Testing aspect in Model Driven Software Development

Gábor Bátori and Domonkos Asztalos

Big changes are occurring in the field of software development. New technologies appear whilst others vanish. The most important motivations of the new technologies are to make the software development lifecycle shorter and to create reusable components. Model-Driven Architecture (MDA [1]) is one approach to this challenge, where the problem domain is modeled at a high level of abstraction, so-called Platform Independent Models (PIMs), and the implementation is derived from these models. The core technology of the MDA is the Unified Modeling Language (UML), which is the standardized modeling language for object-oriented software development. By supporting the MDA with an executable form of the UML, called xUML [2], we can generate 100% of the source code from the high level model with minimal manual intervention. The xUML process is a rigorous and precise development technique. A key part of this process is the ability of simulating and testing the PIMs without any specific platform. This new technology allows to begin the testing process at the early phase of the development lifecycle, but UML technology focuses primarily on the definition of system structure and behaviour and provides only limited means for the testing aspect of the modeled problem. However, testing of the high level models is crucial, because the majority of the software defects (appr. 60-70 per cent) can be eliminated at this level of abstraction.

Aspect-Oriented Programing (AOP [4]) has been proposed as a technique for improving separation of concerns, such as security, logging, error handling etc., in software. Aspect Oriented Software Development techniques allow one to modularize crosscutting concerns into separate "aspects" of a system and integrate those aspects with other kinds of modules throughout the software development lifecycle. Testing is one of these aspects, it is independent of the high level model itself, only depends on the requirements of the software, thus we can define it separately from the application model.

Our main goal is to create a compiler (weaver) which is capable for weaving special code blocks into the platform independent model so that the derived new model can communicate with a tester (e.g. TTCN-3 [3]). These alterations preserve the functionality of the original model, but provide interfaces in order that a tester acquires information about the state of the model. We use "tags" in the UML model to indicate the points where the weaver can insert the special codes. TTCN-3 is used as a test description language in order to reuse these early phase tests at the implementation level, because these functional tests can provide the basis of other types of tests (e.g. performance tests).

References

[1] R. Soley: Model Driven Architecture: An Introduction. `http://www.omg.org`.

[2] Supporting Model Driven Architecture with eXecutable UML Kennedy Carter 2002

[3] ETSI ES 201 873-1: The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language. V2.1.0 (2001-10), 2001; also an ITU-T standard Z.140.

[4] Gregor Kiczales et al.: Aspect oriented programming ECOOP'97 LNCS 1241, pp 220-242. Springer-Verlag, June 1997.