Automating TEST Case Design, Selection and Evaluation Report on 10 Editions of A-TEST Workshop

Tanja E. J. Vos
Open Universiteit
Universidad Politécnica de Valencia
tvs@ou.nl,tvos@dsic.upv.es

I. S. W. B. Prasetya
Utrecht University
s.w.b.prasetya@uu.nl

Sigrid Eldh
Ericsson
sigrid.eldh@ericsson.com

Sinem Getir
Humboldt-Universität zu Berlin
sinemgetir@gmail.com

Ali Parsai
Universiteit van Antwerpen
ali.parsai@uanwerpen.be

Pekka Aho
Open Universiteit
pekka.aho@ou.nl

ACM Reference Format:
Tanja E. J. Vos, I. S. W. B. Prasetya, Sigrid Eldh, Sinem Getir, Ali Parsai, and Pekka Aho. 2019. Automating TEST Case Design, Selection and Evaluation Report on 10 Editions of A-TEST Workshop. In *Proceedings of* . ACM, New York, NY, USA, 4 pages.

DOI: 10.1145/3375572.3375578
http://doi.acm.org/10.1145/3375572.3375578

1 INTRODUCTION

Trends such as globalisation, standardisation and shorter life-cycles place great demands on the flexibility of the software industry. In order to compete and cooperate on an international scale, a constantly decreasing time to market and an increasing level of quality are essential. Testing is at the moment the most important and mostly used quality assurance technique applied in industry. However, the complexity of software and hence of their development amount is increasing. Modern systems get larger and more complex, as they connect large amounts of components that interact in many different ways and have constantly changing and different types of requirements (functionality, dependability, usability, performance etc.). Data processing that impacts all aspects of our life is increasingly distributed over clouds and devices. This leads to new concerns, such as availability, security, and privacy, which are aspects that also needs to be tested. Consequently, the development of cost-effective and high-quality systems opens new challenges that cannot be faced only with traditional testing approaches, and specifically manual testing is simply insufficient and unreliable to manage the speed needed, and ensure the coverage of ever-changing systems. New techniques for systematization and automation of testing throughout the software and system life-cycle are required.

Even though many test automation tools are currently available to aid test planning and control as well as test case execution and monitoring, all these tools share a similar passive philosophy towards test case design, selection of test data and test evaluation. They leave these crucial, time-consuming and demanding activities to the human tester. This is not without reason; test case design and test evaluation through oracles are difficult to automate with the techniques available in current industrial practice. The domain of possible inputs (potential test cases), are even for a trivial method, program, model, user interface or service, typically too large to be exhaustively explored. Consequently, one of the major challenges associated with test case design is the selection of test cases that are effective at finding flaws without requiring an excessive number of tests to be carried out. Automation of the entire test process requires new thinking that goes beyond test design or specific test execution tools. Combining different technologies in new ways and striving towards autonomous testing brings new insight in testing that can reduce the cost of software systems.

These are the problems that this *international Workshop on Automating Test Case Design, Selection and Evaluation* (A-TEST) wants to attack. A-TEST aims to provide a venue for researchers as well as the industry to exchange and discuss trending views, ideas, state of the art, work in progress, and scientific results on automated test case design, selection and evaluation.

A-TEST has successfully run 10 editions since 2009. The first editions took place at the Conference on Information Systems and Technologies (CISTI)[1] and three editions at the Federated Conference on Computer Science and Information Systems (FEDCSIS)[2]. Due to the success of the event and the clear fit with the programme of the international symposium on Foundations of Software Engineering (FSE), the past four years we had the 7th, 8th, 9th and 10th edition of A-TEST together with FSE.

2 THEMES, GOALS AND TOPICS

This workshop's main objective is to provide researchers and practitioners a forum for exchanging ideas, experiences, understanding of the problems, visions for the future, and promising solutions to the problems in automated test case generation, selection and evaluation for modern information systems. This way the workshop also provides a platform for researchers and developers of testing tools to work together to identify the problems in the theory and practice of test automation and to set an agenda and lay the foundation for future development. An open attitude to multiple aspects of automation has been held throughout the years.

Topics of the workshop include:

[1] http://www.aisti.eu/
[2] http://www.fedcsis.org

- Techniques and tools for automating test case design, generation, and selection, e.g. model-based approaches, combinatorial-based approaches, search based approaches, symbolic-based approaches, as well as static, dynamic and statistical analysis and machine learning methods.
- New trends in the use of machine learning and artificial intelligence to improve test automation.
- Test cases optimizatio, selection and reduction.
- Test cases evaluation and metrics.
- Test cases design, selection, and evaluation in emerging domains, e.g. Graphical User Interface, Social Network, Cloud, Gaming, Security, Apps. and Cyber Physical Systems.
- Case studies that have evaluated an existing technique or tool on real systems, not only toy problems, to show the quality of the resulting test cases compared to other approaches.

3 FORMAT OF THE WORKSHOP

Although throughout the years we have changed the format a bit, we have mostly invited the following types of paper submissions. **Position papers** (max. 2 pages) that analyze trends in A-TEST and raise issues of importance and challenges from industries that are not resolved. Position papers are intended to generate discussion and debate during the workshop, and will be reviewed with respect to relevance and their ability to start up fruitful discussions. **Work-in-progress papers** (max. 4 pages) that describe novel, interesting, and highly potential work in progress, but not necessarily reaching its full completion. **Full papers** (max. 10 pages) describing original and completed research – either empirical or theoretical – in A-TEST techniques, tools, or industrial case studies. **Tool papers** (max. 4 pages) presenting some academic testing tool in a way that it could be presented to industry as a start of successful technology transfer.

4 HANDS-ON SESSIONS

During the years 2017, 2018, and 2019, we included hands-on sessions in the program of our workshop. These sessions turned out to be a great instrument to (i) getting the audience involved with state-of-the-art technology, and (ii) getting the proof-of-concept providers to discover the trade-offs associated with their technique (and/or tool) using a mixed audience of academics and practitioners of the field. Consequently, the hands-on session format with a short background theory section, a long hands-on practical session, and a short final feedback session, works very well. The following tools have been presented in the past editions of the workshop:

A-TEST 2017

Automated GUI Testing with TESTAR. TESTAR is a tool for automated scriptless testing at the GUI level [2]. During the session, the participants installed the tool on a virtual machine, and follow the provided step-by-step tutorial to complete the given tasks using the TESTAR tool. By the end of the session, the participants have learned about scriptless GUI testing, and gained experience with the TESTAR tool and it use of real-life software.

Model-Based Testing with Axini Toolset. Axini is a company specializing in model-based testing (MBT) for complex industrial software. At Axini, MBT is considered the next step in test automation. The crux of MBT is in the modeling. During the hands-on session, the participants could try out the Axini toolset to model a simple system, and automatically generate and execute test cases for the system under test (SUT). The participants were given an opportunity to use the commercial Axini toolset in a close-to-reality setting, and by the end of the session, they attained a bird's eye view of what it means to model, generate, and execute tests and analyse the results.

A-TEST 2018

Mutation Testing with LittleDarwin. LittleDarwin is a Java mutation testing tool that mainly targets fast deployment in complicated software environments [6]. In this hands-on session, the participants used LittleDarwin to perform mutation testing on an educational software project, and by following the step-by-step tutorial they familiarized themselves with the process of mutation testing. By the end of the tutorial, the participants gained insight about test suite quality, why the simple metrics such as statement or branch coverage are not enough, and the ways to mitigate weaknesses and improve the quality of their tests.

A-TEST 2019

Amplifying Integration Tests with CAMP. CAMP is a tool that amplifies integration tests. Given a deployable SUT and a description of what changes to explore, CAMP generates alternative configurations, each having the SUT deployed in a different environment. In this hands-on session, the participants used CAMP to amplify tests of a simple web-service. By the end of session, the participants learned about integration testing of web services, and how to automatically amplify the tests for such a system.

Visualizing Test Results in Unity3D with TestViz-City. TestViz-City is a visualization prototype tool that shows test results from running several iterations of regression testing [3]. It represents test results in a city landscape, giving the user an easy way to find the error-prone areas of the code. In this hands-on session, the participants used the results of the regression testing of a real-life embedded system project, and attempted to replicate the analysis of testing experts using this visualization tool. By performing this task, they learned how to use the visualization tools at their disposal in an effective manner to analyze SUT.

5 OVERVIEW OF RESEARCH TOPICS

In Figure 1 an overview is given of the topics that have been presented over the 10 years. In each category the topic with the most papers related is at the top.

6 INSIGHTS FROM SUBMISSIONS

6.1 Testing techniques

A-TEST has covered the whole range of testing techniques: random, combinatoric, property based, model based, and search based. Model based testing (MBT) is a subject that is present in almost every year of A-TEST. Despite being a technology that has been around for several decades –introduced back in 90's [4]–, there are still plenty of open challenges in MBT. For example, in practice models can easily become large and complex, for which simple node coverage might not be good enough to expose non-obvious bugs. In

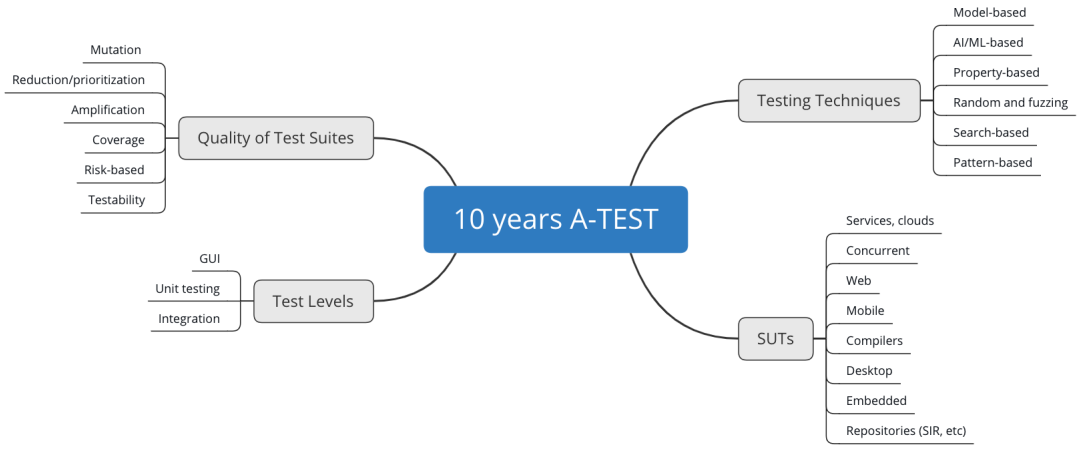

Figure 1: Overview of topics over 10 years of A-TEST.

A-TEST we have seen works that seek to apply e.g. a combinatoric approach to obtain stronger test suites, and prioritization to keep the cost manageable. Better tests can be generated if the models are made richer, e.g. by capturing the parameters of the transitions, and their pre- and post-conditions. This however makes test generation a harder problem, for which pure graph-based traversal algorithms would not suffice. In the A-TEST we have seen works applying search-based algorithms, even in combination with symbolic reasoning, to address this problem. Another challenge is that models can be incomplete, or become incomplete because the software under test evolves. In recent years we have seen increasing attention to AI and machine learning in A-TEST. Although they were mainly used for searching effective tests, it seems an interesting future direction to train AI algorithms to do model maintenance tasks (e.g. to smartly evolve models when they become obsolete). In addition to work in learning AI, we also hope to see more contribution from cognitive AI in the future, either to improve existing testing techniques, or to derive a completely new class of testing techniques.

6.2 Quality of Test Suites

The quality of generated test suites gain importance in the recent years within various techniques. Mutation testing is a state of the art technique in the last decades. The quality of mutation testing in terms of both reduction of redundant mutants and efficient computing are the focus in the last three years of A-TEST. Reduction approaches is another technique to increase the quality of test suites, although we observe little in the last years. Techniques such as coverage, amplification as well as debugging are means to increase quality appearing since 2016. In summary, the quality aspect of test suites appear more as research topics recently in time.

6.3 Systems Under Test (SUTs)

The studies published in A-TEST cover a variety of systems under test. From concurrent systems to cloud-based applications, desktop, mobile, or embedded systems, compilers or web applications. While the diversity of SUTs in each edition of A-TEST has been high, there is a trend towards mobile and web application systems. This coincides with the growing importance of such systems in the developer world, given the fact that most end-users are moving towards mobile platforms [1]. The importance of testing in such platforms comes from the fact that on the one hand, they are more costly and difficult to test due to variety in platforms, and on the other hand these systems are expected to be robust and performant [5]. The shift towards automated testing of mobile and web systems is apparent in the recent editions of A-TEST, since there has been at least one published study on mobile and web systems in the past 4 editions of A-TEST. We expect this trend in A-TEST to only grow as the global trend towards mobile and web systems continues, and the testing problems in this domain remain a popular subject for researchers.

6.4 Test levels

During the last decade, there seems to be a shift from automating the low level tests, such as unit tests, to automating the higher level tests - testing the entire system. Great examples are e.g. GUI tests - testing through the user interface, but also utilizing technologies that can challenge a larger code base in a more black-box fashion. In practise, unit tests complemented by coverage and evolved static analysis support has made developers in agile teams develop full automation, which is common practice - some also use TDD. Nevertheless, unit test automation a now a normal activity. The research papers that still address unit test level, are concentrating on mutation testing to evaluate the quality of the unit tests, or more advanced techniques for unit test code generation. Similarly we can now see functional tests being automated, followed by non-functional (extra-functional properties). This automation also allows for new challenges to emerge - as well as new solutions. A new concern is the ever growing size of test code (automated tests) that is now easily surpassing the size of software code, currently resulting in test selection, optimization and prioritization together with reduction of test size in general.

7 CONCLUSIONS

A-test has proven itself during this passed 10 years to be an important workshop in the field of software testing, and its tool development, allowing new meetings and discussions of problems and solutions and contributing to the research direction in the field.

8 ACKNOWLEDGEMENTS

A-TEST started within the FITTEST project that was funded by the EU under contract number FP7-ICT-257574 and finished in 2014. A-TEST would not have been possible without the continuous support of the University of Utrecht, The Universidad Politecnica de Valencia, the Open Universiteit The Netherlands and Ericsson. Even if there was not project to support the workshop, these organisations made it possible. As from the year 2017, A-TEST has also been supported by the TESTOMAT project, an ITEA 3project under contract number 16032.

REFERENCES

[1] Personal computing device shipments forecast to continue their slow decline with a five-year compound annual growth rate of -1.2%, according to idc, Mar 2019. https://www.idc.com/getdoc.jsp?containerId=prUS44908319.

[2] S. Bauersfeld, T. E. J. Vos, Nelly Condori-Fernández, Alessandra Bagnato, and Etienne Brosse. Evaluating the TESTAR tool in an industrial case study. In *2014 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '14, Torino, Italy, September 18-19, 2014*, page 4, 2014.

[3] Markus Borg, Daniel Brytting, and Daniel Hansson. Enabling visual design verification analytics - from prototype visualizations to an analytics tool using the unity game engine. *CoRR*, abs/1808.09767, 2018.

[4] Ibrahim K El-Far and James A Whittaker. Model-based software testing. *Encyclopedia of Software Engineering*, 2002.

[5] B. Kirubakaran and V. Karthikeyani. Mobile application testing — challenges and solution approach through automation. In *2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering*, pages 79–84, Feb 2013.

[6] Ali Parsai, Alessandro Murgia, and Serge Demeyer. *LittleDarwin: A Feature-Rich and Extensible Mutation Testing Framework for Large and Complex Java Systems*, pages 148–163. Springer International Publishing, Cham, 2017.