Testing in The Distributed Test Architecture: An Extended Abstract

Robert M Hierons
School of Information Systems, Computing and Mathematics
Brunel University, UK
rob.hierons@brunel.ac.uk

Abstract

Some systems interact with their environment at a number of physically distributed interfaces/ports and when testing such a system it is normal to place a local tester at each port. If the local testers cannot interact with one another and there is no global clock then we are testing in the distributed test architecture and this can introduce additional controllability and observability problems. While there has been interest in test generation algorithms that overcome controllability and observability problems, such algorithms lack generality since controllability and observability problems cannot always be overcome. In addition, traditionally only deterministic systems and models have been considered despite distributed systems often being non-deterministic. This paper describes recent work that characterized the power of testing in the distributed test architecture in the context of testing from a deterministic finite state machine and also work that investigated testing from a non-deterministic finite state machine and testing from an input output transition system. This work has the potential to lead to more general test generation algorithms for the distributed test architecture.

1 Introduction

Many approaches to model based testing use either finite state machines (FSMs) or labelled transition systems (LTSs). There has thus been much interest in automating testing on the basis of FSM or LTS models. Some systems have physically distributed interfaces, called ports, and in testing we place a *local tester* at each port. Each local tester applies a test script and observes the interactions at its port: the sequence of inputs and outputs observed at a port is called a *local observation*. If the local testers cannot communicate with one another and there is no global clock then we are testing in the distributed test architecture, which has been formalized by ISO [7].

The use of the distributed test architecture leads to additional problems in testing that have been called controllability problems and observability problems. Controllability problems occur when a tester does not know when to apply an input as a result of it not being able to observe events at other ports. For example, if a test case starts with input x_p at port p, this should lead to output y_p at p only and this is to be followed by input x_q at port $q \neq p$ then the tester at q cannot know when to apply x_q. This is because it does not observe either the input or output from the previous operation and so cannot know whether x_p has been applied. This is illustrated in Figure 1.

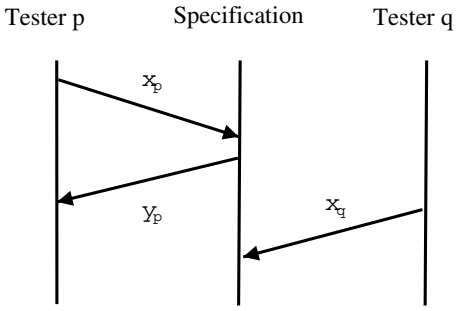

Figure 1. A controllability problem

Observability problems refer to fault masking that can occur as a result of each local tester only being able to observe its own interface. Let us suppose, for example, that the local tester at port p is to apply input x_p, output y_p at p and y_q at $q \neq p$ should be produced by the system under test (SUT), x_p is then input again and the expected output is just y_p at p. Then each local tester will observe the expected sequence of inputs and outputs if the SUT produced y_p only in response to the first input and both y_p and y_q in response to the second input since in each case the local tester at p observes $x_p y_p x_p y_p$ and the local tester at q observes y_q. Fault masking has occurred in this situation, which is illustrated in Figure 2. Interestingly, this example shows that test effectiveness is not monotonic in the distributed test architecture: the test sequence x_p leads to a failure being observed but if

1550-6002/08 $25.00 © 2008 IEEE
DOI 10.1109/QSIC.2008.11

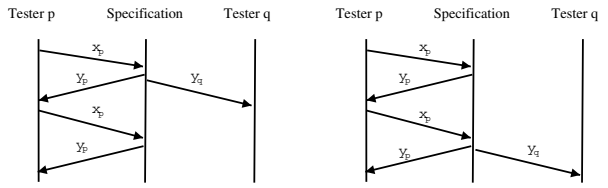

Figure 2. An observability problem

we extend this to $x_p x_p$ we no longer observe a failure.

The problem of testing in the distributed test architecture was originally studied by the protocol conformance testing community and here the focus was on testing from a deterministic FSM (DFSM). However, distributed systems are often nondeterministic and so it is a little surprising that until recently the problem of testing from a nondeterministic model had not been considered. In addition, the focus has been on overcoming the problems introduced by the distributed test architecture but in general these cannot be overcome, even when testing from a DFSM. There is thus a need for a better understanding of the impact of this test architecture and the development of test generation algorithms that achieve as much as possible given the fundamental restrictions. This paper describes some initial work that aims both to lead to a better understand of the effect of the distributed test architecture on testing and to extend the work to nondeterministic models and implementations.

2 Testing from Deterministic Finite State Machines

The problem of testing in the distributed test architecture was originally investigated in the context of protocol conformance testing [2, 3] and almost all work has been on testing from DFSMs (see for example [10, 11]). Since the controllability and observability problems were identified the focus has been on overcoming these problems, either by choosing appropriate test sequences or by connecting the local testers using an external network through which they can exchange coordination messages [8]. However, deploying an external network can increase the cost of testing and coordination messages cannot always be used if tests have timing constraints. In addition, there may be no test sequence, that is free from controllability and observability problems, that achieves a given test objective.

Recent work has defined the notion of two DFSMs being *locally s-equivalent*: this essentially means that they cannot be distinguished by applying an input sequence that causes no controllability problems if we are only making local observations [5]. Thus, if we only apply test sequences that are free of controllability problems then the notion of s-equivalence captures the power of testing from a DFSM: we can distinguish between a DFSM specification and a DFSM implementation in testing if and only if they are not locally s-equivalent. If we can only distinguish between two DFSMs if they are not locally s-equivalent, the challenge is to produce practical test generation algorithms that recognize this. For example, if we have a fault model Φ that describes possible behaviours of the SUT then we could aim to produce a test suite that distinguishes between the specification M and all elements of Φ that are not locally s-equivalent to M.

Interestingly, it is possible to decide whether two given DFSMs or two states of a DFSM are locally s-equivalent in low order polynomial time [5]. If the states or DFSMs are not locally s-equivalent and there are n states and m ports then the algorithm returns an input sequence of length at most $m(n-1)$ that demonstrates this and can be used in testing in order to distinguish them. Since many DFSM test generation algorithms use sequences that distinguish states, there may be scope to use such sequences in testing from a DFSM in the distributed test architecture.

3 Nondeterministic finite state machines

When testing from a DFSM we can produce a test sequence with no controllability problems by choosing an appropriate path from the initial state of the DFSM. We then use the corresponding input sequence. However, if an FSM M is nondeterministic then there may be several possible paths that can be followed when using a test sequence and we have to check all of these.

It may seem that in order to determine whether a test sequence causes a controllability problem we simply need to check all of the paths that can be triggered by this sequence and determine whether any of these have controllability problems. However, we have a more general problem: a tester at port p must be able to decide when to apply an input on the basis of its observations. Figure 3 gives a situation in which we do not have this property, and so there are controllability problems, but none of the corresponding paths have controllability problems. The problem here is the tester at $q \neq p$ has to apply input x_q after the tester at p has applied $x_p x_p$ but for one allowed response it should send x_q after observing y_q and for the other it should send x_q after $y_q y_q$. Thus, if the tester at q observes y_q it does not know whether to apply input x_q or wait for another y_q.

Interestingly, test sequences that have such controllability problems define a set of input/output sequences that are similar to message sequence charts (MSCs) that have implied MSCs. The existence of such implied MSCs can be decided in low order polynomial time [1]. A problem for future work is adapting the approach of [1] to the problem of deciding whether a test sequence causes controllability problems for an FSM.

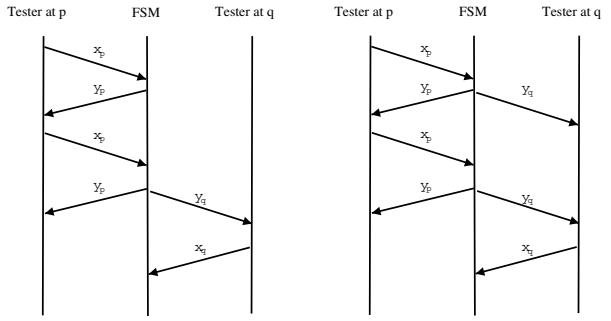

Figure 3. A controllability problem with a nondeterministic model

4 Input output transition systems

An input output transition system (IOTS) is a labelled transition system in which we differentiate between inputs and outputs (see, for example, [9]). An IOTS can be nondeterministic but in addition it may have an infinite state space and there is no need to alternate between input and output. For example, an input could be followed by another input and we might then have an infinite sequence of outputs. IOTSs are thus more general than FSMs. When testing from an IOTS it is normal to use the **ioco** implementation relation that states what it means for an implementation to conform to a specification.

In testing it is usual for the tester to return a verdict: this is pass if no failures are observed but otherwise it is fail. In the distributed test architecture there are multiple local testers and it is natural to have each local tester return a verdict: the overall verdict is pass if and only if all of these verdicts are pass. Let us suppose that we are testing against IOTS s in which input of x_p at port p is either followed by output y_p at p and then output y_q at $q \neq p$ or by output y'_p at p and then output y'_q at q. If testing involves applying input x_p and then observing output, output y_p is observed at p and y'_q is observed at q then each tester sees a local observation that is consistent with s. However, if we bring together these two local observations, $x_p y_p$ and y'_q, we find that no interleavings of these are in the specification and so we know there has been a failure. Thus, testing loses power if we only bring together verdicts made by the local testers and as a result [4] assumes that we can bring together local observations.

When testing from an FSM input and output alternate and so a test run with a finite input sequence must terminate. Once the test has terminated we can simply bring together the local observations. However, this is not the case for an IOTS: we can reach a state from which an infinite sequence of outputs can be produced. It is thus necessary to define the points at which local observations at the different ports can be brought together and in [4] this can be done when the SUT is *quiescent*: when it is in a state where it cannot produce any further output without first receiving input. The approach of [4] is thus to say that a process p conforms to a process s if the following conditions hold, and then we say that p **dioco** s: For every trace σ of p that ends in quiescence, there is a trace σ' of s such that σ and σ' cannot be distinguished if only local observations are made.

5 Equivalent and minimal machines

We often want the models we use to be minimal: there is no smaller equivalent model. The notion of minimality is well understood for FSMs and, for example, there are efficient algorithms for converting a DFSM into an equivalent minimal DFSM [6]. If the DFSM is a design then the hope is that the use of a minimal DFSM will lead to a relatively compact system. Let us suppose that we have a DFSM M that represents a system design and the system will have distributed ports with users or subsystems that are not themselves connected and there is no global clock. As a result, the expected usage of the system corresponds to the distributed test architecture and so we do not require the SUT to be equivalent to M: instead, it is sufficient that the SUT and M cannot be distinguished in the distributed test architecture. As a result, the SUT can be equivalent to a smallest DFSM that cannot be distinguished from M in the distributed test architecture and so we may obtain more compact implementations [5]. There is the challenge of extending this to NFSMs and IOTSs.

6 Conclusions

If the system under test (SUT) has multiple physically distributed interfaces/ports then we place a local tester at each port. If these local testers cannot directly communicate with one another and there is no global clock then we are testing in the distributed test architecture. It has been known for many years that the use of the distributed test architecture can cause problems in testing and in the past the focus has been overcoming these problems when testing from a deterministic finite state machine (DFSM).

Recent work has characterized the power of testing in the distributed test architecture by saying what it means for DFSMs to be indistinguishable in this architecture. The hope is that this will lead to test generation algorithms that produce test sequences that target the corresponding weaker notion of equivalence, aiming to decide whether the SUT and the specification can be distinguished in this architecture. If the use of the SUT corresponds to the constraints imposed by the architecture then we require such test generation algorithms. Otherwise we need a better understanding of the

impact of the distributed test architecture in order to help the tester decide, for example, whether it is worth introducing an external network that connects the local testers.

Until recently, the problem of testing in the distributed test architecture was only investigated for deterministic models and implementations. This appears to be a significant limitation since distributed systems are often nondeterministic. Interestingly, if we look at nondeterministic FSMs (NFSMs) the notion of a test sequence being controllable is quite different from that found with DFSMs. There are additional differences and challenges if we are testing from input output transition systems (IOTSs) since these can have infinite state spaces and input and output need not alternate.

There are several possible avenues for future work. The equivalences imposed on NFSMs and IOTSs when we use the distributed test architecture require additional research. Once a good understanding of such equivalences is obtained there is the challenge of defining test generation algorithms that return test sequences that target these notions of equivalence. In addition, since we have a different notion of equivalence we also have a new definition of minimality and there is the problem of minimizing models.

References

[1] R. Alur, K. Etessami, and M. Yannakakis. Inference of message sequence charts. *IEEE Transactions on Software Engineering*, 29(7):623–633, 2003.

[2] R. Dssouli and G. von Bochmann. Error detection with multiple observers. In *Protocol Specification, Testing and Verification V*, pages 483–494. Elsevier Science (North Holland), 1985.

[3] R. Dssouli and G. von Bochmann. Conformance testing with multiple observers. In *Protocol Specification, Testing and Verification VI*, pages 217–229. Elsevier Science (North Holland), 1986.

[4] R. M. Hierons, M. G. Merayo, and M. Nunez. Implementation relations for the distributed test architecture. In *20th IFIP International Conference on Testing of Communicating Systems (TESTCOM 2008)*, Springer Lecture Notes in Computer Science, page to appear. Springer, 2008.

[5] R. M. Hierons and H. Ural. The effect of the distributed test architecture on the power of testing. *The Computer Journal*, to appear.

[6] J. E. Hopcroft. An n log n algorithm for minimizing the states in a finite automaton. In Z. Kohavi, editor, *The theory of Machines and Computation*, pages 189–196. Academic Press, 1971.

[7] J. T. C. ISO/IEC JTC 1. *International Standard ISO/IEC 9646-1. Information Technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 1: General concepts*. ISO/IEC, 1994.

[8] O. Rafiq and L. Cacciari. Coordination algorithm for distributed testing. *The Journal of Supercomputing*, 24(2):203–211, 2003.

[9] J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software - Concepts and Tools*, 17(3):103–120, 1996.

[10] H. Ural and C. Williams. Constructing checking sequences for distributed testing. *Formal Aspects of Computing*, 18(1):84–101, 2006.

[11] Y. C. Young and K. C. Tai. Observational inaccuracy in conformance testing with multiple testers. In *IEEE 1st workshop on application-specific software engineering and technology*, pages 80–85, 1998.