(11) Proposal fortest4.ps

FORmal methods and TESTing (FORTEST)

Dr R. M. Hierons, Proposal Coordinator

1 Part One: Previous Track Record

Note: URLs and more details may be found in the appendix.

1.1 Academic Partners

1.1.1 Brunel University

Investigators: Dr Rob Hierons (PI) and Dr Mark Harman.

Rob Hierons has produced a number of results that concern testing from a formal specification or model. Hierons is on the editorial board of the Journal of Software Testing, Verification, and Reliability (STVR) and is to be a guest editor, with John Derrick, of a special issue on specification based testing. Mark Harman is interested in program slicing and edited the 1998 slicing special issue of the Journal Information and Software Technology. Harman has looked at ways in which slicing can assist in testing. Harman and Hierons' work has been supported by the EPSRC.

1.1.2 University of Kent at Canterbury

Investigators: Dr John Derrick and Professor Keith Mander.

John Derrick is a Reader in Formal Methods. His current interests include specification and design techniques, refinement and testing. His work has been supported by the EPSRC, the DTI, BT, Nortel, Eurocontrol, the Royal Society, and the London Mathematical Society. Keith Mander's recent research interests have been in the relationship between formal and non-formal notations for specification and testing and in test-data generation. He is interested in the relationship between static and dynamic techniques (for example, proof and test) and in the relationship between development techniques and the business process. His work has been supported by the EPSRC, Rolls Royce and Sun Microsystems.

1.1.3 University of Liverpool

Investigator: Dr Martin Woodward

Martin Woodward has long-standing research interests in software testing and formal methods. His contributions in software testing include: work on path analysis and coverage; the integration of weak mutation and data flow testing; empirical studies of testing techniques; the introduction of firm mutation; chart-based ways of combining software construction and testing; and studies of software testability. His work in the area of software specification has focused particularly on algebraic specifications. He is the Editor-in-chief of the journal *Software Testing, Verification and Reliability.*

1.1.4 Oxford University

Investigator: Professor Jim Woodcock

Jim Woodcock is known for his research and consultancy in the theory and practice of software engineering. He has developed a theory of information flow; theories of sequential and concurrent refinement; combinations of state-based and behavioural specifications; a theory of safety for solid-state interlocking railway signalling; and is contributing to the work on unifying theories of programming. He was a major contributor to the ISO and BSI Standard for the Z Notation.

His research team won the Queen's Award for Technological Achievement in 1992 for its work with IBM UK Laboratories. He was technical consultant to the National Westminster Bank and Logica on a project that, in 1999, produced the first product certified to ITSEC Level E6, the most stringent requirements for the assurance of a secure system. The product was a smart card for electronic finance, and its development required the extension of existing theories of refinement. He is the Director of Oxford University's Software Engineering Programme and is a member of the Scientific Committee for the Smith Institute for Industrial Mathematics and System Engineering.

1.1.5 Sheffield University

Investigators: Professor Mike Holcombe, Dr Kirill Bogdanov, Dr Anthony Simons, Eur Ing Dr Tony Cowling, and Dr. Marian Gheorghe.

All investigators are members of the Verification and Testing group, with an interest in the application of X-machine models to testing. Mike Holcombe is the head of the group, and his work focuses on the theory of testing, models of computing, intelligent agents and concurrent systems. He has held a number of SERC and EPSRC grants including /1/1031 *Functional testing of High Integrity VLSI.*

Kirill Bogdanov adapted the X-machine testing approach to statecharts. Anthony Simons works on formal aspects of object-oriented languages and design methods. He is principal investigator on EPSRC project GR/M56777 *Method for Object Testing, Integration and Verification (MOTIVE).* Marian Gheorghe and Tony Cowling work with Mike Holcombe on testing from communicating systems of X-machines. They are also interested in the formal foundations of models of parallel and distributed systems.

1.1.6 York University

Investigators: John Clark and (Dr) Steve King. Author: Simon Burton.

John Clark leads the testing work at York aimed largely at high integrity systems. Previously he worked for Logica in secure systems design and evaluation and still provides formal methods consultancy for secure systems applications. Steve King specialises in formal specification and refinement techniques. He is Deputy Convenor of the ISO Standardisation Panel and previously worked at the Oxford University Programming Research Group on the IBM CICS project. Simon Burton is a Research Associate funded by Rolls Royce. Recent research concerns automated test generation and bridging graphical and formal notations. He has extensive industrial experience of telecommunications software (and testing tools development).

1.2 Industrial Partners

1.2.1 DaimlerChrysler, Berlin

Investigator: Dr Harbhajan Singh.

Harbhajan Singh is Head of System Integration within the department of Methods and Tools in the Software Technology Lab. He previously worked in the areas of simulation and the testing of very large scale integrated circuits. His present interests include the specification and test of high integrity software. He has worked with Dr Hierons and Professor Holcombe in testing based on formal specifications.

1.2.2 DERA

Investigator: Colin O'Halloran DPhil BSc – DERA Fellow.

Colin O'Halloran's principal role has been: research into mathematical techniques to assess critical systems; the creation of a DERA capability for the assessment of critical computer systems; the application of this capability and exploitation of the research to MoD procurement. He is an international authority on safety-critical computers and a visiting fellow of Kellogg College at Oxford University. He is currently working on the development and application of novel analysis techniques for Eurofighter's flight control system and the use of model checking to direct testing. He was the sole UK representative on the Ariane 5 board of inquiry, showing that the cause of the failure was located within a particular fragment of code within the Inertial Reference System.

1.2.3 Philips Research Laboratories

Investigator: Dr. Paul Krause

The Software Engineering and Applications Group at Philips Research Laboratories has extensive experience in specification based testing. A particular focus has been the automatic generation of tests from state models. Their particular strength is in the application of these and other techniques to real industrial projects in a wide range of domains. Paul Krause has thirteen years experience in formal methods in software engineering and artificial intelligence. His current research interests are domain modelling and requirements engineering for product families, test case generation from UML specifications and software reliability estimation.

1.2.4 Praxis Critical Systems Ltd

Investigator: Eur Ing Keith Harrison

Keith Harrison was responsible for the testing of a variety of safety critical control software projects at Lucas Aerospace. Post Lucas and prior to Joining Praxis Critical Systems earlier this year Keith spent five years involved in the static analysis of the safety critical software on the Hercules aircraft. Keith's interest is in the combination of testing and more formal analysis to reduce the overall cost of verification and validation while maintaining a high level integrity.

1.2.5 Telelogic

Investigator: Dr Jeremy Dick, DPhil DIC MA Bsc(Eng) ACGI

Involved in academic and industrial research into mathematically formal methods for over 18 years, Jeremy has had exposure to a number of formal notations. More recent published research work concerns a tool for the semi-automatic generation of test cases from a VDM specification. Jeremy now works for QSS, part of Telelogic, who have industrial testing products based on TTCN. He also teaches a course on Software Testing as part of the Software Engineering Programme, External Studies, University of Oxford. Jeremy is keen on exploiting more formal approaches to testing in both these contexts.

2 Part Two: Proposed Research and its Context

2.1 Project Overview

With the growing significance of computer systems within industry and wider society, techniques that assist in the production of reliable software are becoming increasingly important. The complexity of many computer systems requires the application of a battery of such techniques. Two of the most promising approaches are formal methods and software testing. FORTEST is a cross-community network that will bring together expertise from each of these two fields.

Traditionally formal methods and software testing have been seen as rivals. Thus, they largely failed to inform one another and there was very little interaction between the two communities. In recent years, however, a new consensus has developed. Under this consensus, these approaches are seen as complementary ([14]). This opens up the prospect of collaboration between individuals and groups in these fields.

While there has already been some work on generating tests from formal specifications and models, FORTEST will consider a much wider range of ways in which these fields might interact. In particular, it will consider relationships between *static testing* (verification that does not involve the execution of the implementation) and *dynamic testing* (executing the implementation).

FORTEST will build a new community that will explore ways in which formal methods and software testing complement. This will allow these fields to inform one another in a systematic and effective manner and thus facilitate the development of new approaches and techniques that assist the production of high quality software. The significance of this topic and the lack of any large UK groups, working on links between testing and formal methods, make it vital that such a network is established in the near future.

This proposal is timely because of the recent increased national and international interest in this subject which makes the formation of a community both feasible and desirable. It is anticipated that the existence of this network will lead to further collaboration and thus to a significant increase in the quality and quantity of research produced in this area.

The main aims of FORTEST are

- to bring together academics and industrialists interested in formal methods and software testing;
- to stimulate collaboration between individuals and groups in these fields;
- to disseminate problems and results to researchers and practitioners in these two fields and to the wider Software Engineering community.

The dissemination of this information will lead to a greater awareness of the links between software testing and formal methods. It will also widen the use of methodologies that assist the development of reliable software.

The network will focus on the following problems.

- How can the relationships between formal methods and software testing be utilised?
- How can the software development process be adapted in ways that simplify the utilisation of these relationships?
- How can techniques, developed to utilise the relationships between formal methods and testing, be automated?

The members of FORTEST satisfy all of the main criteria for a network on formal methods and software testing. It contains members of the formal methods, software testing, and program analysis fields. The academics have verification and validation experience within a number of formalisms and in a number of domains (such as aerospace, security, and telecommunications). The industrial partners bring a variety of problem domains, experience in applying formal methods, and real case studies. The industrial partners will apply, and thus evaluate, ideas that result from this network, feeding back their experiences.

As well as covering a variety of areas, the members of FORTEST have similarities in interests which should facilitate collaboration. For example, researchers from Brunel University, York University, and DaimlerChrysler have been investigating testing from hybrid state-based/model-based specifications. All members of the network are interested in the relationship between testing, model checking and proof. John Derrick, Jim Woodcock and

Mike Holcombe have looked at testing, refinement and the relationship between them. Anthony Cowling is interested in specification and testing of communicating and distributed systems. Mike Holcombe and Kirill Bogdanov have worked on testability, scalability, hierarchy and management of the testing process as well as automatic test set generation. Anthony Simons has looked at type-safety and checking algorithms. The latter three have a shared interest in formally-guaranteed complete functional testing. Keith Mander and John Clark are interested in using testing in order to derive counterexamples.

Brunel University is the centre for the EPSRC project SEMINAL (GR/M78083). SEMINAL (*Software Engineering with Metaheuristic Innovative Algorithms*) provides for the establishment of a research network for which Brunel is the principal institution. The common interest in testing will allow FORTEST and SEMINAL to strengthen one another (testing is arguably the area of Software Engineering that has benefited most from the application of Metaheuristic algorithms). The two networks being centred at the same institution will allow them to be run more effectively and efficiently.

The network will be strengthened by the experience of Telelogic and York University (through CadiZ) in the development and use of tools to support formal methods. Keith Mander and John Clark have experience in software metrics and statistics that will assist the design and evaluation of empirical studies.

2.2 Background

The use of a formal specification or model eliminates ambiguity and thus reduces the chance of errors being introduced during software development. Where a formal specification exists, both the source code and the specification may be seen as formal objects that can be analysed and manipulated. The use of a formal specification thus introduces the possibility of the formal and, potentially, automatic analysis of the relationship between the specification and the source code. This is often assumed to take the form of a proof, but such a proof cannot guarantee operational correctness. For this reason, even where such a proof exists, it is important to apply dynamic testing ([8]).

Testing may be seen as any process that provides information that might either detect faults or provide confidence in the implementation under test (IUT). Thus, as noted earlier, there are both static and dynamic test techniques. Each form of testing provides some information about the IUT.

Static and dynamic testing have very different characteristics and so provide very different types of information. Static testing relies upon some underlying model that describes the link between the source code and the behaviour exhibited by the IUT. This model might, for example, be the semantics of the programming language and in this case any analysis relies upon the correctness of the compiler and hardware. Static testing may provide general information about a model of the system, but cannot be applied directly to the IUT. Dynamic testing, in contrast, provides specific information about the IUT. One challenge is thus to combine these two forms of testing in order to provide general information about the IUT.

2.2.1 Formal Methods and Dynamic Testing Complement

Software testing is an important and, traditionally, extremely expensive part of the software development process. Studies suggest that testing often forms in the order of fifty percent of the total development cost ([3]). Where formal specifications and models exist, these may be used as the basis for automating parts of the testing process ([1, 2, 19, 6, 21, 20, 10, 11, 5, 16, 22, 13]). This may lead to more efficient and effective testing. It may thus transpire that the automation of parts of the software testing process is one of the most significant benefits of using a formal specification language. The links between testing and formal methods do, however, go well beyond generating tests from a formal specification.

The presence of a formal specification or model makes it possible for the tester to be clearer about what it means for a system to pass a test. This may be achieved through the use of test hypotheses ([9]) or design for test conditions ([17, 18, 15]). Similar ideas may be found in the generation of checking experiments from finite state machines ([4, 23, 12]). Using these approaches it is possible to generate tests that determine correctness under certain well understood conditions circumventing Dijkstra's famous aphorism that testing can show the presence of bugs, but never their absence ([7]). Program analysis might be used in order to either prove that these conditions hold or to provide confidence in these conditions holding.

Information gathered by dynamic testing may assist when using a formal specification. Testing may be used in order to provide initial confidence in a system before effort is expended in attempting to prove correctness.

Where it is not cost effective to produce a proof of conformance, the developers may gain confidence in the implementation through systematic testing. This might be complemented by proofs that certain critical properties hold. A proof of correctness might also use information derived during testing. Finally, a proof of correctness relies upon a model of the underlying system and dynamic testing might be used to check this model. An interesting challenge is to generate tests that are likely to be effective in detecting errors in the assumptions inherent in a proof of correctness.

2.3 Programme and Methodology

2.3.1 Goals

FORTEST is a three year project that will develop a new community in order to investigate ways in which the relationships between formal methods and software testing may be exploited. It will raise the awareness of the links between these fields and disseminate new methods and techniques developed by the community. The main goals are thus

1. to develop a community;
2. to disseminate current results and problems and new results produced by this community.

The deliverables of FORTEST are

1. a workshop to be held approximately six months after the start of the project in order to establish an initial community and to formalise the set of problems;
2. a workshop approximately eighteen months after the start of the project in order to widen the range of problems considered and approaches used;
3. a workshop in the last six months of the project to disseminate the results;
4. research meetings, that are internal to the network, to be held quarterly (except where they would coincide with a workshop);
5. a landscape document, that outlines the current state of the art, to be completed within a year of the start of the project;
6. a horizons document, that describes the main results of the project and potential future work, to be produced in the last six months of the project;
7. an internet structure that includes a mailing list and a web site containing information about the project.

The members of the network intend to secure funding in order to continue the network beyond the three years. This funding might come from further grants and from industry. The team will also seek to broaden involvement by inviting other experts, in the fields of testing and formal methods, to join FORTEST as the network progresses.

2.3.2 Programme of Work

The following will be the major problems considered in this project.

1. How can the relationships between formal methods and software testing be exploited?
2. How can elements of the software development process be changed in ways that simplify the utilisation of these relationships?
3. How can techniques, developed to utilise the relationships between formal methods and testing, be automated?

The work will be supported by two types of research meetings and three workshops. The workshops will be advertised to the wider community. The research meetings will include regular quarterly meetings and specialised meetings that will consider particular problems. Representatives from each site will be invited to the regular meetings. While the specialised meetings need not have representatives from each site, they will report to the organising committee. The workshops will help disseminate the results of FORTEST as well as widen the range of influences.

2.4 Project Management

FORTEST will be managed by a central organising committee. The coordinator, who is the principal investigator, will be responsible for organising and chairing the meetings of this committee. The coordinator will also be responsible for managing the electronic facilities. The coordinator will be assisted in this by an administrator and a systems administrator (both part-time).

The organising committee will meet quarterly. These meetings will coincide with the regular research meetings, the organising committee meeting in the morning and the research meeting occurring in the afternoon. These meetings will be held at the sites of UK based collaborators.

Electronic communication within the network will be used to encourage discussion. The web site will also act as a central source of information. The web site and mailing group shall be maintained by the coordinator's site.

2.5 Relevance to Beneficiaries

There will be several groups of beneficiaries. Initially the main beneficiaries will be researchers, both within academia and within industrial research groups. Researchers from the formal methods and software testing communities will gain both from the development of new techniques and methodologies and by exploiting links between these fields. Each community will enrich and strengthen the other.

In the longer term industrialists, and society in general, will gain from the introduction of new techniques and methodologies that assist in the development of high quality software.

2.6 Dissemination and Exploitation

Three workshops and nine regular research meetings will be held during the project. FORTEST will be inclusive rather than exclusive and thus the workshops will be advertised widely. The first workshop, which will be scheduled around six months into the project, will develop an initial community and feed into the production of a landscapes document.

A workshop will be held approximately eighteen months into the project. This will review the progress, refine the goals, and disseminate results and problems.

The third workshop will be held during the last six months of FORTEST and shall bring together the results of the project and facilitate their dissemination. This dissemination shall be assisted by a horizons document, that will describe the main results from the project, and papers written by individuals.

The members of FORTEST shall attend international conferences, presenting results from the project. The most suitable conferences are Formal Methods Europe (FME); The International Symposium on Software Testing and Analysis (ISSTA); The International Conference on Software Engineering (ICSE); The International Conference on Testing of Communicating Systems (TestCom); and Formal Description Techniques for Distributed Systems and Communication Protocols / Protocol Specification, Testing and Verification (FORTE/PSTV).

A central web site shall provide a bibliography and electronic versions of articles and reports written by members of the network as part of the project. It shall also contain links to relevant sites, a Who's Who, a calendar, an overview of the main project activities and a list of open problems. A mailing group shall assist communications within the community and thus aid dissemination.

The industrial partners will exploit the results, feeding these into their development process and their development tools. The results will feed into the ongoing research of all involved.

2.7 Justification of Resources

Funding will be required in order to cover the cost of workshops, travel, administrative support and the development and maintenance of the electronic facilities. The larger companies will cover their own travel expenses. The funds applied for will cover the following.

- Workshops and Research Meetings. The travel and accommodation costs of the representatives of the universities and the SMEs.
- Project coordination. The required administrative resources.
- Electronic facilities. A part-time systems administrator to set-up and maintain the electronic facilities.
- Conference Attendance. The attendance, of representatives of the universities and the SMEs, at international conferences.

The potential impact of FORTEST makes it excellent value for money.

References

[1] A. V. Aho, A. T. Dahbura, D. Lee, and M. U. Uyar. An optimization technique for protocol conformance test generation based on UIO sequences and Rural Chinese Postman Tours. In *Protocol Specification, Testing, and Verification VIII*, pages 75–86, Atlantic City, 1988. Elsevier (North-Holland).

[2] N. Amla and P.Ammann. Using Z specifications in category partition testing. In *COMPASS '92, Seventh Annual Conference on Computer Assurance*, pages 15–18, Gaithersburg, MD, USA, 1992.

[3] B. W. Boehm. *Software Engineering Economics.* Prentice-Hall, 1981.

[4] T. S. Chow. Testing software design modelled by finite state machines. *IEEE Transactions on Software Engineering*, 4:178–187, 1978.

[5] J. Derrick and E. Boiten. Testing refinements of state-based formal specifications. *Journal of Software Testing, Verification, and Reliability*, 9:27–50, 1999.

[6] J. Dick and A. Faivre. Automating the generation and sequencing of test cases from model-based specifications. In *FME '93, First International Symposium on Formal Methods in Europe*, pages 268–284, Odense, Denmark, 19-23 April 1993. Springer-Verlag, Lecture Notes in Computer Science 670.

[7] E. W. Dijkstra. Notes of structured programming. In O. J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, editors, *Structured Programming.* Academic Press, 1972.

[8] J. H. Fetzer. Program verification: The very idea. *Communications of The ACM*, 31:1048–1063, 1988.

[9] M. C. Gaudel. Testing can be formal too. In *TAPSOFT'95*, pages 82–96. Springer-Verlag, March 1995.

[10] R. M. Hierons. Testing from a finite state machine: Extending invertibility to sequences. *The Computer Journal*, 40:220–230, 1997.

[11] R. M. Hierons. Testing from a Z specification. *Journal of Software Testing, Verification and Reliability*, 7:19–33, 1997.

[12] R. M. Hierons. Adaptive testing of a deterministic implementation against a nondetermistic finite state machine. *The Computer Journal*, 41:349–355, 1998.

[13] R. M. Hierons, S. Sadeghipour, and H. Singh. Testing a system specified using Statecharts and Z. *Information and Software Technology*, to appear, 2001.

[14] C. A. R. Hoare. How did software get so reliable without proof? In *Proceedings of Formal Methods Europe, 96 (Lecture Notes in Computer Science 1051)*, pages 1–17. Springer-Verlag, 1996.

[15] M. Holcombe and F. Ipate. *Correct Systems: Building a Business Process Solution.* Springer-Verlag, 1998.

[16] Hyoung Seok Hong, Young Gon Kim, Sung Deok Cha, Doo Hwan Bae, and Hasan Ural. A test sequence selection method for statecharts. *Journal of Software Testing, Verification and Reliability*, 10, 2000.

[17] F. Ipate and M. Holcombe. An integration testing method that is proved to find all faults. *International Journal of Computer Mathematics*, 63:159–178, 1997.

[18] F. Ipate and M. Holcombe. A method for refining and testing generalised machine specifications. *International Journal of Computer Mathematics*, 68:197–219, 1998.

[19] G. Laycock. Formal specification and testing: A case study. *Journal of Software Testing, Verification and Reliability*, 2:7–23, 1992.

[20] H. Singh, M. Conrad, and S. Sadeghipour. Test case design based on Z and the classification-tree method. In *First IEEE Conference on Formal Engineering Methods*, pages 81–90, Hiroshima, Japan, November 1997. IEEE Computer Society.

[21] P. Stocks and D. Carrington. A Framework for Specification-Based Testing. *IEEE Transactions on Software Engineering*, 2:777–793, 1996.

[22] H. Ural, K. Saleh, and A. Williams. Test generation based on control and data dependencies within system specifications in SDL. *Computer Communications*, 23:609–627, 2000.

[23] H. Ural, X. Wu, and F. Zhang. On minimizing the lengths of checking sequences. *IEEE Transactions on Computers*, 46:93–99, 1997.