Towards a UML Profile Enabling Performance Prediction during Software Design

Zhongfu Xu and Axel Lehmann

Universität der Bundeswehr München
Institut für Technische Informatik
85577 Neubiberg, Germany
Email: {xu, lehmann@informatik.unibw-muenchen.de}

1 Introduction

The software development community should be provided with an efficient methodology and tool support for proactive, quantitative performance management during functionality-driven software development. In this paper we present a UML (Unified Modeling Language) profile as part of the framework (shown in Figure 1) that allows the evaluation of software architecture decisions and design solutions with reference to their impact on software performance before coding and testing software.

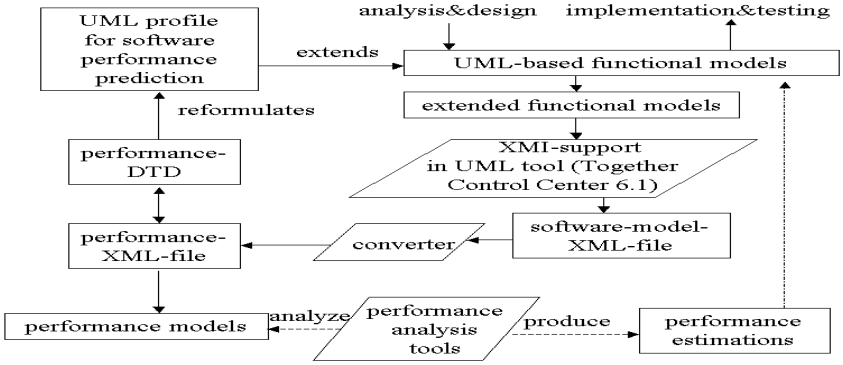

Figure 1: Framework for predicting software performance during analysis and design

2 UML Profile for Software Performance Prediction

The UML class diagram in Figure 2 shows the UML extensions defined in the UML profile for software performance prediction. The extensions are defined as the stereotypes of the UML metamodel classes. New data types and tags associated with the stereotypes are omitted because of space restriction.

UseCases capture the functional requirements of the software system under development. A UseCase may be required by more than one users (human beings or other systems). The users generate **UserRequests** when they require a UseCase. Users of

the same UseCase are categorized so that the intensity and temporal properties of the UserRequests generated by each **UserCategory** can be described uniquely with a **RequestPattern**.

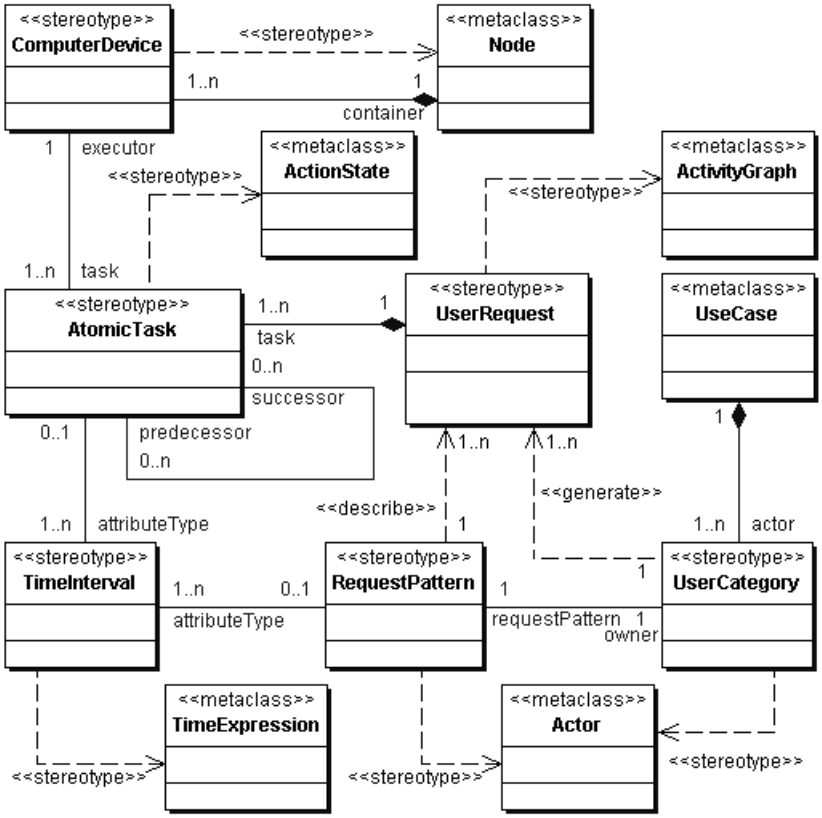

Figure 2: UML extensions in the software performance prediction UML profile

The processing of a UserRequest is abstracted as a sequence of **AtomicTasks**. The AtomicTasks are related to each other in a predecessor/successor relationship. Each AtomicTask is not decomposable and its work can not be interrupted. An AtomicTask is executed solely by one physical **ComputerDevice**. A ComputerDevice may execute more than one AtomicTasks. The ComputerDevices are contained in the runtime computational **Nodes**. Temporal properties of AtomicTasks and RequestPatterns are specified by using **TimeIntervals**.

UML extensions stated above have been used to extend functional software models, primarily use case- design- and deployment models, with information about 1) user request pattern and 2) which computer devices are required for executing the system functionality, and how their parameters and quantitative behavior are. Based on the extended software models, queuing network models are built for software performance evaluation.