Côte de Resyste
Automatic model-based testing of communication protocols

Axel Belinfante, Ed Brinksma, Jan Feenstra,
Jan Tretmans and René de Vries
University of Twente
Formal Methods and Tools group, Department of Computer Science
P.O. Box 217, 7500 AE Enschede, The Netherlands
{belinfan, brinksma, feenstra, tretmans, rdevries}@cs.utwente.nl

1. 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. This also applies to software in mobile systems. 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 under-exposed 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.

2. *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 to 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 do (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 [2, 4]. 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.

3. 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. Until recently formal methods were a merely academic topic, but now their use in industrial software development is increasing, in particular for safety critical systems and for telecommunication software. One particular Dutch project where formal methods have been used successfully is the control system for the storm surge barrier in the Nieuwe Waterweg.

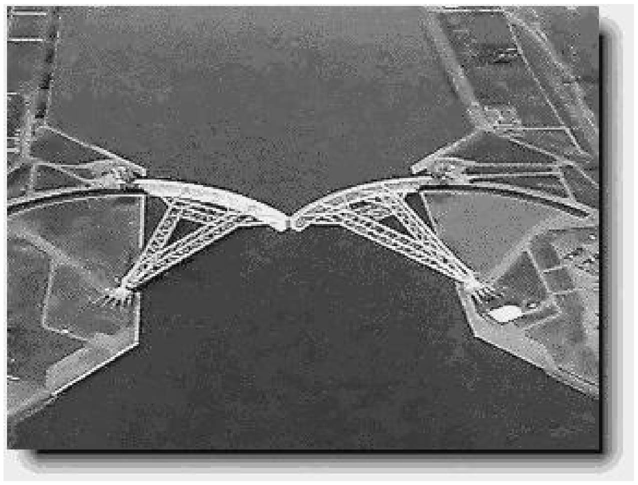

Figure 1. The storm surge barrier in the Nieuwe Waterweg near Hoek van Holland is completely software controlled. Flooding of Rotterdam is avoided and must be reliable. During the system design the formal methods Z and PROMELA have been used. Testing was performed based on these models.

4. 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 [3]. 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.

5. 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 [1]. 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 which could not be achieved with traditional testing tools.

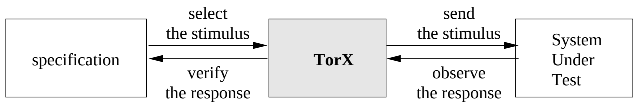

Figure 2. The TORX tool selects stimuli from the specification and stimulates the system under test. The responses are observed and verified against the specification.

6. Applications of Testing with Formal Methods

Currently, TORX and its accompanying test methods are evaluated by applying them to different case studies. 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. The second case study is Lucent Technologies V5.1 access network protocol. A third case study is a mobile application in the context of the 'Rekeningrijden' project. Interpay is developing a system for automated fee charging for turnpike roads. The system should debit an electronic purse and register a balance increment on an account at the toll gate when a vehicle passes. To ensure that the payment transactions are correctly processed by the system, the system should be tested. This testing activity should increase the confidence in the correct functioning of the system. Due to the fact that many cars can pass a toll gate simultaneously, the number of parallel transactions in progress can be large. This fact contributes to the complexity of the generation, execution and analysis of test experiments. The testing activity done by *Côte de Resyste* and Interpay is scheduled in the beginning of 2001.

7. Perspectives

Current work concentrates on improving TORX, on developing methods for effective selection of test sets and on generic test execution environments. The case studies strengthen 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.

References

[1] A. Belinfante, J. Feenstra, R. d. Vries, J. Tretmans, N. Goga, L. Feijs, S. Mauw, and L. Heerink. Formal test automation: A simple experiment. In G. Csopaki, S. Dibuz, and K. Tarnay, editors, 12^{th} *Int. Workshop on Testing of Communicating Systems*, pages 179–196. Kluwer Academic Publishers, 1999.

[2] Dutch Technology Foundation STW. *Côte de Resyste* – COnformance TEsting of REactive SYSTEms. Project proposal STW TIF.4111, University of Twente, Eindhoven University of Technology, Philips Research Laboratories, KPN Research, Utrecht, The Netherlands, 1996. `http://fmt.cs.utwente.nl/CdR`.

[3] J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software—Concepts and Tools*, 17(3):103–120, 1996. Also: Technical Report No. 96-26, Centre for Telematics and Information Technology, University of Twente, The Netherlands.

[4] R. d. Vries, J. Tretmans, A. Belinfante, J. Feenstra, L. Feijs, S. Mauw, N. Goga, L. Heerink, and A. d. Heer. Côte de Resyste in PROGRESS. In S. T. Foundation, editor, PROGRESS *2000 – Workshop on Embedded Systems*, pages 157–164, Utrecht, The Netherlands, October 13 2000.