Scenario-based testing from UML/OCL behavioral models

Application to POSIX compliance

Kalou Cabrera Castillos · Frédéric Dadeau · Jacques Julliand

Published online: 25 February 2011
© Springer-Verlag 2011

Abstract We present in this article a way to produce test suites applied to the POSIX mini-challenge based on a behavioral model of a file system manager written in UML/OCL. We illustrate the limitations of a fully automated test generation approach, which justifies the use of test scenarios as a complement to a functional testing approach. Scenarios are expressed through regular expressions describing sequences of operations, possibly punctuated by intermediate states that have to be reached by the execution of the model. Scenarios are unfolded into extended sequences of operations that are played on the model using symbolic animation techniques. We experimented our approach by testing the conformance of two different file systems w.r.t. the POSIX standard: a recent Linux distribution and a customized Java implementation of POSIX used to evaluate the relevance of our approach and its complementarity with a structural test generation approach.

Keywords Model-based testing · Scenarios · Symbolic animation · UML/OCL · POSIX mini-challenge

1 Introduction

The use of formal methods becomes crucial when one wants to design a system that requires a high level of correctness. In this context, a mini-challenge is proposed to design and verify a POSIX compliant flash-based system [14]. Even though this proposal is not the first attempt to put formal methods into a real case study, it illustrates the increasing need for formal modeling and verification that aims at producing safe and secure software. In the process, testing takes an important part which is used to provide evidences that, first, the system will not present certain kinds of execution errors and second, that it behaves as expected.

Motivations Our proposal is to consider a model-based testing approach [2] that relies on a formal model of the system. The latter is used to compute the test cases from a given test selection criterion, as a sequence of operation calls intended to cover functional requirements. This process also provides the oracle of the test, namely the expected results that will be used to ensure the conformance of the system w.r.t. the model. We apply this technique in the context of the POSIX mini-challenge, and thus, we have designed a formal model of the POSIX file system by means of an UML model whose behavior is given by means of additional OCL constraints [28] on the operations of the classes described in the class diagram.

Automated test generation processes based on structural test selection criteria suffer from a number of limitations. Indeed, the systematic structural analysis and partitioning of the model allows to produce the smallest test sequences that cover a static criteria as the control flow paths, the decisions, the conditions, etc. Specific configurations of the system that needs many executions of the same control flow path or a specific sequence of operation calls are not reached. They can be easily identified by a test expert that, thanks to its know-how,
can easily describe a generic way to reach such configurations. We therefore propose to improve the structural test case generation by a scenario-based testing approach. Scenarios are a means provided to the tester to describe dynamic test purposes that complement the structural test selection criteria.

**Contributions** This article presents a scenario expression language dedicated to object-oriented models that makes it possible to precisely describe scenarios with test driving possibilities. These latter are, roughly speaking, regular expressions over the alphabet represented by the operations of the system. Usual constructs on regular expressions are considered, such as (bounded) iterations, choice, etc. This process relies on the use of the symbolic animation of the UML model in order to instantiate the sequences of operations along with relevant parameter values. This language has originally been designed for B machines during the French RNTL POSÉ project in the context of the security testing [19]. It is now extended to the UML/OCL notations and used for a larger purpose.

Figure 1 summarizes the process of the scenario-based testing approach that we propose. It relies on two artifacts: a behavioral model of the system which can be animated and a scenario which describes how to animate the model so as to produce abstract test cases. The scenario-based testing engine unfolds the considered scenario in order to obtain a test case specification that is played using a symbolic animator on the UML model. All resulting test sequences that satisfy the scenario are instantiated and kept as abstract test cases.

**Outline** The article is organized as follows. Section 2 explains the subject and the scope of the case study. Then the principles of the symbolic animation of models are described in Sect. 3. A scenario description language for object-oriented models, guiding the symbolic animation, is presented in Sect. 4. Experiments are reported in Sect. 5. We present and discuss related works in Sect. 6. Finally, Sect. 7 concludes and presents future work.

2 UML model of POSIX

This section presents the UML model that we have designed from the informal specifications. This model is used in the test generation process. It includes a realistic subset of the functionalities required in the challenge. We designed our model from the Open Group Base Specification of Unix.\(^1\) Figure 2 depicts the class diagram that we consider.

2.1 Considered subset of UML/OCL

The model aims at being used by the Test Designer tool, commercialized by the Smartesting company.\(^2\) This tool generates automatically model-based tests from a UML model [5] with OCL code describing the behaviors of the operations. Test Designer does not consider the whole UML notation as input; it relies on a subset named UML4ST (UML for

\(^1\) http://www.unix.org/single_unix_specification/.