Model Transformation in Software Performance Engineering

Antinisca Di Marco[1] and Raffaela Mirandola[2]

[1] Dipartimento di Informatica, Università di L'Aquila, L'Aquila, Italy
adimarco@di.univaq.it
[2] Dipartimento di Elettronica e Informazione - Politecnico di Milano, Milano, Italy
mirandola@elet.polimi.it

Abstract. Nowadays it is widely recognized the crucial role played in the software development process by the analysis of extra-functional properties (and especially performance) at the architectural level. To foster this kind of quantitative analysis we envisage the need to transform the performance model generation and analysis into a rigorous and sound discipline. To this end we intend to exploit the knowledge (acquired by other disciplines) in the area of model transformation, and import both reasoning and methodologies in the software performance engineering. In this paper we investigate the area of performance model derivation and analysis focusing on model transformation; we propose an initial taxonomy for the area of performance analysis at software architecture level and we delineate our suggestions towards a software performance model driven engineering.

1 Introduction

During recent years, the way software systems are designed and built is undergoing great changes. The main emerging trends are: the central role played by the software architecture concept in the software development; the model driven paradigm of development; and the attention given to the analysis of extra-functional properties during software development. Our work aims at merging these trends, as it intends to pursue the analysis of extra-functional properties (such as performance) at architectural level basing on model driven techniques. The software architecture approach to the design of software applications focuses on the high level modeling of an application in terms of coarse-grained components, interaction patterns among them and overall interconnection topology, abstracting away low level details [28]. The underlying idea is that better software systems can result from modeling and analyzing their relevant architectural aspects since the early phases of the development lifecycle. Analyzing the Software Architecture (SA) of a system aims at predicting the quality of the system before it has been built, to understand the main effects of an architecture with respect to quality requirements. This prediction, based on analysis methodologies applied to some suitable system model, can be exploited to drive decisions about what components and connectors should be used and how they should be architected so as to meet the quality requirements imposed on the design.

Out of any explicitly reference to model driven development, several models (and related tools) have been proposed so far for the analysis of extra-functional properties

C. Hofmeister et al. (Eds.): QoSA 2006, LNCS 4214, pp. 95–110, 2006.

for software architecture, especially for SA described by UML diagrams [3,6]. The basic idea underlying these models (and tools) is the definition of a bridge between the architectural model and existing performance models. However, in these work the focus has been mainly on the ”derived analysis model” rather than on the way of obtaining it.

At the same time, in the software engineering community the focus, in software development has been shifted from a code-centric approach to a model-centric approach with the definition and the adoption of the model driven development (MDD) paradigm, where the focus of model transformations is on a transformation path from high level to platform specific models (up to the executable code) of a software application. Model transformation has been actively investigated in the last years for what concerns the transformation from model-to-code considering both the derivation of intermediate models and the direct code generation. In this field some taxonomies of model transformation approaches have been defined to help a software developer choosing the method that is best suited for his needs [33,11,18].

However, to the best of our knowledge, only few work exist in the literature including in this model driven generation the analysis of extra-functional properties, such as performance [32]. Aim of our work is to exploit the knowledge in the area of model transformation and to import it in the software performance community in order to make the performance model derivation a rigorous and sound discipline rather than an art left to skilled people.

Indeed, carrying out the analysis of extra-functional quality attributes of a SA, can be seen as a transformation process that takes as input some ”design-oriented” model of the software system (plus some additional information related to the extra-functional attribute of interest) and generates an ”analysis-oriented” model, that lends itself to the application of some analysis methodology. However, defining such a transformation from scratch could be quite cumbersome, for several reasons; for example, the large ”semantic gap” between the source and target model of the transformation, and the different target analysis notations one could be interested in to support different kind of analysis (e.g. queueing networks, Markov processes).

To alleviate these problems, we have launched a research initiative aiming at merging these trends, and in particular, we intend to change the focus in the area of performance model derivation, paying more attention to model transformation. As a first step towards this goal we propose an initial taxonomy of model transformation for the area of analysis of performance properties. Note that we do not intend to suggest a general classification for software performance engineering works (for this aspect we refer to [3,6]), rather we look at these works from the different point of view of model transformation to understand how to exploit these techniques for the performance model derivation. Furthermore we present a comparison among some of the existing approaches, and finally, starting from the analysis of the state of the art, we devise some guidelines that pave the way to software performance model driven engineering.

In this paper we present the first results we have obtained in this area. Our long term goal is the definition of a general framework for software performance model driven engineering, where it would be possible to choose which models and which transformation method apply for addressing a specific target problem.