# TAIC PART 2007 and Mutation 2007 Special Issue Editorial

Mark Harman and Zheng Li

*King's College London, Centre for Research on Evolution, Search and Testing (CREST), Department of Computer Science, Strand, London, WC2R 2LS, UK.*

Phil McMinn

*Department of Computer Science The University of Sheffield Regent Court, 211 Portobello, Sheffield, S1 4DP, UK.*

Jeff Offutt

*Volgenau School of Information Technology and Engineering, George Mason University, Fairfax, VA 22030, USA.*

John Clark

*Department of Computer Science, University of York, York YO10 5DD, UK.*

## 1 Introduction

Software testing is a topic of growing importance because of the central role it plays in so many aspects of software engineering, both pre– and post– delivery. There is strong empirical evidence [1,2] that deficient testing of both functional and non-functional properties is one of the major sources of software and system errors. In 2002, NIST estimated the cost of software failure to the US economy at $60,000,000,000; an astonishing 0.6% of GDP [3]. The same report found that more than one third of these costs of software failure could be eliminated by an improved testing infrastructure. As these data reveal, the importance of work on software testing and its potential impact on the global economy is hard to overstate.

TAIC PART 2007 was a unique software testing event that combined aspects of a conference, a workshop and a retreat. Through this unique blend of attributes the event brought together industrialists and academics in an environment that sought to promote meaningful collaboration on problems in software testing.

Among computer science and software engineering activities, software testing is a perfect candidate for such a union of academic and industrial minds, because the problems thrown up by software testing touch upon so many academic research concerns, while the implications of advances in research can have such wide ranging and far reaching implications for industry.

Testing research combines elements of computability and algorithmic complexity theory with the mathematics and pragmatics of representations such as finite state machines, flow graphs, call graphs and dependence graphs. It involves disciplines that cover the spectrum of software engineering activity, from psychology, through engineering to pure mathematics and even philosophy. This astonishing breadth and depth have made the problems of software testing appealing to academics for several decades.

However, as industrialists know, testing software is unlike any other kind of engineering testing activity. It will not suffice to test at two ends of a spectrum of values, in order to infer properties in-between. Software systems exhibit discrete behaviour and may combine this with non-determinism and emergent behaviour. No other engineering artefact is more closely integrated with the human mind, leading to complex hybrid systems that involve software, human judgement and, sometimes, political, legal and social processes. As technology matures, the planet is increasingly becoming enveloped in a 'software skin' of interconnected, interdependent processes, under which software controls and regulates both the 'blood flow' of information and the 'mechanisms of action'.

TAIC PART 2007 was funded by the Engineering and Physical Sciences Research Council (the EPSRC). The EPSRC is the UK's primary funding body for research in Science and Engineering. The EPSRC evaluated TAIC PART on completion by peer review and gave it the highest rating possible ('Outstanding'). TAIC PART was held previously in 2006, and was an outstanding success, building upon previous smaller workshops on testing held in the UK, which steadily built a strong community of researchers and industrialists.

The original TAIC PART 2006 workshop matured these workshops into a larger and more ambitious event. TAIC PART 2006 was a great success, with industrialists present from Ericsson, Vizuri, IPL Bath Ltd., Motorola, IBM, Nokia, DaimlerChrysler and AT & T. Best papers from the event were extended for a special issue of the Software Testing, Verification and Reliability (Volume 18, Number 1). TAIC PART 2006 was also funded by the EPSRC and also rated by peer review to have been 'Outstanding'.

The TAIC PART conference/worlshop/retreat concept was built on three previous workshops held in the UK. The first workshop, managed by Prof. John Clark, was held in 1998 at the University of York. The second workshop, managed by Profs. Clark, Harman and Hierons, was also held at the University of

York in 2003. It received attention from small companies working on testing tools, as well as large–scale industrial users of testing technology, such as IBM and DaimlerChrysler. The third testing workshop was held at the University of Sheffield in 2005, with involvement from several SMEs working on testing tools and techniques and from large organisations, including IBM and Motorola. A special issue of best papers from the 2003 workshop appeared in the Software Testing, Verification and Reliability journal (Volume 14, Number 3, September 2004), and again for the 2005 event (Volume 16, Number 3, September 2006).

In order to attract attendees to the event, through the submission of papers, a programme committee of high international quality was assembled. Many countries were represented, including the UK, USA, France, Canada, Lebanon, Germany, China, Sweden and India. This committee was made up of 13 industrialists and 29 academics. The industrialists on the committee came from Motorola, Ericsson, Nokia, Electromind, Microsoft, SEVEN Networks, Testing Solutions Group Ltd, LDRA software technology Ltd., Tata Consultancy Services Ltd., Vizuri Ltd., IBM and DaimlerChrysler.

The four papers from TAIC PART in this special issue are extended versions of those TAIC PART papers that received the strongest support from the referees. Each of the papers has been extended and fully re-refereed by at least three expert reviewers and has undergone revisions as a result. The first two TAIC PART papers are concerned with test data generation, while the second two focus on fault prediction and localization.

In the paper by Charreteur et al., constraint-based testing is enhanced so that dynamically allocated structures can be handled. Constraint-based testing involves finding a series of constraints that describe the test objective and the semantics of the program under test, and then using a constraint solver to find the test data that will then execute that objective. Handling dynamic memory is challenging, because constraints are derived statically, not at run-time, and yet it is not always possible to determine the 'shape' of a dynamic data structure statically or at compile time. However, Charreteur et al. present constraint operators that can be used to reason over a program's interaction with dynamic memory, allowing test data to be generated for programs using pointer types.

In the paper by Wappler et al., the flag problem for evolutionary test data generation is revisited in the context of function-assigned flags. Code containing Boolean flag variables hinders the effectiveness of evolutionary testing. This is because the fitness function, which is responsible for guiding the search to the required test data, is reduced to two fitness invariant 'plateaux' corresponding to the true and false values of the flag. A new testability transformation is introduced. The transformation seeks to replace flags returned by functions with values that give more guidance to the evolutionary search.

In the paper by Abreu et al., diagnostic accuracy of spectrum-based fault localization is investigated. Spectrum-based fault localization involves keeping a record of program components executed in faulty and fault-free program runs, and then ranking potential fault locations by comparing each of the runs with a similarity coefficient. Abreu et al. analyze the effectiveness of the Ochiai similarity coefficient, previously found to to be more accurate that eight other coefficients in the field, investigating its soundness when the quantity of program runs and the quality of their classification (i.e. whether they passed or failed) are subject to variation. The empirical evaluation includes a large-scale industrial embedded system.

In the paper by Binkley et al., natural language techniques are applied to program identifiers with the aim of predicting faults. Well-chosen identifier names make a program more maintainable and with fewer defects, whilst for programs with poorly chosen identifiers the reverse is true. Binkley et al. contend that programs in the latter category are more vulnerable to faults. They propose three different measures that they apply to programs in their empirical study. The first measure is the percentage of natural language featuring in a program's identifiers, with the reasoning that the more natural language words used, the easier the code will be to read and understand. Hence fewer faults are predicted. The second measure is the percentage of identifiers that violate syntactic conciseness and consistency rules. A high percentage here may indicate confusion about what the code is supposed to be doing, and therefore more faults are predicted. The final measure used is the similarity between language used in a module's comments and its code. This can be used to assess whether the code is well or accurately documented. Any divergence may indicate potential programmer misunderstandings, and thus the introduction of faults.

Mutation testing has been widely studied in Academia for over 30 years and has been demonstrated, in the laboratory, to be an extremely effective approach to testing. Nevertheless, it is fair to say that the approach is less widely explored within industry than perhaps it might be. Therefore, it made perfect sense to co-locate the 3rd Mutation Testing workshop with the TAIC PART conference, so Mutation Testing experts from academia and industrialists interested in testing could meet to share ideas within the overall retreat setting of TAIC PART. It was hoped that this would increase awareness of the Mutation Testing approach within industry and also provide Mutation Testing researchers with a opportunity to hone their techniques to make them more readily applicable within industrial testing scenarios.

The Mutation Testing workshop was sponsored by Certess, an industrial provider of Mutation Testing technology, thereby demonstrating that there is, indeed, industrial interest and relevance in Mutation Testing. Mutation 2007 included 11 papers (selected from 17 submissions), together with a keynote presentation and two panels. Two papers were selected for this special issue. Both are extended and full re-refereesd for this special issue.

The first, a paper by Robert M. Hierons and Mercedes G. Merayo, applies mutation analysis to the problem of analyzing finite state machine models of software. The paper introduces methods to mutate FSMs using probabilities and stochastic time annotations on transitions, then presents ways to generate tests that distinguish between the original and mutated FSMs.

The second, a paper by Ben H. Smith and Laurie Williams, presents an empirical study of the idea of using mutation analysis to improve an existing test set. They found that using mutation in this way proved to be a very effective way to increase the coverage of test sets at reasonable cost.

Mutation 2007 was the third in a series of Mutation Testing workshops, the first of which was held in 2000 and the second in 2006. It was hoped that Mutation 2007 would become the third in an annual series of Mutation Testing workshops. Fortunately, this hope is becoming something of a reality: The 4th Mutation Testing workshop was held in co-location with the 2nd IEEE International Conference on Software Testing (ICST 2009) and a fifth is planned for co-location with the 3rd ICST 2010 in Paris.

## References

[1] R. L. Glass, Facts and Fallacies of Software Engineering, Addison Wesley, 2002.

[2] D. Leffingwell, D. Widrig, Managing Software Requirements: A Use Case Approach, Addison Wesley, 2003.

[3] National Institute of Standards, Technology (NIST), The economic impacts of inadequate infrastructure for software testing, planning Report 02-3 (May 2002). URL `http://www.nist.gov/director/prog-ofc/report02-3.pdf`