Testing Times:
On Model-Driven Test Generation for Non-Deterministic Real-Time Systems

Ed Brinksma
University of Twente
Formal Methods and Tools Research Group
PO Box 217, 7500 AE Enschede, Netherlands
brinksma@cs.utwente.nl

Abstract

Although testing has always been the most important technique for the validation of software systems it has only become a topic of serious academic research in the past decade or so. In this period research on the use of formal methods for model-driven test generation and execution of functional test cases has led to a number of promising methods and tools for systematic black-box testing of systems, e.g. [2, 7, 3, 4]. Most of these approaches are limited to the qualitative behaviour of systems, and exclude quantitative aspects such as real-time properties. The explosive growth of embedded software, however, has also caused a growing need to extend existing testing theories to the testing of real-time reactive systems. In our presentation we present an extension of Tretmans' ioco *theory for test generation [6] for input/output transition systems that includes real-time behaviour.*

A central concept in the non-timed theory is the notion of quiescence*, which characterizes systems states that will not produce any output response without the provision of a new input stimulus. By treating quiescence as a special sort of system output the notion of behavioural trace can be generalized to include observations of quiescence. In turn, this leads to an implementation relation that defines unambiguously if implementated behaviour conforms to a given specification model, viz. if after all specified generalized traces of the implementation all possible generalized outputs are allowed according to the specification. Or, more informally, if all ouputs and quiescences are correctly predicted by the specification. In practice, the above implementation criterion means that implementations can be more deterministic than their specifications.*

Although it is good engineering practice not to introduce unnecessary nondeterminism in reactive systems, nondeterminism is often unavoidable in the context of testing, and should therefore be part of a sensible testing theory. The reason for this is twofold:

- *Although the implementation under test may be deterministic, it can often only be tested through a testing environment that includes operating system features, communication media, etc. that typically introduce nondeterminism into the observed behaviour.*
- *The implementation under test often consists of concurrent components in an asynchronous parallel composition. The loss of information about the relative progress of components results in nondeterministic properties of their integrated behaviour.*

Our proposed extension of the ioco *theory to real-time systems is based on an operational interpretation of the notion of quiescence. This gives rise to a family of implementation realations parameterized by observation durations for quiescence. We define a nondeterministic (parameterized) test generation algorithm that generates test cases that are sound with respect to the corresponding implementation relation. This means that when an implementation fails any of the generated tests it must be non-conforming. The algorithm is also exhaustive in the sense that for every non-conforming implementation a test case can be generated that will detect its non-conformance.*

Because we work with a continuous model of time, viz. the positive reals, the meaning exhaustiveness of our test generation algorithm is limited. Even if we had an unlimited amount of time at our disposal we could only derive countably many test cases by repeated runs of the derivation algorithm, whereas obviously there are uncountably many relevant test cases. Fortunately, under mild and reasonable assumptions we can show that repeating runs of our test generation algorithm is still in some sense complete: in the limit it will detect non-conformance almost everywhere, i.e. it will detect all faults whose occurrence in time is not negligable ($\approx$ does not take place at only isolated points in time).

The development of a real-time testing theory forces us to confront modelling issues with respect to physical aspects of time and implementation. From a physical point of view it is questionable whether negligable behaviour can be implemented. This has also implications for specification formalisms that can be used to specify such behaviour (e.g. timed automata [1]). It would seem that realistic specifications allow for tolerances in the evaluation of clock conditions. This would then introduce a third source of nondeterminism in the testing theory of real-time systems.

References

[1] R. Alur and D. Dill, A theory of timed automata *Theoretical Computer Science*, 126(2), 183–235, 1994.

[2] A. Belinfante, J. Feenstra, R.G. de 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, *Int. Workshop on Testing of Communicating Systems 12*, pages 179–196. Kluwer Academic Publishers, 1999.

[3] J.-C. Fernandez, C. Jard, T. Jéron, and C. Viho. Using on-the-fly verification techniques for the generation of test suites. In R. Alur and T.A. Henzinger, editors, *Computer Aided Verification CAV'96*. Lecture Notes in Computer Science 1102, Springer-Verlag, 1996.

[4] J.-C. Fernandez, C. Jard, T. Jéron, and C. Viho. An experiment in automatic generation of test suites for protocols with verification technology. *Science of Computer Programming – Special Issue on COST247, Verification and Validation Methods for Formal Descriptions*, 29(1–2):123–146, 1997.

[5] T. Jéron and P. Morel. Test generation derived from model-checking. In N. Halbwachs and D. Peled, editor, *Computer Aided Verification CAV'99*, pages 108–121. Lecture Notes in Computer Science 1633, Springer-Verlag, 1999.

[6] 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.

[7] J. Tretmans and E. Brinksma. TorX: Automated Model-Based Testing,. in: Editors: A. Hartman and K. Dussa-Ziegler, First European Conference on Model-Driven Software Engineering, Nuremberg, 2003