Chapter 42
Parameterized Transformation Schema for a Non-Functional Properties Model in the Context of MDE

Gustavo Millán García
Pontifical University of Salamanca, Spain

Rubén González Crespo
Pontifical University of Salamanca, Spain

Oscar Sanjuán Martínez
Carlos III University, Spain

ABSTRACT

The integration between design models of software systems and analytical models of non-functional properties is an ideal framework on which lay the foundation for a deep understanding of the architectures present in software systems and their properties. In order to reach this integration, this chapter proposes a parameterized transformation for a model of performance properties derived from a system model in the MDE context. The idea behind a parameterized term is to leave open the transformation framework to adopt future improvements and make the approach reusable. The authors believe that this kind of integration permits the addition of analysis capabilities to the software development process and permits an early evaluation of design decisions.

INTRODUCTION

In 1987 F. Brooks published a famous paper called "No Silver Bullet: Essence and Accident in Software Engineering" (Brooks, 1986) as a clear allusion to the fact that there are not magic solutions to the fundamental problems affecting the overrall development of software systems. This paper focused on the inherent complexity of the software and its invisibility . Therefore, the field of software engineering is

DOI: 10.4018/978-1-5225-3923-0.ch042

trying to address this inherent complexity by using models to better understand the characteristics of a software systems increasingly complex and bigger. The use of models allow us to see a problem more clearly and help us to visualize properties that can not easily seen from the encrypted form of the system.

The approach proposed by MDE (Model-Driven Engeneering) (Atkinson, 2003) is a clear recognition of the concept of model as an key artifact within the construction and development activity of a software system. MDE discusses the software development process by promoting the use of models as first-class artifacts. But not only promotes the use of different models but fundamental proper integration and consistency allowing an integrated view of different properties and dimensions of system software at different levels of abstraction.

One of the major advances in the use of models in software engineering is the integration of design tools and semiautomatic code generation (Czarnecki, 2000) from these models. Thereby many potential errors resulting from manual coding could be avoided . A clear example of these design and integration tools is the Eclipse development environment and the Eclipse Modeling Project (Fundation, 2001) .

There is however a class of models that have nothing to do directly with the primary objective of generating system code, but rather to allow quantitative analysis of non-functional properties of system, such analysis would be based on performance models, reliability or security models.

This chapter discusses MDE software development view, focusing on the transformation for non-functional properties models, more precisely a model based on queue theory (Lazowska, 1984). The chapter provides a study of the transformation issues of a system model to a performance model based on queuing theory. As a solution to the gap problem between models this chapter provides a parameterized transformation schema applied to this kind of context. At the end of the chapter, we describe a simple example for proving our approach

THE MDE VIEW OF SOFTWARE DEVELOPMENT PROCESS

The model-driven engineering software view of software development has its roots in a general method to represent the details of the system in some kind of formalism (model) focused mainly on software and hardware issues and then through a chains of transformations to obtain a software system encoded in some execution platform.

The concept of model as a key artifact in the software development is widely accepted. However this term is often overused and misused. A definition of model term could be found in (ModelWare, 2007).

Def. Model: "formal representation of entities and relationships (abstraction) with some correspondence (isomorphism) to the real world for some purpose (pragmatism)."

Many models have been proposed in software development, but not all are formal. This variety of informal models has contributed to make more difficult tracking of consistence between models.

The transformation process promoting by MDE could be shown in the following idealized sequence $M1 \xrightarrow{tM1M2} M2..Mn \xrightarrow{tMncode} code.$

In this scheme of transformations chain *M1, M2, ..., Mn* are design models of software system at several details or levels (formalization) of abstraction and *t* represents any correspondence or association between elements of different models implied in the process. Ideally, this transformation is hierarchical by the different levels of abstraction (isomorphism).