Guest editor's introduction to the special section on TAIC-PART2010 – Testing: Academic and Industrial Conference-Practice and Research Techniques

Anthony J H Simons
Department of Computer Science
University of Sheffield
Sheffield, UK

Gordon Fraser
Department of Computer Science
Saarland University
Saarbrücken, Germany

Leonardo Bottaci
Department of Computer Science
University of Hull
Hull, UK

Almost all organisations, irrespective of their size, rely on IT systems. Individuals are making increasing use of web applications for personal and social activities. Furthermore, as systems in general become more technologically advanced, an increasing proportion of the implementation is in software rather than in hardware. As a result, the quality and reliability of systems and the data they contain is increasingly dependent on the correctness, robustness and security of the associated software. The trend is clear: software is becoming pervasive and, as a result, software quality assurance, in the widest sense, is becoming a crucial everyday concern for all of us. Software testing is, by far, the most important means of assuring software quality.

The Testing: Academic and Industrial Conference-Practice and Research Techniques 2010 (TAIC-PART 2010) was the fifth conference in a series of highly successful events. The conference brought together industrialists and academics to promote collaboration on problems in software testing. Among the wide range of topics in computer science and software engineering, software testing is an ideal candidate for academic and industrial collaboration because advances in research can have such wide-ranging and far-reaching implications for industry. Conversely, the advances in computing and communications technology and the growth of the associated software engineering activity are producing new research challenges at an increasing rate.

TAIC-PART 2010 received forty paper submissions. After a rigorous reviewing process in which each paper was subjected to at least three independent reviews followed by programme committee discussion, 15 full papers and seven abstracts were accepted. One of these submissions was from industry, four were from collaborations between industry and academia and the remainder came from academia. The papers originated from 13 countries in Europe, North and South America, and Asia.

The two best papers selected by the editors for publication in this special section are significantly extended versions of those TAIC-PART papers that received the strongest support from the referees. Each of the extended papers has been refereed by at least three expert reviewers and has undergone revisions as a result. They not only address important issues in security testing and fault localisation but also use novel and interesting techniques that could be deployed more widely.

The first paper by Ben Kam and Thomas Dean, Linguistic security testing for text communication protocols, considers the problem of protocol security testing. Protocols rely heavily on syntactic structures, and the authors adopt a syntax-based approach to security testing. The basic idea is to capture existing input to the application under test. The input is then parsed into an abstract syntax tree. Using a set of rewrite rules, the input is transformed to create test cases. A key contribution of their work is that they use a relatively simple context-free grammar together with XML mark-up rules to specify lexical, syntactic and context sensitive constraints. From this augmented grammar, they automatically generate a program that inserts XML tags into the appropriate parts of the captured test input. Multiple test generators can then use the tagged test input to produce different test cases, which are used to validate the security protocols of the software under test.

In a case study, the authors tested an application that uses the iCalendar protocol for the communication of calendar information. Just over a thousand test cases were generated from a sample input; and upon execution of those test cases, an error was found, corresponding to a weakness in the iCalendar security protocol. Overall, the Syntax-based Security Testing framework is a relatively lightweight way of producing a protocol independent testing tool. New testing schemes can be accommodated by augmenting the grammar to add additional mark-up and creating the appropriate test case generators.

The second paper by Mihai Nica, Simona Nica and Franz Wotawa, On the use of mutations and testing for debugging, is concerned with the problem of debugging but from a testing viewpoint. The main goal is to use testing as a way of localising faults within software. The underlying concept is to treat the statements in a program as a set of equations on the variables that appear in those statements. Each test case constrains the values of input and output variables. A failing test produces an inconsistency in the system of equations. The fault localisation process works by attempting to restore consistency. By removing individual statements until the system of equations is consistent, it is possible to identify candidate faulty statements, of which there are more than the actual number of faults.

These candidates are subject to mutation, as a way of attempting to effect a repair. Candidates are retained if their repaired mutation causes an increase in consistency, whereas other candidates are discarded. Further test cases may be introduced to compare alternative surviving candidates and alternative repairs, which may need some input from the tester, or an oracle to help localise the fault. The approach is largely automated, but there are limitations as to the program constructs that can be represented as a set of equations. In addition, the constraint solver does not scale to large programs. In spite of these limitations, however, the approach is applied to a real world program with encouraging results.

Both papers have a strong technical and also an empirical content. They capture the interplay of research and practice that is central to the aim of the conference and are fine examples of research in software testing.

Finally, we would like to thank all the authors and reviewers for their eff ort and time. Furthermore, we would also like to thank the editors-in-chief of Software: Practice and Experience, Nigel Horspool and Andy Wellings, and the editorial staff for their support.