From Use Case Model to software performance results: An SPE approach

Dushyant Kumar Singh
dks4885@gmail.com
Department of Computer Engineering
Aligarh Muslim University, Aligarh

Abstract
Software Performance Engineering (SPE) is a comprehensive methodology for constructing software to meet the performance goals. The process continues through different stages of Software Development Life Cycle (SDLC) either detailed design, coding, testing and deployment phase. Evaluating performance at different stages of SDLC have been addressed by a number of methodologies but the data collected during the feasibility & requirement analysis phase is not considered yet for the performance prediction. Here, we derive an algorithm to predict the performance of software systems. This use UML Use Case models as software specifications to be taken as input to the technique. A performance parameter, Use Case Performance Point (UCPP) is generated as an internal result. And the performance model (a mathematical function) using this value outputs the response time value. Response Time is the performance measure considered for prediction.

Keywords – Software Performance Engineering, Software Development Life Cycle, Unified Modeling Language, Use Case model

1 Introduction
1.1 Introducing SPE
Software Performance is an important non functional attribute of software systems for producing quality software. Performance is a characteristic of a software product that one could, in principle, measure by sitting at the computer with stopwatch in his hand. “Performance is the degree to which a software system or components meets its objectives for timeliness” [1].

Performance is a make-or-break quality for software. Nearly everyone runs the performance problems at one time or another. Poor performance costs the software industry millions of dollars annually in lost revenue, decreased productivity, increased development and hardware costs and customer relations. Today’s software development organizations are compelled to achieve more with less. This means upgrading software applications to accommodate a new infrastructure, improve response time, throughput or both. In an ideal scenario, performance would be engineered into software starting early in the development process. This early identification and resolution of performance problems is desirable as the cost of design change increases with the later phases of the software development life cycle.

Software Performance Engineering (SPE) is a method to predict the performance of software systems early (analysis phase) in the life cycle. 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.

1.2 Introducing UML
Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting artifacts of software intensive systems. Example artifacts are requirements, architecture, design, source code, test cases, prototypes etc [5]. 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. UML represents the unification of a number of efforts to build notations for expressing models of Objects Oriented Analysis and Design (OOAD). With UML, one can represent different structures in a consistent and coherent way using object-oriented principles [6].

Performance models can be generated from collaboration and deployment diagrams after the software systems are developed and deployed, from state chart and activity diagram during design phase, from sequence diagram during analysis and design phase [3]. No approach is yet developed considering the UML specifications at early (feasibility / requirement analysis) phase of SDLC. 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. The UML model corresponding to these software specifications is the Use Case diagram.

An important part of the UML is the facility for drawing Use Case diagrams. These are used during the analysis phase of a project to identify and partition system functionality. They separate the system actors and use cases.

Actor is the external object that supplies stimuli to the system and receives output from it. These could be a human being or some another system. Use case describes the behavior of the system when one of these actors sends one particular stimulus. This behavior is described textually. Behavior describes the nature of the stimulus that triggers the use case; the inputs from and outputs to other actors, and the behaviors that convert the inputs to the outputs. This textual behavior can be represented graphically and the representation is called Use Case Diagrams.

Actors are drawn as a little stick diagram and use case as an oval. Actors are connected to the use case with lines [6, 7]. Fig 1 shows a sample Use Case Diagram.

Fig 1: Sample Use Case Diagram

1.3 Contribution & Outline of Paper

These UML Use Case specifications are used to calculate a performance index (an intermediate result) called Use Case Performance Point (UCPP). This UCPP value is fed as input to the Performance model (a mathematical function) to get an output in form of a performance measure. Performance measure taken considered in this work is “Response Time”. Fig 2 gives the design flow for the given methodology.

Section 3 details the complete approach for performance prediction. Algorithm to calculate UCPP is discussed in section 3.1 with mapping of UCPP to statistical response time data in section 3.2. Section 4 tells about developed performance model and how response time can be calculated from this model. At last section 5 concludes the work in paper and its future extensions.

Fig 2: Design flow of performance prediction technique

2 Performance Background

SPE was coined by Dr. Connie U. Smith and Dr. Lloyd G. Williams in 1990. Earlier approach used was “fix-it-later” approach. Above this, SPE continues through detailed design, coding & load testing phase to predict performance [1, 2]. Later, a book named “Performance Engineering for Software Systems”, a classic book of SPE was given by Dr. C.U. Smith. SPE-ED, a tool for performance evaluation is the outcome of his research.

Jose Merseguer transforms UML specifications as Activity diagrams to SPN in [11], while Sequence & State diagrams to analyzable PN models in [10]. Salvatore Distefano proposed a distributed algorithm implemented in a tool “WebSPN” to solve Petri Nets [12]. Other PN solvers are “TimeNET” & “GreatSPN”.

Moreno Marzolla developed a performance evaluation tool “UML-Ψ” based on process-oriented simulation. Hence, technique is named as “Simulation Based Performance Modeling of Software Architecture”.

3 The Proposed Technique

3.1 UCPP Algorithm
The algorithm for calculating the UCPP value is given here.

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 use case
   Multiply the count by the weighting factor
end For

For all categories
   Sum up the product and divide net by count of use case to obtain total unadjusted use case weight average (UUCWA)
end For

Sum up UAW and UUCWA to get the Unadjusted Performance Point (UPP).

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)

Now, The UPP value is adjusted with TCF since performance of the software systems is affected by the technical factors associated in the software development.

Finally, Use Case Performance Point (UCPP) is calculated as

UCPP = UPP * TCF

3.2 Mapping Methodology
The statistical response time data as gathered form case studies of earlier researches is mapped to UCPP values to derive a performance model.

Response time values for a number of software systems are gathered. Let these be

RT1, RT2 ………………. 

Corresponding to these, UCPP values can be calculated as

UCPP1, UCPP2 ………………. 

These values of RT and UCPP can than be mapped using any of the approximation / curve fitting method. Better is the exponential least square approximation.

The approximation function for exponential least square approximation is

\[ Y = a \cdot \exp(bx) \]

Y => Response Time
X => UCPP

Therefore, Performance model developed will be

Response Time = a * exp (b * UCPP)

a, b are performance deciding factors (constants)
4 Results

Finally, the performance prediction of software systems at the early stage of SDLC can be performed with the available approach.

Input: Use Case Model
Intermediate result: UCPP, calculated using the algorithm discussed in section 3.1
Output: Response Time, outcome of performance model using UCPP value.

5 Conclusion & Future work

A Performance model is derived here as performance prediction technique. Background studies conforms that, this is the only technique for performance prediction at such an early stage of SDLC. This technique is more statistical in nature, motivated from COCOMO for effort estimation. Technique evolved is the extension of effort estimation to performance prediction.

First and foremost important future work to this is realization of the technique. This involves taking real performance data for performance model derivation. A more number of case studies must be considered for testing of the performance model.

6 References