Software Performance Evaluation by Models

Murray Woodside

Department of Systems and Computer Engineering
Carleton University, Ottawa, Canada
cmw@sce.carleton.ca

1 What Software Performance Evaluation Is About

To assess the performance of a software design or product on a particular platform, performance measures such as response time and throughput are determined, and compared to required values. If they miss the mark some changes may be made in the design while it is still being shaped. Early analysis and corrections save time and money and reduce the risk of project failure through performance failure, as has been argued eloquently by Smith [1] and Hesselgrave [2]. Fig. 1 shows how different performance information is connected to different aspects of software development.

<table>
<thead>
<tr>
<th>DEVELOPMENT ASPECT</th>
<th>PERFORMANCE INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Requirements</td>
<td>Performance Requirements</td>
</tr>
<tr>
<td>Analysis</td>
<td>Budgets</td>
</tr>
<tr>
<td>Early Software</td>
<td>Model and Predictions</td>
</tr>
<tr>
<td>Descriptions + Workload Estimates</td>
<td>OK?</td>
</tr>
<tr>
<td>Software + Tests, and Measurement Tools</td>
<td>Test Measures</td>
</tr>
</tbody>
</table>

Fig. 1. Performance Information During Software Development

To manage the performance aspects of a new product, evaluation extends all through the process of development:

1. It begins with a sound and testable statement of requirements.
2. This may be broken down during analysis into budgets for subsystem responses.
3. For early analysis of a new product or new feature, some estimates of workload demands are made based on experience, and attached to the early design description.
4. Measured demands are used once they are available.

© Springer-Verlag Berlin Heidelberg 2000
5. A model is created and validated and used to make early predictions, which may be used to recognize problems and improve the design.

6. As code is produced, tests are run to give better demand estimates, and finally performance measures from system performance tests, to compare to requirements.

This chapter focuses on building models for performance prediction based on information attached to the early high-level design descriptions. This is the bridge linking the expertise of the designer to that of the modeler. The design descriptions may be custom made for performance analysis, as in the execution graphs in Smith’s SPE method [1], or may be created as part of the software design, as in the software CAD tools CAEDE [3] and SARA [4] and their successors such as the tools based on SDL that are described in [5].

The early design descriptions divide into two broad categories or viewpoints, which have characteristic performance parameters and steps for extracting performance models from them. A broad integrating framework connects the two designer viewpoints with the model viewpoint.

This discussion focuses on abstract simplified models. Simulation models are an alternative approach, explored for instance in MIDAS [6]. Material may be found elsewhere on performance requirements [2], measurements [8], testing [9] and on methodology for improving designs [1].

The difficult problems in evaluation come from the complexity of software designs, the limitations in the capabilities of some model types to capture the essentials of the system behaviour, and the lack of agreed languages or notations for early design descriptions. In practice, the gap between designers and evaluators is also a challenge both for obtaining the models and for interpreting the results. An integrated methodology is needed, and this chapter describes techniques which may help to bridge the gap.

2 Performance Parameters within Software Descriptions

Designers use a variety of more or less formal descriptions and notations to represent the requirements, the intended behaviour, and the design of a software system. Although these descriptions have many forms they fall into two broad categories or viewpoints, one describing structure and one describing behaviour, with a few cases that combine the two.

- A behaviour description (called a path model here) captures the sequence of operations of the program in response to a particular stimulus. It tends to be used more at the beginning of a project, to capture requirements and functionality.
- A description of structure (called a module model here) captures the design as a set of modules and relationships. It is used to define the architecture, design and deployment of the system.