

2nd Year DPhil Report

Nigel J. Tracey

31, March 1998

Contents

1	Introduction	2
2	Thesis	4
2.1	Introduction	4
2.2	Current Shortcomings	4
2.2.1	Automated Functional Testing	5
2.2.2	Automated Structural Testing	5
2.3	Generality	5
2.4	Thesis Abstract	6
2.4.1	Introduction	6
2.4.2	Review	6
2.4.3	Optimisation for Test-Data Generation	7
2.4.4	Framework Applications	8
2.4.5	Evaluation	8
3	Progress	9
3.1	Introduction	9
3.2	Research Work	9
3.3	Development Work	9
3.4	External Presentations and Publications	10
4	Plans	12
4.1	Introduction	12
4.2	Thesis Writing	12
4.2.1	The Review	12
4.2.2	Optimisation for Test-Data Generation	12
4.2.3	Framework Applications	13
4.2.4	Evaluation	13

Chapter 1

Introduction

Software testing is an expensive process, typically consuming at least 50% of the total costs involved in developing software [Bei90]. Few programmers like testing and even fewer like test specification and design. It remains, however, the primary method through which confidence in software is achieved both by the development team and the customer [DO91]. Automation of testing is desirable both to reduce development costs and also to improve the quality of (or at least confidence in) software. While automation of the testing process – the maintenance and execution of test-cases – is taking hold commercially, the automation of test-data generation is significantly less advanced. It has been suggested by Ould that it is this area that is the most important aspect of testing requiring automation [Oul91].

Optimisation techniques are search techniques which aim to find minimal (or maximal) values of a particular cost (or objective) function. They are flexible and powerful, with the ability to obtain good results for many extremely difficult problems. Their flexibility is due to the fact that the search is simply directed by a cost function – very little other problem specific knowledge is required.

The aim of the work presented in this report is to investigate the use of optimisation techniques in the automation of software test-case data generation for both functional and non-functional criteria. To maintain a coherent focus, and to give criteria by which to judge success, the following hypothesis will form the central thread to the work.

”Non-linear optimisation algorithms are very useful for solving a wide range of complex problems. They can usefully be applied in the derivation of test cases for both functional and non-functional properties of computer systems, to improve the current state-of-the-art in testing. The use of non-linear optimisation techniques will allow generality, both in terms of the number of testing problems which can be addressed and the units which can be tested”

This report is divided into four chapters, the first being this brief introduction and *road-map*. The second chapter further outlines the central ideas of the thesis. This highlights areas where an original contribution is being made to the field and also provides an extended abstract for the final thesis. The third chapter presents the work and research achievements to date. It places this work within the framework of the thesis and also details external presentations

CHAPTER 1. INTRODUCTION

of this research. The final chapter lays down the route from the current state of work to a completed thesis. This details the research work still to be carried out along with target deadlines. A chapter-by-chapter timetable for completion of the thesis is also given.

Chapter 2

Thesis

2.1 Introduction

This chapter details the central idea of this research and explains the rationale and originality behind it. Software testing is often divided into the testing of functional and non-functional properties. The testing of functional properties is concerned with ensuring that the functionality of the software is as specified. In comparison testing non-functional properties is concerned with ensuring that the constraint imposed by process, product or external requirements are met [Som92]. Research into testing non-functional properties is limited, with very little work on automation of the testing process [Spa90]. The current industrial state-of-practice appears to simply reuse functional tests (which in practice is little better than random testing).

In many respects this distinction is artificial and has been unhelpful in the development of testing methodologies and testing automation. The primary reason for testing is to gain confidence that the software will perform as expected and a deviation from a timing or resource requirement can be just as critical as incorrect logical operation.

The remainder of this chapter will discuss the failings of current research approaches to test-case and test-data automation and explain how some of these can be overcome. The chapter will conclude with an extended abstract for the thesis.

2.2 Current Shortcomings

The predominant focus of the research into the automation of software test-data generation has been for testing functional properties of software. The testing of functional properties can be achieved using functional testing (*black-box*) or structural testing (*white-box*). In functional testing, the inputs and outputs for a test-case are derived solely from the software unit's specification. In general, automation of this derivation is only possible when formalised notations are used to capture the functional specification. Structural testing methods use knowledge of the software unit's implementation to systematically derive the test input data and then measure the test-coverage (or completeness) [Som92].

2.2.1 Automated Functional Testing

Random test generators are the simplest form of black-box test automation as the test-data is generated randomly without reference to the source code. Other black-box test automation has been carried out using syntax-based testing [Bur67], state-based specifications [FvBK⁺91, Bur96] and formal-methods [Yan95, Meu97]. Perhaps the biggest problem with test-cases generated from high-level specifications is how to map the tests to the software implementation for execution. Also the test-case generation methods are tied very closely to the specification notation and the testing problem being addressed, they therefore lack general applicability. Indeed many of the methods presented in the literature can only process subsets of the specification notation being used further reducing the general applicability.

2.2.2 Automated Structural Testing

Many techniques for generating test-data for functional properties have been proposed [BEL75, Cla76, DO91, Kor96, JSE96, Kin76, Kor90, Kor96, MS76, OP96, Spa90, Wat95]. These can broadly be classified into static and dynamic methods.

Static Test-Data Generation

Static techniques do not require the software under test to be executed. They generally use symbolic execution to obtain constraints on input for a particular test criterion. Solutions to these constraints represent the test-data. Many of the limitations of the static techniques come from their use of symbolic execution. It is difficult to analyse recursion, dynamic data-structures, array indices which depend on input data and some loop structures using symbolic execution. Also, the problem of solving arbitrary constraint systems is provably intractable.

Dynamic Test-Data Generation

The use of dynamic techniques is based on the contention that test-data generation is best formulated as a numerical maximisation (or minimisation) problem. Dynamic techniques generally involve a directed search for test-data which meets a desired criterion executing the software for each step of the search.

The application of many of the existing dynamic techniques is limited by the lack of generality. This lack of generality can be seen in the limited data-types or control-flow structures which can be processed and also the limited testing criteria addressed. It is also the case that for complex software with large parameter spaces simple directed search techniques will fail in sub-optimal solutions (see [Tra97]) or will be computationally intractable.

2.3 Generality

The aim of this research is to attempt to overcome the limitations of current test-data generation approaches. This will be achieved by developing a unified framework for test-data generation for both black and white-box testing of functional properties and also non-functional properties. The generality of

the framework will come from the use of general-purpose optimisation techniques. Optimisation techniques are a flexible and powerful search method with an ability to find good results for many extremely difficult problems.

Within this test-data generation framework new testing problems can be addressed simply by devising an *objective-function* to guide the optimisation-based search to the desired solution. This framework will also be extended to aid with high-level test-case management issues such as optimal test-subset selection for multiple criteria (i.e. cost, effort, adequacy, etc . . .) and regression testing.

2.4 Thesis Abstract

This section gives a proposed extended abstract for the thesis. To allow this to stand-alone as a thesis abstract it may duplicate some of the material presented elsewhere in this document.

2.4.1 Introduction

Software testing is an expensive process, typically consuming at least 50% of the total costs involved in developing software systems [Bei90]. However, it remains the primary means for obtaining confidence in software systems. Automation of testing is desirable both to reduce development costs and also to improve the quality of (or at least confidence in) software. Ould [Oul91] has suggested that it is the process of test-case data generation which, if automated, would give the biggest beneficial effect. This thesis is concerned with the development of a unified test-case data generation framework. This framework will be able to generate test-data for a wide variety of testing problems and a wide range of software systems. Its principal aims are generality (both to testing problems and testable units), efficiency (obtaining test-data significantly quicker and therefore cheaper than manual techniques) and quality (obtaining high quality test-data which allows an improvement in the quality of the resulting software).

2.4.2 Review

A review of related literature is presented. This begins with an examination of existing methods for test-case data generation automation. This is followed by details of the current approaches to automation of test-case management issues. Finally the non-linear global optimisation literature is reviewed.

Testing Functional Properties

The testing of functional properties can be divided into two categories; functional (*black-box*) testing and structural (*white-box*) testing. While many techniques exist for test-data generation in the research community, they are rarely used in industry. Details of existing approaches to automating the testing of functional properties lead to the conclusion that the biggest problem is a lack of generality. This lack of generality is evident in the limitation on the data-types, control-flow structures and testing problems which each of the existing methods can address. There is also a large issue of scalability. Each method is only

shown to work on small toy software. No evidence is presented that any of the methods would scale to industrial software examples.

Testing Non-Functional Properties

The vast majority of research into non-functional properties has focused on static analysis techniques and not testing. Hence testing of non-functional properties is the poor relation of function testing research. There is virtually no literature on automation of test-data generation for non-functional testing problems.

Test-Case Management Issues

In terms of automation a body of work exists in the field of regression test-set selection. However, again there is little evidence of the industrial strength of these techniques. Also generality of the approaches is an issue. It is unclear how the current approaches could be broadened into a general framework which would allow other testing problems to be addressed, or multiple criteria to be optimised in the test-subset selection.

Heuristic Global Optimisation Techniques

Heuristic global optimisation techniques show themselves to be a powerful and flexible approach to solving many difficult problems. Three optimisation techniques are presented - simulated annealing, tabu-search and genetic algorithms. For each technique a presentation of its problem specific requirements, enhancements and its power in coping with large complex search spaces is given.

2.4.3 Optimisation for Test-Data Generation

To allow power and flexibility of optimisation techniques to be exploited it is necessary to show how test-data generation problems can be expressed in terms of an optimisation problem. For many software testing criteria there is the concept of a *good* test-case – that is a test-case which is more likely to distinguish a correct program from an incorrect program (for example for branch coverage, test-cases on the boundary points of often considered better than those in the middle of domains. Generally it is the software tester’s responsibility to find the best-test-case (or test-set) within particular constraints (usually time or financial constraints). Searching for near optimal solutions in a difficult search space is what optimisation techniques are designed to do. However, despite this important similarity, there are a number of questions to be answered before optimisation techniques can be used in automating test-case data generation. These fall into two classes – those which are specific to the optimisation technique and those which are specific to the testing problems. Those in the first class are mainly concerned with how to represent candidate solutions and how to generate new ones. Test-case data is obviously most naturally represented by a collection of data values which forms the input to the software-under-test. The generation of new candidate solutions is dependent on the optimisation technique, but for simulated annealing the concept of a neighbourhood is used. This aims to capture the concept that some candidate solutions can be considered close together or in the neighbourhood of each other. This can be modelled

using some proportion of given data-types. More details of methods for address test-case data generation using optimisation techniques can be found in [Tra97]. The test problem specific optimisation decisions will be discussed along with applications of the framework below.

It is through the use of optimisation techniques for test-case data generation that we hope to overcome the short-comings of previous approaches. Optimisation techniques make no few assumptions about the problem they are trying to solve, they are simply guided by the cost-function. This allows generality over both the testing problems which can be addressed and also the constructs which can be tested. New testing problems simply require new cost-functions. The fact that the data generation stage is dynamic means we have access to all information that the software does and hence we are not limited in the structures that can be processed in the same way as static techniques (for example the problem with recursion, arrays, etc . . . and symbolic execution). The ability of non-linear heuristic optimisation techniques also increase the ability of the system to generate high quality test-case data even when faced with very large, and possibly highly non-linear complex search spaces.

2.4.4 Framework Applications

The flexibility and generality of proposed optimisation based framework for test-case data generation is demonstrated by its application to a number of testing problems. The include (or will):-

- WCET
- Constraint-Solving
- Specification Conformance
- Structural Coverage
- Assertions and Exception Condition
- Mutation Analysis
- Numerical Precision
- Regression Testing
- Optimal Test-Subsets

Details of these some of these applications can be found in [Tra97, TCM98a, TCM98b].

2.4.5 Evaluation

To evaluate the efficacy of the test-case data generation framework a large industrial scale case-study will be presented. For this we have obtained source code to a industrial aero-engine controller from Rolls-Royce and have also elicited co-operation from British Aerospace and Praxis Critical Systems in obtaining further code examples.

Chapter 3

Progress

3.1 Introduction

This chapter provides details of the progress towards completion of a successful DPhil thesis. It discusses the research already carried out, the tool-set development and the material already produced.

3.2 Research Work

The research work to date as focused on through review of the available literature and then on some early applications of the optimisation-based test-data generation framework to examine efficacy. [Tra97] presents the results of a review of the test-data generation and optimisation technique literature, although this review work is obviously still in progress. Much of the research work carried out to date can be found in [TCM98a, TCM98b].

3.3 Development Work

The development work carried out has resulted in a prototype tool-set which has been used to examine the efficacy of the applications of the framework. Currently the only implemented optimisation technique is simulated annealing. This is because it was considered more important to test a number of testing problems before investigating optimisation technique efficiency. Figure 3.1 shows the tools which make up the tool set. The purpose of each of these tools is briefly discussed below.

Extractor – This tool is responsible for processing the software under test. It extracts all the information necessary for the rest of the testing system and stores it in files. This tool is based on the GNAT [Inc97] compiler front-end as this gives access to a full semantic tree of the test software. This tool is also responsible for producing instrumented or altered (i.e. straight-line) versions of the software under test when this is needed by testing problem being addressed.

Normal Form – This tool converts a constraint system into DNF.

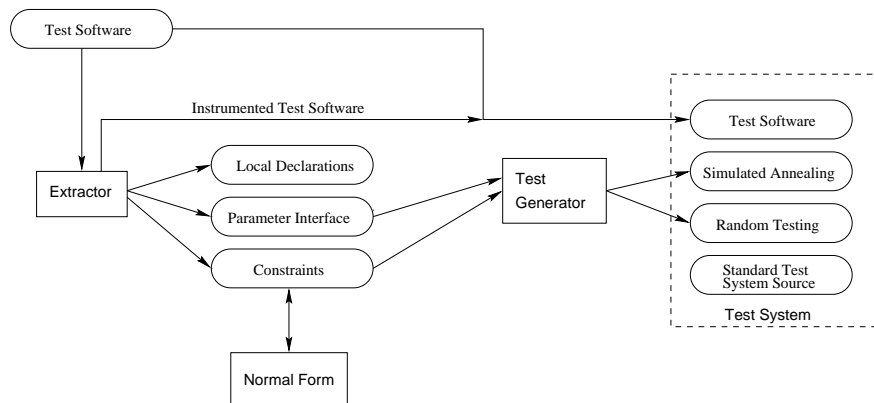


Figure 3.1: Prototype Tool-Set

Test Generator – This tool uses the information from the other tools to produce the test-system. This test system is compiled and linked with the software under test. When executed, this test system will use the desired search strategy (at present either random or simulated annealing) to find a solution to the desired testing problem.

3.4 External Presentations and Publications

The results of the research to date have been presented in a number of external seminars and publications. These are described below:

Rolls-Royce – A seminar was given to outline the capabilities of the tool-set to Rolls-Royce. The presentation focused on the application of the framework to WCET testing, constraint-solving and specification failure testing. This seminar generated interest in the approach and support in the form of industrial source-code for an aero-engine controller for use in the evaluation of the techniques.

British Aerospace – This seminar focused on the application of framework to structural testing and exception condition testing. This seminar resulted in the offer of co-operation in the evaluation of the exception condition testing application.

Praxis – This seminar focused on the specification failure and exception testing applications of the framework. This resulted in the offer of annotated source code from Praxis for use in evaluation of the framework.

IFIP-DCIA – This paper was presented at the IFIP International Workshop on Dependable Computing and Its Applications in South Africa, January 1998. A copy of the paper is attached to this proposal.

ISSTA – This paper was presented at the ACM/SIGSOFT International Symposium on Software Testing and Analysis in Florida, USA, March 1998. A copy of the paper is attached to this proposal.

CHAPTER 3. PROGRESS

ASE – This paper is in preparation for submission to the IEEE International Conference on Automated Software Engineering (1998).

Chapter 4

Plans

4.1 Introduction

This chapter provides a chapter-by-chapter timetable for the completion of the thesis. It also identifies research and development work necessary for completion of each chapter.

4.2 Thesis Writing

4.2.1 The Review

The majority of the work for the review has already been carried out and written-up. However, new relevant material is appearing all the time and this will need to be read and written up. Also as new applications of the framework are considered new bodies of material will need to be surveyed.

Research and Development : 1 Month

Writing Up : 0.5 Month

4.2.2 Optimisation for Test-Data Generation

The work required for this section involves investigation in to other optimisation techniques and comparisons of their efficiency and suitability for the different testing problems. So far only simulated annealing has been developed and integrated into the tool-set. Preliminary investigations on how to apply the other optimisation techniques to software testing have been undertaken and are presented in [Tra97]. An investigation of problem specific enhancements to the optimisation techniques will also need to be undertaken.

Research and Development : 3 Month

Writing Up : 1 Month

4.2.3 Framework Applications

Several applications of the framework have already been implemented and undergone a preliminary evaluation. These include WCET, Constraint-Solving (now integrated into Cadiz [CT97]), Specification-Conformance and Structural-Coverage testing. New applications of the framework will involve research into the testing problems and then development of the tool-set to allow it to generate test-data for these new problems, e.g. mutation testing, regression test-sets, precision, etc . . .

Research and Development : 3 Month

Writing Up : 3 Month

4.2.4 Evaluation

This section will take the form of several case-studies. Each will demonstrate the performance of the test-case data generation for a particular testing problem using *real* large-scale industrial software source-code. Access to suitable source-code and test-data for this evaluation has already been achieved or offered from several companies after presenting at external seminars – Rolls-Royce, British-Aerospace and Praxis Critical Systems.

Research and Development : 6 Month

Writing Up : 2 Month

Much of the work will be carried out concurrently such as the development and writing-up new framework applications and the associated evaluation.

Bibliography

- [Bei90] B. Beizer. *Software Testing Techniques*. Thomson Computer Press, 2nd edition, 1990.
- [BEL75] R. Boyer, B. Elspas, and K. Levitt. Select – a formal system for testing and debugging programs by symbolic execution. *Proceedings International Conference of Reliable Software*, pages 234–245, 1975.
- [Bur67] W. Burkhardt. Generating programs from syntax. *Computing*, 2(1):83–94, 1967.
- [Bur96] Simon Burton. Automatic test generation. Master’s thesis, Department of Computer Science, University of York, 1996.
- [Cla76] L. Clarke. A system to generate test data and symbolically execute programs. *IEEE Transactions on Software Engineering*, SE-2(3):215–222, September 1976.
- [CT97] John Clark and Nigel Tracey. Solving constraints in law 22117. Law/d5.1.1(e), European Commission - DG III Industry, 1997. Legacey Assessment Worbenche Feasibility Assessment.
- [DO91] R. Demillo and A. Offutt. Constraint-based automatic test data generation. *IEEE Transactions on Software Engineering*, 17(9):900–910, 1991.
- [FvBK⁺91] Susumu Fujiwara, Gregor v. Bockmann, Ferhat Khendek, Mokhtar Amalou, and Abderrazak Ghedamsi. Test selection based on finite state models. *IEEE Transactions on Software Engineering*, 17(6):591–603, June 1991.
- [Inc97] Ada Core Technologies Inc. The gnat ada-95 compiler, 1997. <http://www.gnat.com/>.
- [JSE96] B. Jones, H. Sthamer, and D. Eyres. Automatic structural testing using genetic algorithms. *Software Engineering Journal*, 11(5):299–306, 1996.
- [Kin76] J. King. Symbolic execution and program testing. *Communications of the ACM*, 19(7):385–394, 1976.
- [Kor90] B. Korel. Automated software test data generation. *IEEE Transactions on Software Engineering*, 16(8):870–879, 1990.

BIBLIOGRAPHY

- [Kor96] B. Korel. Automated test data generation for programs with procedures. *ACM ISSTA*, pages 209–215, 1996.
- [Meu97] Christophe Meudec. *Automatic Generation of Software Test Cases from Formal Methods*. PhD thesis, Belfast University, 1997.
- [MS76] W. Miller and D. Spooner. Automatic generation of floating-point test data. *IEEE Transactions on Software Engineering*, SE-2(3):223–226, September 1976.
- [OP96] A. Jefferson Offutt and Jie Pan. The dynamic domain reduction procedure for test data generation. <http://www.isse.gmu.edu/faculty/ofut/rsrch/atdg.html>, 1996.
- [Oul91] M. Ould. Tesintg - a challenge to method and tool developers. *Software Engineering Journal*, 6(2):59–64, March 1991.
- [Som92] Ian Sommerville. *Software Engineering*. Addison-Wesley, fourth edition, 1992.
- [Spa90] E. H. Spafford. Extending mutation testing to find environmental bugs. *Software – Practice and Experience*, 20(2):181–189, February 1990.
- [TCM98a] Nigel Tracey, John Clark, and Keith Mander. Automated program flaw finding using simulated annealing. In *International Symposium on Software Testing and Analysis*, pages 73–81. ACM/SIGSOFT, 1998.
- [TCM98b] Nigel Tracey, John Clark, and Keith Mander. The way forward for unifying dynamic test case generation: The optimisation-based approach. In *International Workshop on Dependable Computing and Its Applications*, pages 169–180. IFIP, 1998.
- [Tra97] Nigel J. Tracey. Test-case data generation using optimisation techniques – first year dphil report. Department of Computer Science, University of York, 1997.
- [Wat95] Alison Lachut Watkins. The automatic generation of test data using genetic algorithms. *Proceedings of the 4th Software Quality Conference*, 2:300–309, 1995.
- [Yan95] Xile Yang. The automatic generation of software test data from z specifications. Technical report, Department of Computer Studies, University of Glamorgan, 1995.