Predicting Performance of Software Systems during Feasibility Study of Software Project Management

D Evangelin Geetha, T V Suresh Kumar, K Rajani Kanth
M S Ramaiah Institute of Technology
Bangalore – 560054, India
degeetha@msrit.edu, tvsureshkumar@msrit.edu, rajanikanth@msrit.edu

Abstract— Software performance is an important non-functional attribute of software systems for producing quality software. Performance issues must be considered throughout software project development. Predicting performance early in the life cycle is addressed by many methodologies, but the data collected during feasibility study not considered for predicting performance. In this paper, we consider the data collected (technical and environmental factors) during feasibility study of software project management to predict performance. We derive an algorithm to predict the performance metrics and simulate the results using a case study on banking application.

***Keywords*—Software Performance Engineering, Feasibility Study, Use case point**

I. INTRODUCTION

Performance is an important issue for a software system. Even several development methodologies for software system have been evolved, performance issues addressing are often discussed separately from Software Development Life Cycle (SDLC). There are several approaches for performance considerations in hardware industries, system software and networking environment and application programming.

For software developers, if performance evaluation is intrinsic throughout SDLC, it is easier for them to produce quality software. Researchers captivate developers, methodologies to produce quality software systems and have done some work to circumvent difficulties in making performance tools. Several industries canopy their performance issues and addressing performance issues after coding phase. Tuning during coding phase has severe consequences for performance intensive software systems. It gives impetus thinking in looking into issues while developing software systems. The plausible solutions may be easy for team members to construct quality software systems. Current situation in building performance intensive software system is cumbersome. This led to several researchers to work in this important performance engineering area. Basic idea of works including ours is to make team members to understand easily the performance issues and consequences. Few authors addressed the performance issues early in SDLC [7], [8], [9], [10].

In general, performance evaluation system must address performance questions from varied stakeholders. The questions range from user's point of view to tester point of view. From user point of view, the questions may be related to response time of the software solutions. For remaining it may be from analysis phase to testing phase. Whatever may be the likely questions to encounter it is important to develop performance evaluation systems with easy to use by team members. - Selecting appropriate evaluation techniques and corresponding performance metrics required. The prediction process using these techniques and metrics do help team members of software systems to evaluate the performance at various phases of SDLC. Several authors worked towards this direction to increase the acceptance of performance evaluation of software systems at all phases of SDLC [7], [8], [11], [16], [17].

Software Performance Engineering (SPE) is a method to predict the performance of software systems early (analysis phase) in the life cycle [3]. SPE continues through the detailed design, coding and testing stages to predict and manage the performance of the evolving software and to monitor, report actual performance against specifications and predictions. Modeling software systems to predict the performance using Unified Modeling Language (UML) is a better choice since UML is seamless from requirements to deployment.

The Unified Modeling Language (UML) is a graphical modeling language that is used for visualizing, specifying, constructing and documenting software systems [19]. UML is a large and varied modeling language intended to model most application domains, to use in many styles of programming, and at many stages of the development lifecycle. Performance models can be generated from use cases during requirements phase [8], sequence diagrams during analysis and design phase [10] and from state chart and activity diagrams during design phase [7], [11].

In all the approaches, the researchers do not consider the data gathered during feasibility study of the project. During feasibility study, project managers, CEOs, and concerned will participate in discussions with stakeholders to assess the project feasibility. In all these feasibility studies several issues would be discussed: Identification of document, description of current situation, problem description, business and financial aspects, technical aspects and organizational aspects of proposed development, development costs and operational costs, envisaged benefits and recommendation. These technical factors gathered during feasibility study are not accounted for estimating performance in early SDLC. These technical factors can be realized only during detailed design or deployment of software systems. In all the works done in early phases, the authors have not considered the technical factors that are important for early prediction. Keeping these in view, in this paper, we propose an algorithm to predict performance of software systems with technical factors and environmental factors gathered during feasibility studies. We use in this paper,

1-4244-0983-7/07/$25.00 ©2007 IEEE

technical and environmental factors gathered for estimating size using use case point approach [1].

II. RELATED WORK

Smith introduced SPE and the overview of SPE approach exhaustive in [2], [3]. She extended her work to assess the performance of distributed object-systems [5]. Generation of performance models and performance assessment throughout the life cycle is widely discussed in [2], [3]. The method for integrating both design and performance activity in client-server application discussed in [6], [13]. Performance analysis of Internet based software retrieval systems using petrinets and a comparative study proposed in [13]. Performance analysis using different diagrams of (UML) widely discussed in [7], [8], [9], [10], [11], [18]. The use of SPE during software architectural design phase is discussed in [4], [9], [14], [16], [17]. Performance evaluation in the early phase using queuing network models is discussed in [16], [18]. In [7], [8], [11], UML models are transformed into layered queuing network models to predict the performance. [12] Describes the issues in SPE project management. In the literature, several methodologies used to assess the performance of software systems early in the life cycle. Most of the methodologies transform UML based performance models into queuing or layered queuing models and solve the models. However, the data collected during feasibility study not used for predicting the performance. We describe an algorithm that considers technical and environmental factors to assess the performance in this paper.

III. PROPOSED ALGORITHM FOR PREDICTING PERFORMANCE

SPE is important for software engineering and in particular for software quality. Smith describes the SPE process to assess the performance of software systems early in the life cycle by using two models, software execution model and system execution model [2]. The software resource requirements required for software execution model of SPE process can be obtained by use case point method by considering appropriate technical and environmental factors. The guidance provided by the use case point method [1] is used to calculate the effort based on the nature of the actors and use cases available in the use case model developed during requirement analysis. The procedure for estimating effort using use case point approach is well defined in [1].

Each technical and environmental factor is assigned a value between 0 and 5 depending on its assumed influence on the project. The adjusted use case points obtained from use case model are converted into equivalent Lines of Code (LOC) by considering the programming language to be used for implementation. The number of LOCs in turn converted into number of executable statements (in assembler program) [15]. The amount of data in terms of kilobytes is obtained from number of executable statements. Then the software execution model solved with the amount of data obtained and the overhead specifications (computer resource requirement for each of the software request) [2]. The total response time and the total resource requirement of each of the hardware device for the scenario obtained from software execution model [2]. Total hardware device requirement provided as input to the system execution model and the performance metrics such as throughput, response time, residence time, device utilization, queue length for each device and also for the system obtained. Based on these we have developed the following algorithm.

A. Algorithm

We describe an algorithm to evaluate performance of software systems during feasibility study. As mentioned above we use data about technical and environmental factors gathered during cost estimation using use case point [1].

Identify significant scenarios that affect the performance of the software systems
Develop use case model for the scenario
For each actor
    Classify the actors as simple, average and complex
end for
For each category of actor
    Count the number of actors
    Multiply the count by the weighting factor
end For
For all categories
Sum up the product to obtain total unadjusted actor weights (UAW)
end For
For each use case
    Classify the use case as simple average and complex
end For
For each category of use case
    Count the number of transactions
    Multiply the count by the weighting factor
end For
For all categories
    Sum up the product to obtain total unadjusted use case weights (UUCW)
end For
    Sum up UAW with UUCW to obtain unadjusted use case points (UUCP)
    Assign values for technical factors
For each technical factor
    Multiply the value by its weight
end For
Sum up the product to obtain TFactor
Calculate Technical Complexity Factor (TCF)
    TCF = 0.6 + (0.01 * TFactor)
Assign values for environmental factors
For each environmental factor
    Multiply the value by its weight
end For
Sum up the product to obtain EFactor
Calculate Environmental Factor (EF)
    EF = 1.4 + (-0.03 * EFactor)
Multiply UUCP, TCF and EF to obtain adjusted use case point (UCP)
Calculate Lines of Code for a specific programming language
Convert into lines of code in assembler program

Calculate equivalent number of kilobytes
Calculate response time using software execution model
Calculate system response time using system execution model

IV. CASE STUDY

We consider banking application to discuss the validation of our algorithm [2].

A. Description of the Case Study

The banking system we have considered is highly distributed in nature. The database of the application deployed in a mainframe server. The customers can make transactions through the ATM, connected with the server through WAN. The most significant performance scenarios we have considered in banking system are deposit, withdrawal and balance enquiry.

B. UML Models for Banking System

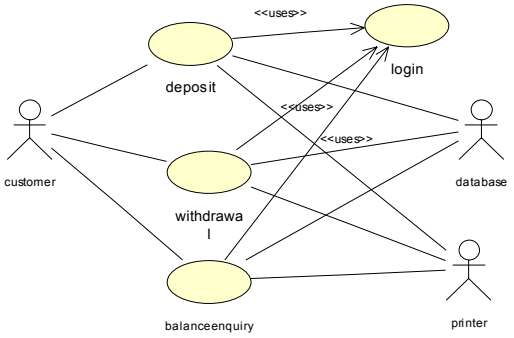


Figure 1. Use case diagram for case study

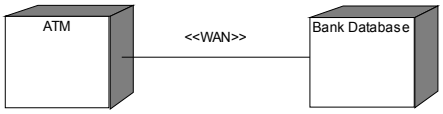
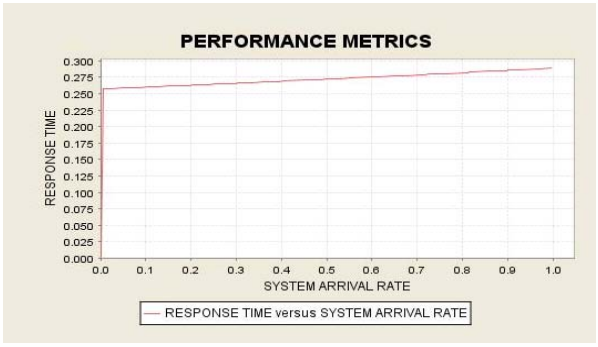
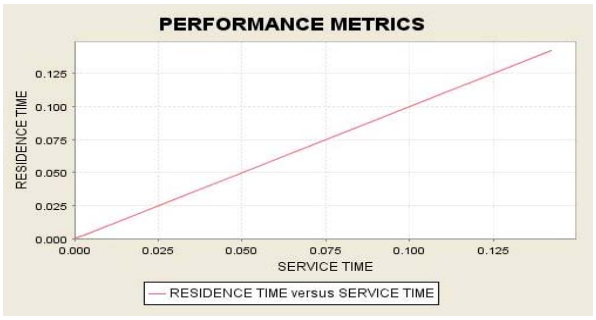


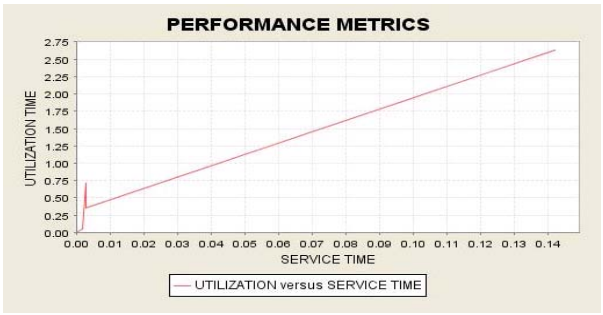
Figure 2. Deployment diagram for case study

The functional requirements for the scenarios of the banking system modeled using use case diagram shown in Figure 1. The software modules for the specified scenarios and the database server deployed as shown in Figure 2.

With these models and considering technical and environmental factors of use case point approach, the calculation proceeds as follows: The use case model given in Figure 1 consists of four use cases namely, *login, deposit, withdrawal, and balanceenquiry* and three actors namely, *customer, database and printer*. The use case *login* is a secondary use case used by the other three use cases. As discussed in [1], the use cases *login, withdrawal and deposit* belong to the category *average*, since the number of transactions for these use cases is between four and seven. The use case *balanceenquiry* belongs to the category *simple* because it has only three transactions. The actors described in the model categorized as follows: *customer* belongs to *complex*, *database* belongs to *average* and *printer* belongs to *simple*. Using the prototype tool UAW is calculated for the actors customer, database and printer, and UUCW is calculated for the use cases login, deposit, withdrawal and balanceenquiry and UAW and UUCW are summed up to get unadjusted use case points.







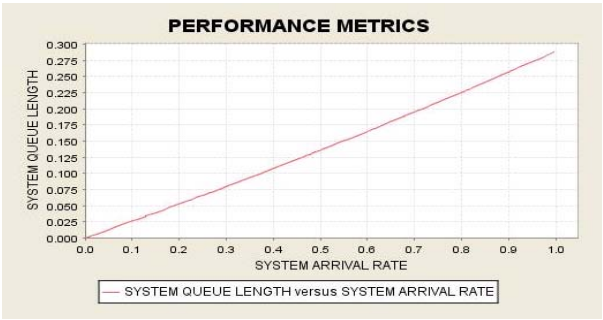


Figure 3. Graphs – Performance Metrics

Since banking application is highly distributed in nature, the values for the technical factors [1] are considered as follows: for T_1 to T_3, T_5, and T_8 to T_{10} the value is 5, for T_4, T_{11} and T_{13} value is 1 and for T_6 value is 4. We assume that modeling of the system to be developed using UML and the programming language to be used for implementation is Java.

By considering these, the values for environmental factors are assigned as follows: for E_1 to E_5 and E_8 value is 5, for E_6 value is 4 and for E_7 value is 2 [1].

By considering the values for technical and environmental factors, and following the steps provided in the algorithm, TFactor and EFactor calculated. Adjusted use case point value is obtained as 47. The LOC is calculated by multiplying UCP by 53 considering Java as the programming language; the number of executable statements (6 statements per line of code) and the amount of data in kilobytes are calculated [15]. The amount of data obtained is used as input (software requirement) for software execution model to obtain the response time. For details regarding use case point approach please refers [1].

C. *Software execution model*

To obtain the complete performance the amount of data obtained from use case point estimation is integrated with the executing platform configuration, the architecture of software modules, and hardware devices with user profile and software workload. The software resources we examine for this case study are: page size, input, data size, server data base access, and output data. Hardware resources are CPU1, CPU2, Disk, Delay, Internet and printer. By assuming approximate values for amount of hardware resource required for each of the software resource, we have obtained the response time for the scenario mentioned in section IV A as 5.041480 seconds.

D. *System execution model*

The System execution model solution is obtained by solving queuing network model for the above software execution model. Different performance parameters such as throughput, response time, residence time, device utilization, queue length for each hardware resource and the system are obtained. The comparison between different performances metrics are obtained in the form of graphs as given in Figure 3.

E. *Discussion of case study*

We have obtained simulated results for system execution model by assigning different values for technical and environmental factors depending on degree of influence (the range for technical and environmental factor from 0 to 5). 1 means weak influence, 3 means average, and 5 means strong influence throughout. The different categories of values and the corresponding response time obtained by software execution model are as follows:

- Strong influence of technical and environmental factors – 6.94 seconds
- Average influence of technical and environmental factors – 7.82 seconds
- Weak influence of technical and environmental factors – 6.06 seconds
- Strong influence of technical and weak influence of environmental factors – 11.24 seconds
- Weak influence of technical and strong influence of environmental factors – 3.59 seconds

The total response time obtained from software execution model for different categories are compared with the response time as 14.6 seconds obtained in [2]. It is observed that response time obtained for category strong influence of technical and weak influence of environmental factors is approximately equivalent. The simulated results of arrival rate vs. response time reported using graphs as shown in Figures 4, 5, 6, 7 and 8.

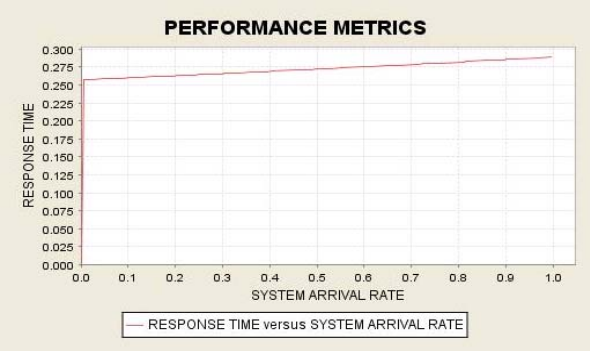


Figure 4. Strong influence of technical and environmental factors

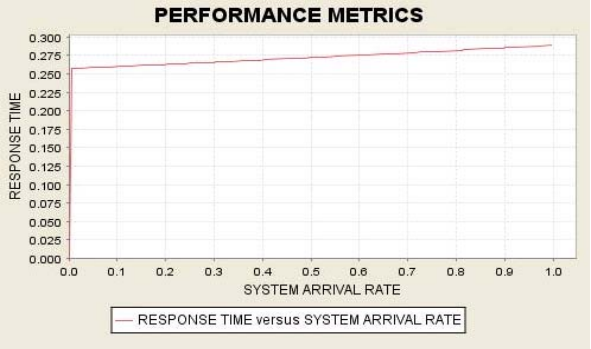


Figure 5. Average influence of technical and environmental factors

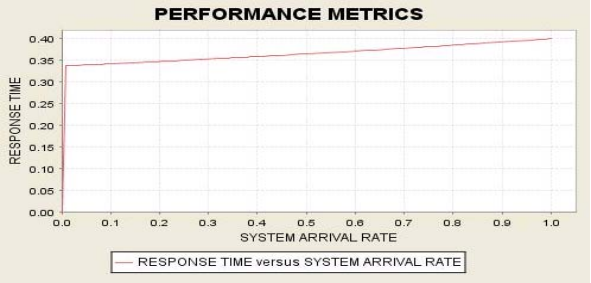


Figure 6. Weak influence of technical and environmental factors

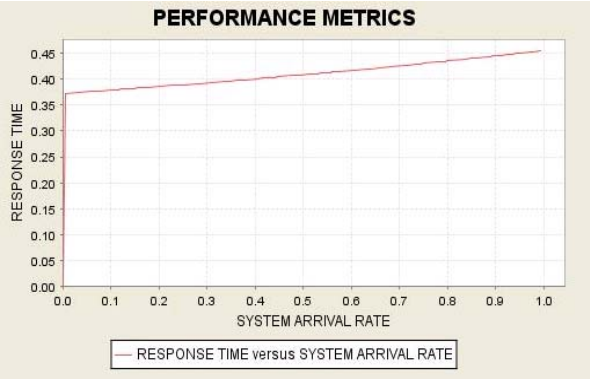


Figure 7. Strong influence of technical and weak of environmental factors

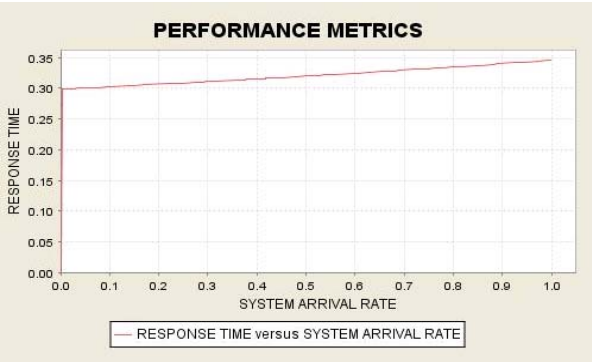


Figure 8. Weak influence of technical and strong of environmental factors

From the simulated results, it is observed that the strong and average influences are almost same. In the weak influence, the response time is higher compared to strong and average influence. Also it is noted that the response time is higher when strong influence values for technical factors and weak influence values for environmental factors are considered compared to weak influence values for technical factors and weak influence values for environmental factors are considered.

V. Conclusion and Future Work

In this paper, we proposed an algorithm to predict performance of software systems during feasibility study. Early performance prediction process requires several technical and environmental factors for predicting performance. We considered the data collected during cost estimation using use case point approach. We present a case study by considering these factors. Further work may include comparing the prediction results obtained in remaining phases of SDLC.

References

[1] Bente Anda, Hege Dreiem, dag I. K Sjobergand Magne Jorgensen, "Estimating Software development Effort based on Use Cases – Experiences from Industry", www.idi.ntnu.no/emner/tdt4290/docs/faglig/uml2001-anda.pdf.

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

[3] Connie U. Smith and Lloyd G. Williams, Performance Solutions, 2000.

[4] Connie U. Smith and Lloyd G. Williams: "Building Responsive and Scalable Web Applications", Proceedings CMGC, December 2000.

[5] Connie U. Smith and Lloyd G. Williams: "Performance Engineering Models of CORBA-based distributed-object systems", Performance Engineering Services and Software Engineering Research, 1998.

[6] Daniel A. Menasce and Gomaa H: "A Method for Design and Performance Modeling of Client/Server Systems", IEEE, 2000.

[7] D.C. Petriu, H Shen, "Applying the UML Performance Profile: Graph Grammar-based derivation of LQN models from UML specifications", in Computer performance Evaluation-Modeling techniques and Tools, (T.Fields, P.Harrison, J.Bradley, U.Harder, Eds.) LNCS 2324, pp.158-177, Springer, 2002.

[8] Dorin Petriu, Murray Woodside: "Analysing Software Requirements Specifications for Performance", Proceedings of the 3rd international workshop on Software and performance 2002, Rome, Italy July 24 - 26, 2002, pp.1-9.

[9] Evangelin Geetha D, Suresh Kumar T V and Rajani Kanth K: "Early Performance Modeling for Web Based Applications", LNCS, Springer Verlag, December 2004, pp. 400-409.

[10] Evangelin Geetha D, Ram Mohan Reddy, Suresh Kumar T V and Rajani Kanth K: "JAPET: A Java Based Tool for Performance Evaluation of Software Systems", Proceedings of SKIMA 2006, International Conference on Software Knowledge Information Management and Applications, December 2006, Chiang Mai, Thailand, pp. 64-69.

[11] Hoda Amer: "Automatic Transformation of UML Software Specification into LQN performance Models using Graph Grammar Techniques", Carleton University, 2001.

[12] Jayshankar Sankarasetty, Kevin Mobley, Libby Foster, Tad Hammer, Terri Calderone, "Software Performance in the Real World: Personal Lessons from the Performance Trauma Team", Proceedings of the 6th international workshop on Software and performance 2007, Buenes Aires, Argentina February 05 - 08, 2007, pp. 201 – 208.

[13] Jose Merseguer, Javier Campos and Eduardo Mena: "Performance Analysis of Internet based Software Retrieval Systems using Petri Nets", ACM 2001.

[14] Lloyd G. Williams and Connie U. Smith, "PASA: A Method for the Performance Assessment of Software Architectures", WOSP '02, July 24-26, 2002 Rome, Italy, pp. 179 – 189.

[15] Robert T. Futrell, Donald F. Shafer, and Linda I. Shafer: "Quality Software Project Management", Pearson Eduacation, 2006.

[16] Simonetta Balsamo Roberto Mamprin Moreno Marzolla: "Performance Evaluation of Software Architectures With Queuing Network Models", 2004.

[17] Simonetta Balsamo Moreno Marzolla, "Performance Evaluation of UML Software Architectures with Multiclass Queueing Network Models", Proceedings of the 5th international workshop on Software and performance, 2005, Palma, Illes Balears, Spain July 12 - 14, 2005, pp. 37 – 42.

[18] Vittorio Cortellessa, Antinisca Di Marco, Paola Inverardi: "Transformations of Software Models into Performance Models", *ICSE'05,* May 15–21, 2005, St. Louis, Missouri, USA, ACM 1581139632/05/0005.

[19] www.omg.org