Test generation from state based use case models

Sidney Nogueira\textsuperscript{1,2}, Augusto Sampaio\textsuperscript{2} and Alexandre Mota\textsuperscript{2}

\textsuperscript{1} Mobile Devices R&D Motorola Industrial Ltda, Rod SP 340, Km 128, 7 A, Jaguariuna, SP 13820 000, Brazil
\textsuperscript{2} Centro de Informática, Universidade Federal de Pernambuco, Caixa Postal 7851, Recife, PE 50732-970, Brazil

Abstract. We present a strategy for the automatic generation of test cases from parametrised use case templates that capture control flow, state, input and output. Our approach allows test scenario selection based on particular traces or states of the model. The templates are internally represented as CSP processes with explicit input and output alphabets, and test generation is expressed as counter-examples of refinement checking, mechanised using the FDR tool. Soundness is addressed through an input–output conformance relation formally defined in the CSP traces model. This purely process algebraic characterisation of testing has some potential advantages, mainly an easy automation of conformance verification and test case generation via model checking, without the need to develop any explicit algorithm.

Keywords: Natural language; Test model; Use cases; CSP; Test generation; Conformance testing

1. Introduction

Testing consists in verifying whether the actual system behaviour matches the intended one. Hence, testing is related to some model (sometimes a mental model) \cite{UPL06}, which is the basis for the construction of test cases. Test cases are then constructed to assess the correctness of particular features of the system. A good set of test cases is directly related to how adequately the model captures the features of the implementation under test (IUT). Nevertheless, designing test cases manually can yield inconsistent test cases even if the model is trustworthy. Moreover, when the model changes, test cases must be updated and this is not always feasible manually, mainly when the number of tests grows. So manual craft and execution of tests can be costly and error prone.
The selection of a good set of test cases and their automation aim at making the testing process more effective, less susceptible to errors and less dependent on human interaction. The purpose of model-based testing (MBT) is to use explicit models to automatise testing. Instead of manual design, tests are generated by a tool that processes the input model. Complementary, the generated tests can be automatically run against an implementation. Conformance testing is a kind of MBT whose objective is to check whether an IUT satisfies its specification according to some defined relation (conformance relation); an inherent assumption is that the class of specifications can be modelled in some known formalism so that it can be related with the specification model (test hypothesis). There are several conformance relations [BJK+05] mainly based on formal notations like Finite State Machines (FSM) and Labelled Transition Systems (LTS). Based on a conformance relation, test cases can be automatically generated from the model using algorithms [HBB+09] that ensure the satisfaction of properties by the generated tests (e.g. soundness).

Despite the advances of conformance testing, both in theoretical and practical fields, there are process related barriers for its wider adoption, such as the introduction of new tools and paradigms in traditional testing flows. Forcing the users to adopt new tools and formal notations does not seem to work, so user-friendly notations and interactive tools are necessary to reduce the gap between formal specifications required by MBT tools and informal specifications adopted in standard testing processes, often described in natural language. A promising direction to overtake this barrier is to develop domain specific approaches [Ber07].

Due to its convenience and easy to use notation, and adherence to object-oriented development methodologies, use cases [Gro07] have been adopted as the input model for test generation in many development contexts. Despite being part of object-oriented development methodologies, use cases can be developed by analysts and testers who do not have object-oriented programming skills. Very often use cases are the only available requirements documentation. Moreover, from use cases it is possible to validate the system in the early stages of the development, minimising costs.

Particularly, use case templates [CS08] are the standard input models for conformance testing in the Brazil Test Center (BTC) project [Sam05], a cooperation between the Federal University of Pernambuco and Motorola Inc., in the context of testing embedded software that run in mobile phones. A use case template is a document that defines the syntactic elements for authoring use cases and their relationships. It is structured as a set of features, each one described as a set of use cases. A feature is a clustering of individual requirements that describe a cohesive, identifiable unit of functionality; often described as input and output events that flow sequentially between the actors (mainly the user) and the GUI of the IUT. Thus, use case templates are very suitable for the description of the features to be tested. In BTC, use cases are described in a domain specific language for mobile applications [Tor06, Lei07]: this is a Controlled Natural Language (CNL), which is a small subset of English with a fixed grammar. The template has proved suitable [NCT+07] to specify individual features (mobile device functionalities) as well as several patterns of feature interaction.

Originally, use cases did not have a formal semantics, required for a fully automated test generation and reasoning about the properties of the generated test cases. However, more recently, several semantics have been proposed to use cases [CANM08, NFLT06, BL02, WP99, HVFR05, SC08] aiming at generating test cases. LTS and FSM are the main models used as basis to automate test generation from use case models; these are very concrete models and often adopted as the operational semantics of more abstract process algebras like CSP [Ros98], CCS [Mil89] and LOTOS [88089]. Contrasting with operational models (such as LTS or FSM), process algebra models can naturally evolve to incorporate additional requirements; the operators of a process algebra also allow complex models to be built from simpler ones in a compositional way. Test generation can take advantage of this more abstract level to be formalised in terms of the process algebra itself, generating concise, precise and proved correct solutions.

In BTC, from the use cases, models expressed in the CSP process algebra [Ros98] are automatically constructed [CS08] and used as input for our CSP based automatic generation approach of test scenarios. In this approach, instead of developing explicit algorithms, test scenarios are obtained from counter-examples of refinement verifications using FDR [For05], a model checker for CSP. There is no explicit manipulation of state spaces or control flow; test selection is captured by CSP test purposes, which are CSP processes that describe the properties of interest to be captured by the generated tests. In this approach, soundness is established via a conformance relation, named espio, which defines the set of observations considered in testing: the implementation must produce a subset of the outputs for the inputs that are specified. As test hypothesis it is assumed that the class of implementations to be tested can be specified by some I/O process. Generated test scenarios are used to construct test cases that are sound with respect to espio, ensuring that, only incorrect implementations can fail the tests.