TeTra: Testability Transformation

Mark Harman

May 3, 2002

Mark Harman, Brunel University
http://www.brunel.ac.uk/~csstmmh2/
Co-Investigators
Rob Hierons, Department of Information Systems and Computing, Brunel University, UK
Sebastian Danicic, Department of Mathematical and Computing Sciences, Goldsmiths College, UK
Chris Fox, Department of Computer Science, University of Essex, UK

Collaborators
Knowledge Software Ltd.
Software Migrations Ltd.
DaimlerChrysler

Keywords: Software Testing, Automatic Test Data Generation, Evolutionary Testing

1 Overview

TeTra is an EPSRC-funded project, due to start late in 2002, but for which we are already laying the ground work. This page presents a summary, based upon the proposal submitted to the EPSRC in February 2002.

Generating good quality test data is difficult and expensive, yet it is vital for software reliability. The TeTra project aims to remove barriers to automated generation of test data using a specifically adapted form of program transformation called 'testability transformation'. This will require the formulation of new kinds of transformation which preserve properties relevant to test-coverage and the construction of novel algorithms based upon this formulation. TeTra will therefore extend research in both transformation and testing and will provide a link between the two areas of research activity (which have previously been considered to be disjoint). In order to achieve this, the TeTra team will develop theory, algorithms and tools for testability transformation and will evaluate these using case studies provided by the industrial collaborators. The principal deliverables will be:

- A theoretical framework for testability transformation together with evaluation benchmarks
- Algorithms for testability transformation
- Prototype tools and results of their evaluation

Testing is increasingly recognised as a crucial part of the software development process and one which is important for almost all forms of software. Unfortunately, testing is also a hard problem, not least because of the difficulty in automating the identification of high quality test data. The TeTra project will be concerned with the problem of automatic generation of test data using evolutionary testing techniques, based upon white box (or structural) test adequacy criteria. In particular the project will focus upon branch and condition-based coverage criteria, which are currently regarded as an industry standard [3] and which are mandatory for many applications, for example avionics [12].

Branch coverage criteria require that a test set contains, for each feasible branch of the program under test, at least one test case which causes execution to traverse the branch. Unfortunately, automatically generating test data which have this property is not generally computable, and so research and development have focused upon techniques which aim to optimise test data with respect to such a criterion. One such optimisation technique which has been of particular recent interest has been Evolutionary testing [6, 7, 8, 9, 10, 11, 14, 17, 18]. Evolutionary testing uses metaheuristic search based techniques[1] to find good quality test data. Test data quality is defined by a test adequacy criterion, which underpins the fitness function which drives the search implemented by the genetic algorithm.

For structural criteria, such as those which will be of importance for the TeTra project, a fitness function is typically defined in terms of the program's predicates. It determines the fitness of candidate test data, which in turn, determines the direction taken by the search. The fitness function essentially measures how close a candidate test input drives execution to traversing the desired (target) path. The TeTra project will reformulate program transformation to overcome remaining barriers to wider application of structural evolutionary test data generation. Previously transformation has been applied to many problems including automatic parallelization [13, 19], program comprehension [2, 16], reverse and re-engineering [15], efficiency improvement [1] and (by the proposers, using slice-based transformations) to testing [4, 5]. However, TeTra will be the first to apply transformation to the test data generation problem.

3 Programme and Methodology

3.1 The Problem

Generating test data using evolutionary test data generation has been shown to be successful [6, 17, 14, 10, 8], but its effectiveness is significantly reduced in the presence of programming styles which make the definition of an effective fitness function problematic. For example:

- The presence of side effects in predicates reduces the ability to exploit the inherent parallelism in a predicate's syntactic structure.
- The use of flag variables (and enumeration types in general) creates a coarse fitness landscape, thereby dramatically reducing the effectiveness of the search.
- Unstructured control flow (in which loops have many entry and exit points) affects the ability to determine how alike are the traversed and target paths.

The presence of these features make a program less 'testable'. Although unstructured control flow and reliance upon side effects are often deprecated, their use remains prevalent and so it is not good enough simply to have testing strategies only for 'pure and clean' programs and languages; testing is a real world problem and it requires real world solutions.

3.2 The Proposed Solution

When presented with problems of programming style, a natural solution is to seek to transform the program to remove the problem. In the context of improving testability, this will be termed 'testability transformation'. The TeTra project will focus on testability transformations for improving evolutionary testing by removing unstructured control flow, side effects and use of flag variables. Poor structure and side effects are well-known from other fields of study, for example, comprehension and maintenance. However, existing transformations for ameliorating such problems will need to be adapted to be applicable to testability transformation. The flag problem is less well studied from a transformational point of view and so this application too, will require the development of novel transformation algorithms.

There is an apparent paradox at the heart of this proposal:

> Structural testing is based upon structurally defined test adequacy criteria. The automated generation of test data to satisfy these criteria can be impeded by properties of the software (for example, flags, side effects, and unstructured control flow). Testability transformation seeks to remove the problem by transforming the program

[1] Typically genetic algorithms and simulated annealing have been used, but TeTra requires only that the technique used is characterised by some fitness (or cost) function, for which the search seeks to find an optimal or near-optimal solution.

program. Since the program's structure is altered and the adequacy criteria is structurally defined, it would appear that the original test adequacy criterion may no longer apply.

Testability transformation therefore requires *co-transformation* of the adequacy criterion; this avoids the apparent paradox. An (informal) definition of a testability transformation is given below:

Definition 1 *Testability Transformation*
A Testability transformation maps a program, p and its associated test adequacy criterion, c to a new program p' and new adequacy criterion, c', such that any set of test data which is adequate for p' with respect to c' is also adequate for p with respect to c.

Observe that, while traditional transformations are meaning preserving functions on programs, testability transformations are 'test set preserving' functions on pairs containing both a program and its associated adequacy criterion. Therefore, TeTra will construct transformations which allow the test adequacy criterion to be re-formulated so that it preserves the meaning of the original adequacy criterion with respect to the original program. The definition of transformation rules and algorithms based upon this new notion of transformation is the essential research challenge which underpins the TeTra project.

3.3 Novelty, Impact, Timeliness and Adventure in Research

The TeTra approach is novel, both in its application of transformation to the problem of automated test data generation and in the way in which it becomes necessary to change the traditional view of transformation in order to achieve this:

- **Transformation merely as a means to an end**
  Program transformation has previously been regarded as an *end* in itself, rather than merely as a *means* to an end. Hitherto, the goal of transformation research has been to transform poor programs into better ones. Thus research has tended to assume that the original program will be discarded once the transformed program has been constructed. By contrast, in the TeTra approach, it is the transformed program which is discarded and the original which is retained. Testability transformation requires the transformed program solely for the purpose of test data generation. Definition 1 guarantees that such test data will be adequate for the original. Once the test data is generated, the transformed program is no longer required.

- **Novel Notions of Equivalence Preserving Transformation**
  Program transformation has traditionally been concerned with the preservation of functional equivalence. In the TeTra approach, the transformed program must provide a wholly different guarantee; that test data generated from it is adequate for the original program. This means that the transformations applied need not preserve any notion of traditional equivalence, opening the possibility of a novel set of program transformation rules and algorithms.

Software test data generation is widely recognised as a hard problem, yet industry standards rightly require thorough testing which meets well-defined and understood test adequacy criteria [3, 12]. Evolutionary testing is a promising approach to attack the problem of automated generation of adequate test data, yet the structural problems described earlier inhibit its wider application. The TeTra project presents an opportunity to make a significant step forward in automated test data generation, thereby moving forward research and practice in an area of major concern. The proposed combination of transformation and evolutionary testing could not come at a better time. The DaimlerChrysler evolutionary testing system and the FermaT transformation system represent the current state of the art. The FermaT system was made publically available under GPL only very recently (in November 2001), yet the proposers have had two years experience working with it as part of the GUSTT project.

4 Project Management

The principal work-packages are summarised below. Scheduling and duration information for each can be found in the TeTra Gantt Chart.

WORK PACKAGE 01: **Benchmark Evaluation Programs**:	DELIVERABLE: Evaluation Criteria,Benchmarks

The project will start with the development of a set of programs and associated unit-level testing problems. The programs will be identified by the industrial partners, and will be refined to act as exemplars of particular evolutionary testing problems. This initial phase will be necessary to ensure that the team have available a wide spectrum of examples which contain industrially relevant problem cases, intellectually challenging examples that highlight particular features and

of application.

WORK PACKAGE 02: **Theoretic Framework**:	DELIVERABLE: Report/paper

A theory of testability transformation will be established. Testability transformations need not be equivalence preserving but must preserve the sets of adequate test data which can be generated from a program and its transformed version. This will involve new notions of equivalence preservation, relevant to testability. The TeTra project will therefore develop a novel program transformation theory to underpin the algorithmic and tool development work. This will allow proofs of correctness for the transformations which will be developed in much the same way that traditional transformations are proved correct with respect to functional equivalence.

WORK PACKAGE 03: **Algorithms**:	DELIVERABLE: Report/paper

Based upon the theory and with regard to the benchmark programs, algorithms for testability transformation will be developed. The algorithms will focus upon transformations to alleviate problems associated with side effects, unstructured control flow and reliance upon flag variables. These features are particularly prevalent in embedded systems of the type with which DaimlerChrysler has most experience. Such programs typically contain side effects (for efficiency), poor structure (because they are often machine generated) and flags (because there are many exceptional controller states which tend to be monitored by flag variables).

WORK PACKAGE 04: **Initial Evaluation**:	DELIVERABLE: Report/paper

Using the benchmark programs, the transformation algorithms will be evaluated. For work package 4, the transformation step of the algorithms will be performed by hand on the smaller of the benchmark programs. It is important to evaluate the algorithms in this way before committing to the effort of prototype tool development. The evaluation will motivate modifications to the algorithms. Also additional benchmark programs will be made available as a result of on-going concurrent work on Evolutionary Testing by the collaborators.

WORK PACKAGE 05: **Prototype Tool Development**:	DELIVERABLE: Tools

To fully explore and evaluate the testability transformation algorithms arising from previous work-packages, prototype implementation will be required. Implementation will use the FermaT transformation workbench supplied by Software Migrations Ltd.

WORK PACKAGE 06: **Full Evaluation**:	DELIVERABLE: Report/paper, enhanced algorithms and tools

The prototype tools will be used to provide a thorough evaluation of the algorithms. Since the algorithms aim to remove impediments to evolutionary testing, the criteria for successful evaluation will be that test effort is reduced (fewer generations required to achieve a benchmark level of coverage) and quality is increased (higher levels of coverage obtainable).

The full evaluation stage will also measure the actual performance of the algorithms relative to the predicated upper bounds on algorithmic complexity, as these are often found to differ widely in other application areas for program transformation.

WORK PACKAGE 07: **Horizons**:	DELIVERABLE: Horizons report

In collaboration with the industrial and academic collaborators, areas for extension of testability transformation will be identified. Specific areas to be considered will include constraint based test data generation and regression testing, which are not addressed in the core of the TeTra project, but for which testability transformation is also likely to be a valuable assistant.

[1] Aho, A. V., Sethi, R., and Ullman, J. D. *Compilers: Principles, techniques and tools.* Addison Wesley, 1986.

[2] Bennett, K., Bull, T., Younger, E., and Luo, Z. Bylands: reverse engineering safety-critical systems. In *IEEE International Conference on Software Maintenance* (1995), IEEE Computer Society Press, Los Alamitos, California, USA, pp. 358–366.

[3] British Standards Institute. BS 7925-2 software component testing, 1998.

[4] Harman, M., and Danicic, S. Using program slicing to simplify testing. *Software Testing, Verification and Reliability 5*, 3 (Sept. 1995), 143–162.

[5] Hierons, R. M., Harman, M., and Danicic, S. Using program slicing to assist in the detection of equivalent mutants. *Software Testing, Verification and Reliability 9*, 4 (1999), 233–262.

[6] Jones, B., Sthamer, H.-H., and Eyres, D. Automatic structural testing using genetic algorithms. *The Software Engineering Journal 11* (1996), 299–306.

[7] Jones, B. F., Eyres, D. E., and Sthamer, H. H. A strategy for using genetic algorithms to automate branch and fault-based testing. *The Computer Journal 41*, 2 (1998), 98–107.

[8] Michael, C., McGraw, G., and Schatz, M. Generating software test data by evolution. *IEEE Transactions on Software Engineering*, 12 (Dec. 2001), 1085–1110.

[9] Mueller, F., and Wegener, J. A comparison of static analysis and evolutionary testing for the verification of timing constraints. In *4th IEEE Real-Time Technology and Applications Symposium (RTAS '98)* (Washington - Brussels - Tokyo, June 1998), IEEE, pp. 144–154.

[10] Pargas, R. P., Harrold, M. J., and Peck, R. R. Test-data generation using genetic algorithms. *The Journal of Software Testing, Verification and Reliability 9* (1999), 263–282.

[11] Pohlheim, H., and Wegener, J. Testing the temporal behavior of real-time software modules using extended evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Orlando, Florida, USA, 13-17 July 1999), W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., vol. 2, Morgan Kaufmann, p. 1795.

[12] Radio Technical Commission for Aeronautics. RTCA DO178-B Software considerations in airborne systems and equipment certification, 1992.

[13] Ryan, C., and Walsh, P. The evolution of provable parallel programs. In *Genetic Programming 1997: Proceedings of the Second Annual Conference* (Stanford University, CA, USA, 13-16 July 1997), J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, Eds., Morgan Kaufmann, pp. 295–302.

[14] Tracey, N., Clark, J., and Mander, K. The way forward for unifying dynamic test-case generation: The optimisation-based approach. In *International Workshop on Dependable Computing and Its Applications (DCIA)* (January 1998), IFIP, pp. 169–180.

[15] Ward, M. Reverse engineering through formal transformation. *The Computer Journal 37*, 5 (1994).

[16] Ward, M., Calliss, F. W., and Munro, M. The maintainer's assistant. In *Proceedings of the International Conference on Software Maintenance 1989* (1989), IEEE Computer Society Press, Los Alamitos, California, USA, p. 307.

[17] Wegener, J., Grimm, K., Grochtmann, M., Sthamer, H., and Jones, B. F. Systematic testing of real-time systems. In *4th International Conference on Software Testing Analysis and Review (EuroSTAR 96)* (1996).

[18] Wegener, J., Sthamer, H., Jones, B. F., and Eyres, D. E. Testing real-time systems using genetic algorithms. *Software Quality 6* (1997), 127–135.

[19] Williams, K. P. *Evolutionary Algorithms for Automatic Parallelization.* PhD thesis, University of Reading, UK, Department of Computer Science, Sept. 1998.