Côte de Resyste

Test Automation with Formal Methods

Jan Tretmans

Software Testing

Software quality is an issue that currently attracts a lot of attention. Software invades everywhere in our society and life and we are increasingly dependent on it. Moreover, the complexity of software is still growing. Consequently, the quality, functional correctness and reliability of software is an issue of increasing importance and growing concern. Systematic testing of software plays an important rôle in the quest for improved quality.

Despite its importance, testing is often an underexposed phase in the software development process. Moreover, testing turns out to be expensive, difficult and problematic. One source of problems can be an imprecise or ambiguous specification, so that a good basis for testing is missing. Another reason is the usually manual and laborious testing process without effective automation, so that testing is error-prone and consumes many resources. Furthermore, quite often the testing phase gets jammed between moving code delivery dates and fixed custom delivery dates. On the other hand, research and development in testing have been rather immature. Testing methodology is mostly governed by heuristics.

Fortunately, this situation is gradually improving. Triggered by the quest for improved quality and imposed by increased product liability, testing is considered more important and treated more seriously. Being a software tester is becoming a true profession.

Côte de Resyste

The project *Côte de Resyste*—Conformance Testing of Reactive Systems—aims at improving the testing process by using *formal methods*. In *Côte de Resyste* we develop theories, methods and tools which enable fully automatic testing of software systems based on formal specifications. In doing so, *Côte de Resyste* concentrates on *functional testing* of *reactive systems*. Functional testing involves checking the correct behaviour of a system: does the system do what it should so (as opposed to, e.g., testing the performance or robustness). Reactive systems are mostly technical, event-driven systems in which stimulus/response behaviour is important, such as embedded systems, communication protocols and process control software. Administrative systems are typically not reactive systems.

Côte de Resyste is a cooperation between Philips Research Laboratories Eindhoven, Lucent Technologies R&D Centre Enschede, Eindhoven University of Technology and the University of Twente. It is a 4 year, 23 man-year project supported by the Dutch Technology Foundation STW. Our main challenge is to develop test techniques and tools with a high practical applicability, while starting from a well-defined and sound theoretical basis. The applicability and usability is evaluated by performing case studies supplied by Philips, Lucent and associated partners.

Formal Methods

Currently, most system specifications are written in natural languages, such as English or Dutch. Although such informal specifications are easily accessible, they are often incomplete and liable to different and possibly inconsistent interpretations. Such ambiguities are not a good basis for testing: if it is not clear what a system shall do it is difficult to test whether it does what it should do.

With formal methods systems are specified and

modelled by applying techniques from mathematics and logic. Such formal specifications and models have a precise, unambiguous semantics, which enables the analysis of systems and the reasoning about them with mathematical precision and rigour. Moreover, formal languages are more easily amenable to automatic processing by means of tools. For example, tools exist that are able to verify fully automatically the absence of deadlock based on a formal description of the design. Whereas until recently formal methods were a merely academic topic, their use in industrial software development is now increasing, in particular for safety critical systems and for telecommunication software.

Testing with Formal Methods

A formal specification is a precise, complete, consistent and unambiguous basis for design and code development as well as for testing. This is a first big advantage in contrast with traditional testing processes where such a test basis is often lacking.

A second advantage of the use of formal specifications for testing is their suitability to automatic processing by means of tools. Algorithms have been developed which derive tests from a formal specification. These algorithms have their theoretical underpinning in the theories of *labelled transition systems, process algebra* and *testing preorders*. Moreover, these algorithms have been implemented in tools leading to automatic, faster and less error-prone test generation. This opens the way towards completely automatic testing where the system under test and its formal specification are the only required prerequisites. Formal methods provide a rigorous and sound basis for algorithmic and automatic generation of tests. Tests can be formally proved to be valid, i.e., they test what should be tested, and only that.

TORX: A Tool for Formal Testing

Within *Côte de Resyste* we are developing the formal testing tool TORX. TORX integrates automatic test generation and automatic test execution in an *on-the-fly* manner. *On-the-fly* testing implies that derived test actions are immediately executed and, moreover, that only the part of the test that will actually be executed is derived (*lazy test generation*). TORX is a prototype tool with which some academic and industrial case studies have been successfully tested. One of the examples was a chat-box protocol, which was tested based on specifications in the formal languages LOTOS and *Promela*. Due to the *on-the-fly* testing method long tests consisting of more than 500,000 test events could be generated and executed completely automatically. From a set of 28 mutants of the chat-box protocol, with carefully inserted faults, all erroneous implementations could be successfully detected.

TORX is constantly being extended. We are now adding *test purposes*, which allow controlling the direction of testing. An extension with *symbolic test derivation* is under development.

Applications of Testing with Formal Methods

Currently, TORX and its accompanying test methods are evaluated by applying them to embedded consumer electronics software of Philips and to an access network protocol of Lucent. One of the evaluation studies concerns a communication protocol between video recorders and television sets for downloading channel presets. The results of this study are promising: some faults were detected which had slipped through the conventional testing procedures. This strengthens our view that within a few years it will be possible to perform automatic testing of reactive, modestly complex, industrial software systems based on their formal specifications. The expectation is that the extra effort required for developing the necessary formal system specifications will be more than compensated by faster, cheaper and more effective testing.

This does not mean that all problems have been completely solved, yet. One of the most important research questions, which is currently investigated, is how the completeness and coverage of an automatically generated test suite can be expressed, measured and, ultimately, controlled. Even more intriguing is the question how test suite coverage can

be related to a measure of product quality. After all, product quality is the only actual reason to perform testing.

Information

More information, including (references to) articles, can be found at the *Côte de Resyste* web page http://fmt.cs.utwente.nl/cdr, by e-mail from cdr@cs.utwente.nl, or from:

Arjan de Heer,
Lucent Technologies R&D Centre Twente,
NL-7521 PL Enschede,
arieheer@lucent.com;

Lex Heerink,
Philips Research Laboratories Eindhoven,
NL-5656 AA Eindhoven,
lex.heerink@philips.com;

Loe Feijs,
EESI
Eindhoven University of Technology,
NL-5600 MB Eindhoven,
feijs@win.tue.nl;

Jan Tretmans,
University of Twente,
NL-7500 AE Enschede
tretmans@cs.utwente.nl.