Software Testing

Renee Bryce and Rick Kuhn

Many open problems exist for the Software Testing community. Products released with inadequate testing can cause bodily harm, result in large economic losses, and affect the quality of day-to-day life. Many innovative techniques exist for testing systems in different domains. In this special issue, we solicited papers that focus on important problems within the Software Testing community. We received 40 submissions that were each reviewed by top researchers and practitioners in order to select the best papers for the broad IEEE Computer audience. Let us share an overview of trends that we learned from submissions and reviewer feedback.

Practical Testing Problems

One key to improving the state of practice is sharing ideas through case studies and lessons learned. Practitioners are understandably reluctant to adopt unproven methods, or ones that have been demonstrated only in small academic studies. When real improvements are shown, they are likely to be adopted. Testing is typically half the total cost of software development, so improvements can have a significant impact on the bottom line.

A critical issue in improving test efficiency is matching the test approach to the application. Obvious differences exist between, for example, network protocol software and e-commerce applications. The protocol is likely to be based on a complex state machine with relatively few numerical calculations, while the other may have an elaborate user interface and a large number of inputs with numerous calculations and graphic output. This example also illustrates one of the key considerations in practical testing – the need for human involvement with many new applications. Testing that requires visual verification of results on a screen requires the absolute maximum in efficiency, to reduce the number of tests that are subject to the expense of human verification.

In addition to the type of computation and the need for human involvement, another critical dimension is the source of potential failures. Applications that must deal with a human adversary can require a fundamentally different approach to testing than those where nature is the only “adversary”. Even this distinction can be blurred when we consider human error as a source of failure. While the user may not actively try to defeat the system, human carelessness can sometimes be as difficult to predict as the behavior of an attacker.

Taking these considerations together, we begin to see one of the reasons why software testing is so often approached in many different ways. The overlap and interaction of system characteristics produce a vast number of combinations that are difficult to narrow down to a few testing “templates”. Protocols and GUIs are clearly different, but they may both have an underlying complex state machine; the careless user at the keyboard may be nearly as dangerous as the motivated attacker, and so on. This issue presents papers that highlight innovative approaches to testing the increasingly diverse and complex range of software applications today.

In This Issue

The first paper, "*Moving Forward with Combinatorial Interaction Testing*", provides an overview of Combinatorial Interaction Testing (CIT). Indeed CIT has been a trendy topic over the past several years with dozens of tools, highly cited research papers, and an annual workshop at the International Conference on Software Testing, Verification and Validation. This paper provides a brief overview of CIT and discusses open issues in this area.

The second paper, "*An Extensible Framework for Online Testing of Choreographed Services"*, focuses on an area with high economic impact. The Service-Oriented Architecture market has shown continuous growth to the tune of several billion dollars. This paper introduces open problems, including issues that arise in choreography-based systems. Further, they provide an architecture of a framework that supports a continuous online testing process.

The third paper, "*Penetration Testing in Web Services"*, investigates the effectiveness of automated tools for detecting vulnerabilities in web service applications. While the automated approach had some success in vulnerability detection, results were far below code inspection by experts. Tools also had a significant level of false positives, which can negate many benefits of the automated approach by adding extra work. The presence of human adversaries makes penetration testing more of an art than some other areas of testing. Work clearly remains to better understand how to effectively incorporate judgment into the testing process.

The final paper, "*Mobile Application Testing – Research Practice, Issues, and Needs"*, gives an overview of available tools for mobile testing. They compare over a dozen popular commercial and open-source mobile testing tools, which is useful for practitioners to quickly identify tools that may help them. On the other hand, this paper also discusses open problems that exist in the area of mobile application testing. Many issues exist for cost-effective and scalable mobile test environments, standards, automation, and tools. This area offers many opportunities for new software testing techniques and empirical studies that provide guidance to practitioners in the rapidly growing mobile market.

In summary, many open problems exist in the area of Software Testing. This special issue highlights just a few emerging techniques that apply to different domains. The articles provide motivation for future work to not only develop tools that provide solutions within the areas of interest, but also to pursue empirical studies that provide guidance on how to apply techniques based on the characteristics of applications and domains.