Combining System Development and System Test in a Model-Centric Approach

M. Born¹, I. Schieferdecker¹, O. Kath², and C. Hirai³

¹Fraunhofer FOKUS, Kaiserin-Augusta-Alle 31, 10589 Berlin, Germany
{born, schieferdecker}@fokus.fraunhofer.de
²Technical University-Berlin, Franklinstr. 28/29, 10623 Berlin, Germany
kath@cs-tu-berlin.de
³Hitachi SDL,Yokohama, Japan
c-hirai@sdl.hitachi.co.jp

Abstract. In this paper we will present, how a model centric approach cannot only be used to rapidly develop the system but also at the same time to support the provision of the system tests which is an integral part of the overall system development. The key technology used to achieve this is a set of meta-tools which contains model repository generators and model transformer generators.

1 Introduction

Model centric development of software system has recently become an important software engineering strategy for handling the complexity and the increasing requirements to larger and highly distributed software systems. This phenomena can be observed in different domains, from telecom over public sector to automotive and defense. The fundamental idea of model centric development is to replace the programming language code as the main artefact in the development process by models. These models exist on different levels of abstraction throughout the development process. They are not independent from each other and have various relations like trace or transformation relations. These relations allow to establish and maintain consistent views on the system, spanning over different abstraction levels from requirements through specifications and test cases. Furthermore, the degree of development automation is substantially increasing through the consequent application of model transformations wherever possible.

Another important observation is that the overall resources which are spent for a development project are distributed to a large percentage (up to 70%) to the requirements/analysis phase and testing phase. Therefore, the model centric approach will only be a success if it is possible to reduce the development effort in exactly these phases without a lack of quality. The approach which we want to discuss in this contribution integrates the system design tightly with the development of system tests, starting at higher levels of abstraction.

Our architecture for system development follows the idea of the Model Driven Architecture (MDA) as introduced by the Object Management Group [17]. Within the
MDA, models are classified into platform independent models (PIM) and platform specific models (PSM). The term platform in the MDA sense refers to wide spread integration platforms like J2EE [12], CORBA Components [18] or Web services as well as to domain specific platforms like Autosar [16] for the automotive domain. The idea of MDA is that PIMs can be automatically transformed into PSMs and programming language code can be generated from PSMs.

The test software can be modelled and developed in exactly the same way as the functional system software. Abstract testing artefacts are derived and modelled from the existing information in PIMs. These platform independent test models (PITs) can be transformed to platform specific test models (PSTs), potentially taking additional information from PSMs. Then, the programming language test code, i.e. the code of the test components of the test system, can be generated from the PSTs. This situation is depicted in Fig. 1, still completely independent from any particular modelling language, test language or programming language.

Fig. 1. The overall approach for combining system development and system test

In order to set up such an environment with concrete modeling, testing and programming languages, we apply a pattern which is used for each individual technique to be integrated. The pattern is applied for PIM, PSM, system code and for PIT, PST and test code. The pattern (Fig. 2) has the following steps:

- The modeling principles and relations of the modeling technique have to be defined.
- The modeling principles and relations have to be formalized in a metamodel.
- A notation for the modeling technique has to be defined.
- A process and guidelines of how to use the modeling technique have to be defined.
- Possible connections to other modeling techniques in the overall process have to be defined.