Guest Editorial:
Special issue on Automation of Software Test: Improving Practical Applicability

Christof Budnik, Siemens Corporation, USA, christof.budnik@siemens.com

Gordon Fraser, University of Passau, Germany, gordon.fraser@uni-passau.de

Francesca Lonetti, ISTI-CNR, 56127, Pisa, Italy, francesca.lonetti@isti.cnr.it (corresponding author for receiving proofs)

Hong Zhu, Oxford Brookes University, UK, hzhu@brookes.ac.uk

Test Automation

An important challenge in software testing has been to improve the degree of attainable automation, in developing advanced techniques for generating the test inputs as well as in finding innovative support procedures to automate the testing process.

Test automation is a very active field of research, covering different aspects such as tools, application domains, the lifecycle development process, project dimensions, and empirical work reporting successful results or failures [1]. Test automation is also highly relevant in practice: The market for automated test support tools expanding, opening relevant business opportunities for new innovative testing platforms. The ultimate challenge is the development of a powerful integrated test environment that goes beyond automated test execution, and by itself can automatically take care of selection, deployment and integration of the testing tools that assist test activities across the software testing process [2].

A high level of automation raises many challenges: The analysis of the large amount of test results produced automatically creates cost and effort. The trade-off between automated and manual testing has been highly discussed in literature [3] and decision support systems for selecting most effective and efficient testing tools for specific purposes in a specific context are provided [4]. Costs can be reduced by automating even the test automation, for example by using natural language test steps enabling a sequence of procedure calls with accompanying parameters that can drive testing without human intervention. This technique has been proven effective in reducing the cost of test automation by automating over 82% of the steps contained in a test suite [5]. Finally, with a wider acceptance of test automation, the quality of test code or test scripts that perform test automation becomes a major concern in practice and recently an active topic in research.

This special issue

This special issue focuses on a number of practical applicability aspects of test automation, including test suite performance, tester profile, domain specific language implementations as well as test model extraction and user interface testing, reflecting the frontier in research and the best practice in industry. It includes revised and extended versions of best papers presented at the 11th IEEE/ACM International Workshop on Automation of Software Test (AST 2016), held in conjuction with the

38th International Conference on Software Engineering (ICSE'16), in Austin, TX, USA, May 14 - 22, 2016, as well as new original submissions.

This issue initially received a total of 13 submissions. Of these, 3 were successively withdrawn, and after a rigorous peer-review according to the journal's high standards, 4 papers have been rejected and 6 accepted.

This issue consists of the 6 papers that are briefly discussed as follows:
We have two papers looking at test prioritization, an approach which orders test cases to reduce the costs of finding faults. In “Similarity-Based Prioritization of Test Case Automation”, Daniel Flemström, Pasqualina Potena, Daniel Sundmark, Wasif Afzal and Markus Bohlin, develop a prioritisation technique that reuses already automated parts of test cases. In “Test Case Prioritization Techniques for Model-Based Testing: A Replicated Study”, João Felipe Silva Ouriques, Emanuela Gadelha Cartaxo and Patrícia Duarte Lima Machado, present industrial case studies showing the factors influencing the performance of test cases prioritization techniques in the context of model-based testing.

There are three papers looking at test automation in different domains: In “Automated Testing of DSL Implementations - Experiences from Building mbeddr”, Daniel Ratiu, Markus Voelter and Domenik Pavletic, present their experience on testing different aspects of the implementation of domain specific languages and associated tools, aiming at increasing the automation of language testing. In “Model extraction and test generation from JUnit test suites”, Pablo Lamela Seijas, Simon John Thompson and Miguel Ángel Francisco Fernández, describe how to infer state machine models from legacy unit test suites, and how to generate new tests from those models. In “Mobile GUI Testing”, Inês Coimbra Morgado and Ana C. R. Paiva, present a tool for automating testing of mobile applications.

Finally, test quality is the topic of “An assessment of operational coverage as both an adequacy and a selection criterion for operational profile based testing” by Breno Miranda and Antonia Bertolino, who introduce the operational coverage concept that takes into account how much the program's entities are exercised in practice, in order to reflect the usage profile into the coverage measure.

Acknowledgments

We thank the authors and reviewers of this special issue for their great effort and contributions. We also thank the Software Quality Journal production team for their hard work in assembling and editing the issue.

References

[1] Dorothy Graham and Mark Fewster. Experiences of test automation: case studies of software test automation. Addison-Wesley Professional, 2012.

[2] Garousi V, Elberzhager F. Test automation: not just for test execution. IEEE Software. 2017 Mar;34(2):90-6.

[3] Garousi V, Mäntylä MV. When and what to automate in software testing? A multi-vocal literature review. Information and Software Technology. 2016 Aug 1;76:92-117.

[4] Raulamo-Jurvanen P. Decision support for selecting tools for software test automation. ACM SIGSOFT Software Engineering Notes. 2017 Jan 5;41(6):1-5.

[5] S. Thummalapenta, S. Sinha, N. Singhania, and S. Chandra. Automating test automation. In 34th International Conference on Software Engineering (ICSE), pages 881-891, June 2012.