble the components of the system. The influencing factors are also considered while dealing with performance of component based system
Becker et al. (2006) surveyed existing Component based performance prediction methods including a discussion on the support for parameterized component performance models.

I. RESOLVE-P: Sitaraman et al. (2001) take the usage of the components into their predictions by using an extended Big-O Notations to specify the time and memory consumption of software components that depending on the input parameters passed to service calls. The composing services are supported on an abstract level by composing the specified Big-O demands [17]. RESOLVE specifies the functionality of a component with a list of service signatures and also pre and post condition for each service. According to authors they point out the limitations of classical big-O notations for generic software components with polymorphic data types. Therefore, they increase the functional RESOLVE specifications with an adapted big-O notation for execution time and memory consumption. The specification does not distinguish between different processing resource (e.g., CPU and hard disk), does not include calls to required services, and passive resources. This approach targets a fundamental theory of performance specification and do not deal with prediction or measurement frameworks.

II. PACT: (Hissam et al. (2002)) gave a conceptual framework so called Predictable Assembly or Prediction Enabled Component Technology. The assembly consists of certified components whose properties are combined according to a composition theory. The framework takes component properties and their assembly into account [3]. It is only a conceptual framework it depends on the actual method used. CCL (Component constructive Language) is used for architectural description. It supports synchronous and asynchronous communication with required services. CCL allows by specifying component behaviour with state charts. Resource demands are attached to the CCL components using annotations. CCL supports composite components but not memory consumption. For analysis, tools transform CCL models into intermediate constructive model, which focuses on the relevant part for performance analysis and helps in the implementation of further transformations. PECT mainly targets analyzing real-time properties of component-based embedded systems.

III. CB-SPE: (Smith and Williams 2002) Software Performance Engineering is a method that focuses on software performance early in the software development life cycle. To evaluate designs it uses quantitative methods. SPE also provides patterns, models, and advices to help performance engineering. SPE is used throughout the life cycle phases, to predict and to manage software performance [6]. The CB-SPE approach by Bertolino and Mirandola uses UML extended with the SPT profile as design model and queuing networks as analysis model. The modelling approach is divided into a component layer and an application layer [14]. On the component layer, developers model the schedulable resource demands of individual performance services in dependence to environment parameters. In the application layer, software architects pre-select components performance models and compose them into architecture models.

Bertolino and Mirandola (2004) apply the SPE method to component-based systems by separating component performance models and assembly models. With this, the external service calls and the execution environment become parameterized. However, the software architect has to specify a performance critical scenario to the SPE method. As he should not posses' information on the component internals, this is a drawback of the method. Further, this method does not take input parameters into account.

IV. CBML: Wu and Woodside (2004) to build the parameterized component models use LQN models of components. LQNs model the behaviour and resource demands of software entities with so-called `tasks'. Resource demands are specified as mean values of exponential distribution functions, but there is no support for memory consumption. For each component an LQN model specifying it's provided and required interfaces as well as the control flow and resource usage dependencies. These single component LQN models are combined according to an assembly model into a system LQN model which gets evaluated [13]. Wu and Woodside (2004) also consider inserting components which they call completions (Woodside et al., 2002) for environmental services like middleware services into the system model automatically to increase the prediction accuracy of the environmental influence.

To define CBML components there is an XML schema, but there are no graphical editors for CBML models.

V. CB-APPEAR: Eskenazi et al. (2004) present a method for the performance prediction of existing components which undergo evolution. A parametric performance model is derived for these components by putting them into a test bed which figures the dependencies between method invocations and invocations of environmental services out. Depending on the complexity of the parametric dependency, the resulting model is either analytical or simulation based. However, the approach makes strong assumptions which are necessary to derive the performance models by testing [7].

VI. Hamlet: Hamlet et al. (2004) this approach comes from the area of software testing. It was first proposed for reliability prediction of component-based systems, and later extended for performance prediction. The authors use a restricted component definition to reduce the complexity of software composition. A software component is a mathematical function with input single integer value. Component composition of two software components means that the first component sends all its outputs to the second component in a pipe-and-filter style. This method requires component specification by component developers and performance prediction by software architects. This approach executes the components and measure how the component usage requests moves in orders to gain accurate performance predictions. However, their component model is limited as in their model components are simple functional transformations having only a single service. The approach does not consider concurrency or scheduling [13]. However, it considers the influence of the internal state of components to performance in a restricted form. This approach assumes a very restricted class of components, no validation on an industrial system.

VII. ROBOCOP: Bondarev et al. (2005) introduce a prediction method for embedded systems designed using ROBOCOP. This method can deal with implementation details specified by the component developer parameterized by external services, the component's hardware environment, and usage. However, due to its focus on embedded systems, the support for parameterizations is limited. ROBOCOP components contain a resource specification and a behavioral specification as well as the executable implementation of the component. Component developers specify ROBOCOP components in a parameterized way. Software architects compose these specifications and the parameters instantiate by the component developers. Scheduled resource demands of software components are constants in ROBOCOP [13], which also allows limited resource demands for semaphores and memory. The so-called Real-Time Integration Environment, which is implemented as a number of Eclipse plugins, supports the whole design and performance prediction process with ROBOCOP. The Support for software layers like operating systems or middleware platforms is outside the scope of this work and the Composite components are also not supported.

VIII. KLAPER: Grassi et al (2007) this method for performance prediction of component-based software systems is so-called kernel modelling language called KLAPER. The language is implemented in MOF (Meta Object Facility) and aims at helping the implementation of model transformations from different kinds of component system models (e.g., UML) into different kind of performance models (e.g., qeueuing networks). With KLAPER, it shall be possible to combine component performance models by different component developers based on different notations in a single prediction. There is no support for composite components. The language includes scheduled and limited resource demands as well as control flow. The authors implemented QVT transformations from KLAPER to EQNs and Markov chains. Because this is only be used by model transformations, no graphical modelling tools. The language is able to model dynamic software architectures where the connectors between components can change at runtime [11].

IX. Palladio Component Model (PCM): (Becker et al., 2009) PCM is a domain specific modeling language to describe component-based software architectures. Its major aim is to enable performance predictions for software architectures at design time. PCM is a dedicated component-based software development process, which distinguishes between the four roles of developers, architects, deployers, and domain experts. Each role has a limited view on the entire system model and contributes within its responsibility only specific parts to this holistic model. Becker et al. (2007)[15] enhance the model further by introducing a new SEFF concept called Resource Demanding SEFF (RD-SEFF) considering parametric dependencies to input Parameter and the execution environment. For this, an extension to the PCM's meta-model introduced so called stochastic expressions. Component developers can use them for example to specify resource demands depending on input parameters. Thus, it introduces a model-based simulation tool for predictions. Based on this, Koziolek et al. have added additional concepts to specify return value abstractions for external calls and component configuration parameters. Additionally, the authors introduce a model-driven approach to derive an analytical performance prediction model using model-2-model transformations [13]. The PCM-Bench tool allows independent graphical modelling for all four developer roles. Further model transformations map the whole model into performance models, which are solved by simulation or numerical analysis.

C. Mixed Approach

Mixed approaches are based on the group of measurement and model-based approaches. The mixed approach assumed to benefit from the strengths and avoids the weaknesses of the two approaches. These approaches mainly focusing on component specification that support the runtime performance information on software components and software application execution environment.
In [9] a methodology is presented, which aims for predicting the performance of component in distributed systems both during development. The methodology combines monitoring, modelling and performance prediction. UML models are created dynamically with non-intrusive methods based on performance prediction models. The application performance is then predicted by generating workloads and simulating the performance models.

In [10] an approach to predict the performance of component-based applications during the design phase is presented. The proposed methodology derives a quantitative performance model for a given application from the mentioned component platform. The results obtained for an EJB application are validated with measurements of different implementations. Using this methodology, it is possible for the software architect to make early decisions between application architectures in terms of their performance and scalability.

IV. PERFORMANCE MODELS

A performance model is an abstract representation of a real system that captures its performance properties, which are related to the quantitative use of resources during runtime behaviour and it is capable of reproducing its performance. The model can be used to study the performance of different designs. The evaluation of the performance model is done by analytical methods or by simulating the model.

A. Layered Queuing Network Modeling

In queuing networks, queues and their service represent processing resources which process work jobs queuing for service. Jobs travel through a network of service centers using probabilistic routes. The result of a qeueuing network analysis gives the average response time of the overall system, waiting times for queues, average queue length, and server utilization. For this, the class of queuing networks

Proc. of the Intl. Conf. on Advances In Engineering And Technology - ICAET-2014
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-1-63248-028-6 doi: 10.15224/ 978-1-63248-028-6-01-01

exists which can be solved by iterative methods like the Mean-Value-Analysis. The complexity of a queuing network depends on the characteristics of the service centre's and the assumptions on the jobs. For networks where arrival rates and service times are generally distributed, no analytical solution is known. The only known methods apply simulation techniques [6].

B. Stochastic Process Algebras

Based on process algebras developed by Milner, extensions for performance prediction exist which introduce stochastic time demands for the actions of the algebra. The advantage of using process algebra is the possibility to specify the behavior of the processes. Compared to queuing networks where the routes of the jobs in the network are usually probabilistic, the processes of process algebra behave according to the semantics of the algebra. This also allows formal analysis of additional system properties like deadlock freedom [6]. For an analysis, the process specifications are transformed into Markov chains exploiting the memory less property of the exponential distribution. Models based on general distributions cannot be solved analytically resulting in a need for a simulation based evaluation tool.

C. Stochastic Petri-Nets

Enhancements exist for Petri-nets as introduced by Petri which enable performance predictions based on Petri-net models. A Petri-net consists of a set of places and transitions, which are traversed by tokens. Transitions remove and add tokens on places whenever they fire [15, 6]. Transitions are active whenever more tokens are on all places affected by the transition as required by the transition's specification. Among all active transitions one is selected to fire resulting in the final change of the Petri-net's state. Stochastic enhancements add exponential distributed activation times to transitions which specify a minimum time which has to pass at least for the transition to fire again. Additionally, probabilistic routing of the tokens can be specified. Stochastic Petri nets have exact mathematical definition of their execution semantics. It's very hard to understand these mathematical equations so that's why the scope is limited.

TABLE 1

Comparisons of Performance Prediction Approaches

Name	RESOLVE	HAMLET	CB-SPE	CBML	PECT	ROBOCOP	KLAPER	PALLAIDO
Target Domain	Generic	Generic	Distributed systems	Distributed Systems	Embedded systems	Embedded systems	Distributed systems	Distributed systems
Component design model	RESOLVE	Simple mathematical functions	Ant.UML diagrams (UMLSPT)	CBML	CCL	ROBOCOP	UML-SPT, KLAPER	PCM
Model transformation	n/a	n/a	n/a	ad hoc (C-code, XMI I/O)	ad hoc	QVT	ad hoc (RTIE tool)	ad hoc(java)
Performance model	n/a	Simple algebra	n/a	Execution Graph+ QN	LQN	QN, RTQT, MAST	Markov chain, EQN	EQN, LQN
Performance model solver	n/a	Analytical	n/a	Analytical	Analytical, Simulation	simulation	Simulation	Analytical+ simulation (LQN), simulation (EQN)
Prediction Feedback into Design model	n/a	n/a	n/a	n/a	n/a	Areas of interests visualizes	Planned	Planned
Tool support	n/a	Modelling, Analysis (CAD tool)	Modeling, analysis (CB-SPE Tool Suite)	Modeling analysis (LQNS), Simulation LQSIM	Modeling Simulation (PACC Starter KIT)	Modeling, simulation (RTIE)	Model transformation (KLAPER QVT)	Modeling, analysis, Simulation-PCMBench

V. CONCLUSION

In this paper, we have reviewed performance prediction approaches for component based software systems. The area of performance for component-based software engineering has significantly matured over the last decade. By predicting the performance of component based system, the throughput, resource utilization and the response time of the component running on them could also be enhanced to a great extent. Also, there is a great scope of research in this field for performance in component based software system is one of most important issues related with this field.

REFERENCES

[1] IEEE, Standard Glossary of software engineering tech. 1991.
[2] Chessman, J.,UML Components: A Simple Process for Specifying Component-based Software. Addison-Wesley,2000.
[3] Hissam, S.A., Packaging Predictable Assembly, ACM Conference, Berlin, Germany, June 2002, Springer.
[4] Chastek, and S. Yacoub, Performance Analysis of Component-Based Applications, in Springer 2002, Berlin.
[5] Lloyd G. Williams, Connie U. Smith, Making the business case for software performance engineering, in: CMG, 2003.
[6]. Balsamo, S., Marzolla, M.: A Simulation-Based Approach to Software Performance Modeling. In: ESEC/FSE-11,2003,ACM
[7]. Eskenazi, A.V., Obbink, H., Pronk, B.: Analysis and Prediction of Performance for Evolving Arch. IEEE, 2004, France.
[8]. Eskenazi, Hammer, D.K., Obbink, H., Pronk, B.: Analysis and Prediction of Performance for Evolving Architectures. In IEEE, ed.: Proceedings of the 30th EUROMICRO Conference 2004.
[9]. Diaconescu, A., Murphy, J.: Automatic Performance Management in Component Based Software Systems. In IEEE, 1st International Conference on Autonomic Computing May 2004.
[10]. Liu, Y., Fekete, Predicting the Performance of Middleware-Based Applications at the Design Level. In: Proceedings of the 4th International Workshop on Software and Performance, 2004.
[11] Vicenza Grassi, "A kernel language for performance analysis of CBS", 5th International conf on Software, ACM 2005.

[12] Steffen Becker, Heiko Koziolek, "Model-based Performance Prediction with the Palladio Component Model," 2007, ACM.
[13] Koziolek, H., Performance evaluation of component-based systems: A survey. Performance Evaluation,2009.
[14] Jiang, Y., et al. Extended software component model for testing and reuse. 2010. China: IEEE Computer Society.
[15] Cortellessa, V., nverardi, P. Model-Based Software Performance Analysis. Springer, 2011.

I, Monika Kalotra doing M.tech from Guru Nanak Dev University Amritsar. Punjab.