The software testing community plays a key role in a world that is increasingly shaped, if not controlled, by software. The importance of testing is never more apparent than when some piece of software is released without having undergone sufficient vetting, the results of which can be bodily harm, large economic losses, and even diminished quality of life. Fortunately, increasingly innovative techniques exist for testing systems in different domains, which helps ensure software’s reliability.

We solicited articles for this special issue that focus on important problems within the software testing community. Top researchers and practitioners reviewed the 40 submissions. We selected the best articles for Computer’s broad readership.

PRACTICAL TESTING PROBLEMS

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

A critical issue in improving test efficiency is in matching the test approach to the application. Obvious differences exist between, for example, network protocol software and e-commerce applications: the protocol for the former is likely to be based on a complex state machine with...
GUEST EDITORS’ INTRODUCTION

relatively few numerical calculations, whereas the latter might have an elaborate user interface and a large number of inputs with numerous calculations and graphic output. This latter example also illustrates one of the key considerations in practical testing—the need for human involvement in testing. Software testing requiring visual verification of results on a screen mandates absolute maximum efficiency to reduce the number of tests subject to the expense of human verification.

In addition to the type of computation and the need for human involvement, another critical testing dimension examines the source of potential failures. Applications that must defend against a human adversary can require a fundamentally different approach to testing than those in which the interface with the natural environment is the only “adversary” to worry about. Even this distinction can blur when we consider human error as a source of failure. Although the user might not actively or intentionally try to defeat the system, human carelessness can sometimes be as difficult to predict as attacker behavior.

Taking these considerations together, we begin to see the rationale for a wide variety of approaches in software testing. The overlap and interaction of system characteristics produces a vast number of combinations that are difficult to narrow down to a few testing “templates.” Protocols and GUIs are clearly different, but both might have an underlying complex state machine; a careless user at the keyboard could be as dangerous as the motivated attacker; and so on. We selected four cover features that highlight innovative testing approaches for the increasingly diverse and complex range of software applications today.

IN THIS ISSUE

The first article, “An Extensible Framework for Online Testing of Choreographed Services,” by Midhat Ali and colleagues, focuses on a high economic impact area—namely, the service-oriented architecture (SOA) market, which has shown continuous growth. The authors introduce current problems in testing SOA software, issues that arise in choreography-based systems, and then describe in detail a framework architecture that supports a continuous online testing process.

The second article, “Penetration Testing in Web Services,” by Nuno Antunes and Marco Vieira, investigated the effectiveness of automated tools for detecting vulnerabilities in Web service applications. Although the automated approach had some success in vulnerability detection, results were far inferior to code inspection by experts. Tools also had a significant level of false positives, which would negate many benefits of the automated approach because of the extra work. The presence of human adversaries makes penetration testing more of an art than science in some areas of testing. To better understand how to effectively incorporate judgment into the testing process will require additional research.

The third article, “Moving Forward with Combinatorial Interaction Testing,” by Cemal Yilmaz and colleagues, provides an overview of combinatorial interaction testing. Indeed, CIT has been a trendy topic over the past several years, as evidenced by the emergence of dozens of tools, highly cited research papers, and an annual workshop at the International Conference on Software Testing, Verification, and Validation. The authors provide a brief overview of the field.

Finally, in “Mobile Application Testing—Research Practice, Issues, and Needs,” by Jerry Gao and his colleagues, reviews available tools for mobile application testing. Specifically, the authors compare numerous popular commercial and open source tools and discuss open problems in the area of mobile application testing. This area offers many opportunities for new software testing techniques and empirical studies to provide guidance to practitioners in this rapidly growing market.

Software testing as a field of interest will only continue to grow as we increasingly rely on software products in our daily lives. This special issue highlights some emerging techniques and provides motivation for future work that will not only develop solutions and tools but will also provide additional guidance about how to apply them.

Renée Bryce is an associate professor in the Department of Computer Science at the University of North Texas. Her research focuses on software testing, including combinatorial testing, test suite prioritization, and test suite reduction. Bryce received an MS in computer science from Rensselaer Polytechnic Institute and a PhD in computer science from Arizona State University. She is a member of IEEE. Contact her at renee.bryce@unt.edu.

Rick Kuhn is a computer scientist at the US National Institute of Standards and Technology. His research focuses on software assurance, including combinatorial testing, empirical studies of software failure, and access control. Kuhn received an MS in computer science from the University of Maryland at College Park and an MBA from the College of William & Mary. He is a senior member of IEEE. Contact him at kuhn@nist.gov.