Developing Software Performance with the Performance Refinement and Evolution Model

Chih-Wei Ho
Dept. of Computer Science, North Carolina State Univ.
890 Oval Drive, Campus Box 8206
Raleigh, NC 27695-8206, USA
+1-919-513-5082
dright@acm.org

Laurie Williams
Dept. of Computer Science, North Carolina State Univ.
890 Oval Drive, Campus Box 8206
Raleigh, NC 27695-8206, USA
+1-919-513-4151
williams@csc.ncsu.edu

ABSTRACT
Performance is one important attribute of a software system. To develop a software system of acceptable performance, the team needs to specify precise performance requirements, design appropriate test cases, and use appropriate techniques to analyze the performance characteristics. However, the lack of a management framework for performance engineering may impair the effectiveness of the techniques. In this paper, we propose the Performance Refinement and Evolution Model (PREM) as a performance management framework. Based on the specification of quantitative measurement and workloads, PREM classifies performance requirements into four levels. At each level, we show what information should be included in the requirement, and what techniques can be used to estimate the performance. Finally we show some applications of PREM.

Keywords
Software performance engineering process; software performance testing; performance requirements.

1. INTRODUCTION
Performance is an important non-functional requirement for a software system. To build a software system with acceptable performance, the development team needs to take performance into consideration through the whole development cycle [3]. Many models, techniques, and methodologies have been proposed to address software performance issues. A development team needs to apply several performance engineering techniques to achieve the desired performance level. However, the lack of a management framework for performance engineering may impair the effectiveness of the techniques. For example, if an unnecessarily complicated performance model is used during the early stages, the development team may need to make assumptions for the unknown but required factors for the model. The assumptions may render the model inaccurate or even useless. The development team needs to choose proper performance techniques, based on the understanding of the performance characteristics of the system, for the techniques to provide useful information.

We propose the Performance Refinement and Evolution Model (PREM) as a framework to manage software performance development. PREM, as illustrated in Figure 1, is a four-level model. Quantitative requirements and workloads specifications distinguish the levels. A higher level means the better understanding of the performance characteristics. At each level, the model describes how the performance requirements and test cases are specified and proper approaches to analyze the performance. The development team can design the performance engineering process and organize the engineering activities based on PREM.

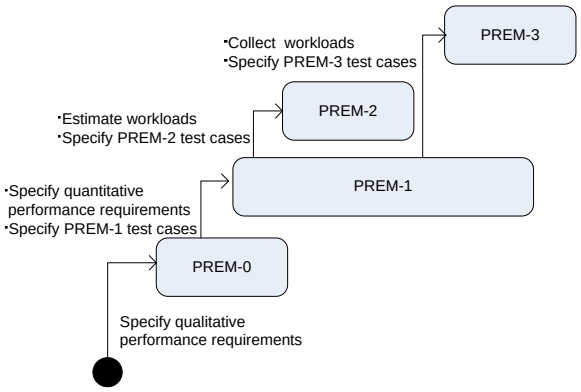

Figure 1. The Performance Refinement and Evolution Model.

2. MODEL COMPONENTS
In this section, we describe the elements in PREM. PREM is based on our previous work [4] that is focused primarily on performance requirements specification. In this paper, we expend the scope of PREM to include performance testing and estimation activities. The elements in this structure are explained as follows.

PREM: PREM is the model we discuss in this paper.

PREM Level: In PREM, performance requirements specifications, testing, and activities are classified in four levels, starting from Level 0. A requirement or test case in a higher PREM level specifies more performance characteristics of the software system, with higher precision. To analyze performance specified at a

WOSP 2007, Feb 5–8, 2007, Buenos Aires, Argentina.

certain PREM level, the development team can apply the activities at the same level.

Starting Criteria: A PREM level has one or more starting criteria. The starting criteria show the required properties of the requirements before the activities at the PREM level can be applied.

Goal Criteria: A PREM level has one or more goal criteria. The goal criteria show the required properties of the requirements for them to be classified as being at a certain PREM level. If a performance requirement satisfies all the goal criteria of PREM level *n*, the requirement is called a PREM-*n* requirement. A performance test case specified based on a PREM-*n* requirement is called a PREM-*n* test case. A performance analysis activity for PREM-*n* requirements is called a PREM-*n* activity.

Activity: When a requirement satisfies all the starting criteria of a PREM level, and the development team decides to achieve the PREM level, the team shall apply some of the activities for the level. After the selected activities are successfully performed, the performance requirement specification and related test cases shall achieve the goal criteria of the PREM level.

Performance Type: A performance type is an attribute that is used to describe the system performance. Currently, PREM is designed to work with the following performance types:

- *Response time* is the time requirement for the completion of an operation;
- *Throughput* presents the quantity of operations that need to complete in an amount of time.

Testing Approach: Performance testing shows whether the software system achieves the desired performance goals. The PREM testing approach shows how test cases shall be designed to reflect the performance requirement of a PREM level.

Specification Part: Specification parts are the elements that are used to specify requirements. A specification part may be mandatory or optional. After the specification parts are identified for a requirement, transformation rules may be applied to generate presentations for the requirement. In some presentations, short names are more appropriate than full descriptions. Therefore, a specification part may be assigned with a shorter name, or *alias*.

Presentation: A presentation shows a requirement in a particular form that is transformed from the specification parts with a transformation rule. In this paper, we provide transformation rules for two important presentations: natural-language-based requirements specification, and performance test case. The transformation rules shown in this paper are intended for general purposes. We do not claim that the transformation rules listed in this paper cover all cases. A development team should define specific transformation rules for the application domain.

3. MODEL DETAILS

In this section, we present PREM based on the meta-model described in Section 2. Due to the limitation of space, the activities for each PREM level are provided in a separate report [5].

3.1 PREM Level-0

Starting Criteria:	Functional requirements are specified in requirements documents.
Goal Criteria:	Qualitative performance requirements are specified in requirements documents.
Testing Approach:	Qualitative evaluation.

PREM-0 represents performance requirements with only qualitative, casual descriptions. An example of PREM-0 requirement is: *The authentication process shall be completed quickly*. PREM-0 requirements are essentially placeholders for future work in performance requirement specification. By specifying PREM-0 requirements, customers identify the operations for which performance matters. PREM-0 requirements are the starting point which customer and developer will refine or evolve to more precise specifications via PREM-1 or higher requirements.

PREM-0 requirements help the development team focus on the appropriate performance requirements. At this level, brief, high-level performance requirements are sufficient for the team to make initial requirements decisions. A free-form, natural-language-based specification should be used for PREM-0 requirements. Therefore, we do not provide the specification parts for PREM-0 requirements. However, identifying PREM-0 requirements helps the team to find out the values for specification parts at higher PREM levels.

3.2 PREM Level-1

Starting Criteria:	Performance requirements are defined qualitatively in requirements documents.
Goal Criteria:	• Quantitative performance requirements are specified in requirements documents. • Appropriate test cases are specified.
Testing Approach:	Run the test scenario and then take performance measurement.

PREM-1 represents performance requirements with quantitatively-measurable expectations. An example of PREM-1 requirement is: *After the user enters the user name and password, and clicks the Submit button on the Log In page, the response time for authentication and Main page rendering shall be within three seconds*. Quantitatively measurable specification is the first step to test the requirement in an objective way.

The first step for PREM-1 requirement specification is to find performance scenarios for the PREM-0 requirement. If use cases are employed to describe the requirements, a scenario is an end-to-end flow specified in a use case. One presentation for performance scenarios, which is used in Software Performance Engineering (SPE) [9], is UML sequence diagrams with features from message sequence chart (MSC) [6], including loops, alternatives, and so on. Most of the MSC extensions are included in UML 2.0.

After the performance scenarios are determined, the next step is to specify quantitative expectations. Anecdotal experiences (e.g., [8]), although not validated, can offer some hints of how well a software system should perform. Several types of models can be used to analyze the performance at PREM Level 1. Bertolino and Morandola show how UML sequence diagrams and deployment diagrams with proper annotation can be used to estimate the performance of a component-based system [2]. The execution graph described in SPE also provides PREM-1 estimation. These per-

formance models break the performance scenario into several small steps. Estimating the response time for each small step is usually easier than estimating the response time for the whole scenario.

We use the following three parts to describe a PREM-1 response time requirement of an operation:
preparation: The phrases that describe the preconditions that must hold before the operation can be carried out. The ***preparation*** part is optional.
event: A phrase that describes the event that initiates the operation on which the response time measurement is taken. Response time measurement starts as soon as the events have happened.
time: The constraint for the time needed to complete the operation.

The transformation rules for PREM-1 response time requirements are shown in Table 1. The optional components are surrounded with square brackets.

Table 1. PREM-1 response time transformation rules

Natural Language	Test Case
[After ***preparation***,]when ***event***, the response time shall be less than ***time***.	[call ***preparation***.alias] startTime ← current time call ***event***.alias endTime ← current time assert(***time*** > endTime – startTime)

For a concurrent system, the specification of throughput is ambiguous if the workloads are not specified. Therefore, we do not provide the specification parts for PREM-1 throughput requirements. To specify throughput requirements and test cases for concurrent systems, PREM-2 specification parts and transformation rules should be used. However, in a system where tasks are processed sequentially, throughput is the reverse of response time. In such systems, throughput requirements can be translated to response time requirements.

3.3 PREM Level-2

Starting Criteria:	Quantitative performance measurements are specified with the requirements.
Goal Criteria:	• Estimated average or peak workloads are specified with the requirements in the requirements documents. • Appropriate test cases are specified.
Testing Approach:	Generate asynchronous requests according to the specified workloads. After the requested operations are completed, take performance measurements.

PREM-2 performance requirements and test cases are specified with estimated workloads. Workloads specification describes the frequency of the requests for the functionalities of a software system. From the users' perspective, a software system has different performance when the software is under different workloads. Therefore, workloads specification is required for concrete performance requirements. An example of PREM-2 requirement is: *After the user opens the Log In page, the user enters the valid user name and password, and clicks the Submit button. On average, this scenario happens 20 times per minute. After the user opens the Log In page, when the user enters the valid user name and password, and clicks the Submit button, 80% of the response time shall be less than 3 seconds*. PREM-2 requirements can be specified with either or both the peak or average workload estimations.

Workloads can be estimated based on experience or observation. However, the process to derive experience- or observation-based estimation tends to be ad-hoc. Joines et al. develop several worksheets for workload estimation [7]. Although based heavily on experience and observation, compared to the ad-hoc approach, the worksheet approach is more systematic. The workloads information for a previous release or other similar systems, if available, is a good source of workloads estimation for the system under development. Even if no such data are available, we may still get the information from existing business data, such as market or industrial research reports available from the government or private research companies.

Several performance models, such as system execution model in SPE, can be used to evaluate the performance under estimated workloads. Performance models that can be used for PREM-2 performance analysis are summarized by Balsamo et al [1].

In addition to PREM-1 specification parts, the following parts are necessary to describe a PREM-2 response time requirement:
load level: A description of whether the peak or average workloads are used. The value can be either "peak load" or "average."
process: A series of ascending numbers that describe the instances of request arrival time for the system. Because different processes require various parameters, specialized specification parts and transformation rules need to be specified for each type of process.
degree: A phrase describing of the degree how the time constraint is satisfied. The value of ***degree*** can be one of following: "the average", "*n*%", or "the maximum."

For throughput requirements, we need to specify the following parts:
preparation and ***event***: As described in PREM-1 response time specification parts.
load level and ***process***: As described in PREM-2 response time specification parts.
rate: The expected throughput of the requirement.

The transformation rules for PREM-2 performance requirements are shown in Table 2. The specification part ***load level*** is not used in the test case. However, in natural-language-based specification, ***load level*** makes the specification clearer by showing whether the system is under the peak or average load. In the test case transformation rules, variables with parenthesis, for example, reqTime(), are arrays with the index starting from 1. The function "size" returns the size of the array; "average" returns the average of the numbers in the array; and "max" returns the largest number in the array. A special array, thread(), contains the threads that are used to create the concurrent requests. ***preparation*** and ***event*** can be called in a thread contained in the thread() array.

3.4 PREM Level-3

Starting Criteria:	Quantitative performance measurements are specified with the requirements.
Goal Criteria:	• Peak or average workloads are collected and specified in the requirements documents. • Appropriate test cases are specified.
Testing Approach	Generate asynchronous requests according to the specified workloads. After the requested operations are completed, take performance measurements.

PREM-3 represents quantitative performance requirements with workloads from collected data. An example of a PREM-3 requirement is "*The system is running under the heaviest possible workloads defined in Appendix IV. The average response time for displaying the promotional message on the mobile tablet after a customer enters a lane where the promotional items are located shall be below 1 second.*" At PREM Level-3, the workloads description defines the workloads for different types of requests. If the description is too lengthy or complicated to be specified with the requirement, it can be moved to a separate document, such as *Appendix IV* in this example.

PREM-3 and PREM-2 share the same starting criteria. As soon as quantitative PREM-1 requirements are specified, the development team is ready to perform PREM-2 and PREM-3 activities. If PREM-3 is needed for a project, the development team can develop PREM-2 models and collect PREM-3 data at the same time. However, this does not suggest that a team should skip PREM Level 2. Although PREM-3 requirements provide more accurate performance information than PREM-2, PREM-3 activities take much more time. If PREM Level 2 is skipped, the requirements will stay at Level 1 until PREM-3 data are collected. By applying PREM-2 activities, the team can have early estimation of the system performance. The only exception might be the situation where the workloads information from a very similar system, perhaps an earlier release of the system under development, is already available. In such a situation, the existing data can be

Table 2. PREM-2 transformation rules

Performance Type	Natural Language	Test Case
Response Time	[After ***preparation***,]***event***. In ***load level*** situation, this scenario happens ***process***. [After ***preparation***,] when ***event***, ***degree*** of the response time shall be less than ***time***.	reqTime() ← numbers from ***process*** [for i ← 1 to size(reqTime) call ***preparation***.alias in thread(i)] for i ← 1 to size(reqTime) wait until time reqTime(i) in thread(i) respTime(i) ← currentTime call ***event***.alias respTime(i) ← currentTime – respTime(i) wait until all requested operations are completed select(***degree***) case "on average": assert(average(respTime) < ***time***) case "*n*%": success ← 0 for all t in respTime() if(t < ***time***) success ← success + 1 assert(success / size(respTime) > *n*%) case "the maximum": assert(max(respTime) < ***time***)
Throughput	[After ***preparation***,]***event***. In ***load level*** situation, this scenario happens ***process***. The system shall handle such requests at the rate of ***rate***.	reqTime() ← numbers from ***process*** startTime ← currentTime [for i ← 1 to size(reqTime) call ***preparation***.alias in thread(i)] for i ← 1 to size(reqTime) wait until time reqTime(i) call ***event***.alias in thread(i) wait until all requested operations are completed runTime ← currentTime - startTime assert(size(reqTime) / runTime > ***rate***)

used directly, and PREM-2 can be skipped.

PREM-3 performance requirements specification and testing are similar to those of PREM-2. PREM-3 and PREM-2 requirements have the same specification parts. However, in PREM-3, workloads for multiple types of requests are specified. For each type of request, we need to specify zero or more ***preparation***, one ***event***, and one ***process***. A natural language transformation rule for PREM-3 workloads specification is provided in Figure 2. The rest part of the requirement is similar to that at PREM Level-2. Similarly, in PREM-3, a multi-thread test case is used to generate the requests. The test case transformation rule can be adapted and modified from PREM-2.

The system is running in ***load level*** situation. [After ***preparation***$_1$,] ***event***$_1$. This scenario happens ***process***$_1$. [After ***preparation***$_2$,] ***event***$_2$. This scenario happens ***process***$_2$. …

Figure 2. PREM-3 natural language transformation rule for workloads specification

4. CONCLUSION

In this paper, we show how performance requirements can be specified by identifying the required parts in the requirements. Transformation rules can be applied on the specification parts to generate different presentations of requirements. We use PREM, a model for software performance development, to find out the specification parts and corresponding values. We show how PREM principles and related techniques can be applied. We also show the transformation rules for natural-language-based specification and test case for PREM Level 1 through 3. When performance requirements are refined and evolved with more performance-related information, the corresponding test cases can be regenerated to reflect the new performance characteristics. We have used PREM to define a software engineering process and specify performance requirements for a Web application [5].

The specification parts we propose in this report need to be validated to show their efficacy. To validate the specification parts, we need to show that:

- The parts are sufficient for all performance requirements.
- Performance requirements specified with the parts are less ambiguous than specification without using the parts.
- Performance requirements specified with the parts are more complete than specification without using the parts.

Currently, we do not have strong evidence suggesting when the requirements refinement should stop. We believe the required precision for performance requirements specification depends on the type of project. For example, for a system that processes its job sequentially and does not accept concurrent requests, PREM-1 requirements and analysis seem sufficient. However, in a hard real time system controlling multiple machines in a factory, PREM-3 requirements and analysis may be required. We need to conduct empirical studies to see how the application of PREM affects the outcome of a project.

5. ACKNOWLEDGMENTS

The authors would like to thank the comments and feedbacks from the RealSearch reading group at NC State University. This research was funded by the Center for Advanced Computing and Communication.

6. REFERENCES

[1] Balsamo, S., A. D. Marco, and P. Inverardi, "Model-Based Performance Prediction in Software Development: A Survey," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 295-310, May 2004.

[2] Bertolino, A. and R. Mirandola, "Towards Component-Based Software Performance Engineering," in *Proceedings of the 6th Workshop on Component-Based Software Engineering*, pp. 1-6, Portland, OR, May 2003.

[3] Fox, G., "Performance Engineering as a Part of the Development Life Cycle for Large-Scale Software Systems," in *Proceedings of The 11th International Conference on Software Engineering*, pp. 52-62, Nice, France, Mar 1990.

[4] Ho, C.-W., M. J. Johnson, E. M. Maximilien, and L. Williams, "On Agile Performance Requirements Specification and Testing," in *Proceedings of Agile 2006 International Conference*, pp. 47-52, Minneapolis, MN, Jul 2006.

[5] Ho, C.-W. and L. Williams, "Managing Software Performance Engineering Activities with Performance Refinement and Evolution Model (PREM)," Department of Computer Science, North Carolina State University *Technical Report No. TR-2006-29*, September 2006.

[6] ITU, *ITU-T Recommendations Z. 120 (04/04) Message Sequence Chart (MSC)*, 2004.

[7] Joines, S., R. Willenborg, and K. Hygh, *Performance Analysis for Java Web Sites*, Boston, MA, Addison-Wesley, 2003.

[8] Sevcik, P., "How Fast is Fast Enough" *Business Communications Review*, 2003, Available at http://www.bcr.com/architecture/network_forecasts%10sevcik/how_fast_is_fast_enough?_20030315225.htm.

[9] Smith, C. U. and L. G. Williams, *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*, Boston, MA, Addison-Wesley, 2002.