Overview of the ICST International Software Testing Contest

Emil Alégroth
Blekinge Institute of Technology
Sweden

Shinsuke Matsuki
Veriserve Corporation
Japan

Tanja E. J. Vos
Open Universiteit
The Netherlands

Kinji Akemine
NTT DATA Corp
Japan

***Abstract*—In the software testing contest, practitioners and researcher's are invited to test their test approaches against similar approaches to evaluate pros and cons and which is perceivably the best. The 2017 iteration of the contest focused on Graphical User Interface-driven testing, which was evaluated on the testing tool *TESTONA*. The winner of the competition was announced at the closing ceremony of the international conference on software testing (ICST), 2017.**

***Keywords*-gui testing; automated testing; comptetition;**

I. Introduction

The evaluation of tools through competition in an academic context, e.g. a conference, has previously been successful in providing usefull feedback to the developers [1]. Feedback that helps them to improve their tools as well as guide future development and research efforts by spreading awareness of the tools and their capabilities to academic researchers.

What started at ICST with the "bug bash" in 2016, has now grown into a tool competition. More specifically, a tool competition focused on tools for testing a system through its Graphical User Interface (GUI).

GUIs represent the main connection point between a software's components and its end users and can be found in almost all modern applications. This makes them an attractive interface for testers to use, especially since testing at the GUI level means testing from the user's perspective and can thus verify the System Under Test's (SUT) behavior. Currently in practice, GUIs can account for 45-60% of the entire source code [2] and are often large, complex and can lack technical interfaces that allow them to be accessed programmatically. These properties can pose a great challenge for a SUT's testability and may be why test automation is not performed of a SUT. Instead, testing is performed with costly, tedious and error prone manual test practices [3], [4].

Many test automation tools and techniques do however exist [5]–[9]. Tools and techniques that have been successfully applied by researchers and practitioners to find severe faults at low effort on the part of the tester. In this competition we challenge these tools' capabilities, in a common setting, to evaluate their pros and cons and identify a winner!

II. Testing at the GUI level

GUI-level testing can be divided into several chronologically defined generations. The first generation focused on exact coordinate interactions whilst the second uses technical interfaces into the SUT. Lastly, the third generation uses image recognition to interact with and assert the SUT as it is displayed to the user. Regardless, all generations have in common that they test the SUT through the elements available at the GUI. Further, whilst most GUI-testing tools are used for regression testing, exploratory testing tools also exist. The goal of this competition is to provide a context where all of these types of tools/techniques can be compared.

III. The SUT: TESTONA

The SUT used in the contest is another testing tool called *TESTONA*,[1] a commercial software tool implementing the classification tree method [10] developed by the company Berner & Mattner.[2]

The classification tree method offers a graphical notation for specifying test parameters. For each influence factor (e.g. parameter) of the SUT, a *classification* is added to a classification tree. For each classification, representative *classes* are then added following the principles of boundary value analysis and equivalence class partitioning [11]. This results in a tree consisting of classifications and classes. Test cases are then defined by selecting exactly one class for each classification.

TESTONA thereby allows automatic generation of combinatorial test suites [12]. A screenshot of the tool is shown in Figure 1. The tool is tested within B&M with two kinds of GUI test techniques, due to historical reasons. A large body of *legacy* tests exists, consisting of 246 test scripts implemented with QF-Test,[3] used as a conventional capture and replay tool in our setup. These test scripts are used for regression testing and are based on a functional specification of the tool.

More recently, B&M has also started to implement GUI tests using Eclipse SWTBot.[4] A total of 2413 SWTBot tests of the tool's functional behavior have been produced based on 36 parametrized scenarios in terms of classification trees. In addition, the tool is tested with a set of performance tests as well as standard unit tests.

IV. Format of the competition

In preparation for the competition, all participants were provided with the defect free version of the SUT described

[1] http://www.testona.net/
[2] http://www.berner-mattner.com/
[3] http://www.qfs.de/en/qftest/
[4] http://www.eclipse.org/swtbot/

978-1-5090-6031-3/17 $31.00 © 2017 IEEE
DOI 10.1109/ICST.2017.67

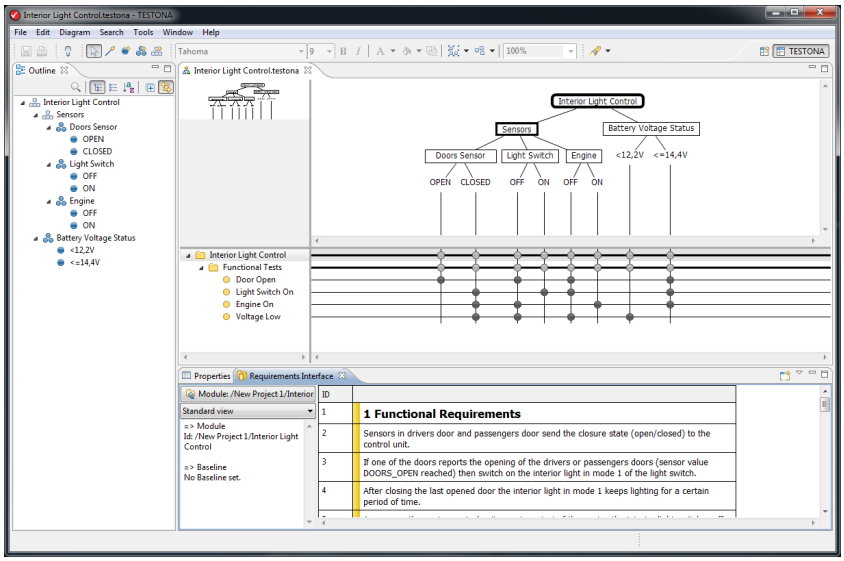

Fig. 1: The SUT: TESTONA

in the previous section. This allowed contestants, especially contestants with more regression-orientated tools, to create test cases, models and/or fine-tune their tools/techniques in preparation for the main competition in Japan.

To make sure we can compare the efforts spent by competitors, each participant was asked to maintain a detailed working diary where they logged how much time they spent on the different prepatory tasks. These inputs will be crucial additional inputs that decides whether we are able to compare and evaluate the tools and techniques industrial viability (effectiveness, efficiency and usage) after the competition.

During the competition, participants could also drop in and compete but all contestants had the same amount of time to find as many faults (failures or defects) as possible in the competition SUT. The competition SUT consisted of a manually mutated version of *TESTONA* with a significant number of mutants seeded on different levels of SUT abstraction. Additionally, due to being a commercial software, the SUT was perceived to contain real faults as well. **Note:** *The number of found faults and severity of said faults were evaluated, contra time/effort spent, to determine the winner of the competition.* The winner of the competition was announced at the closing ceremony of the conference.

V. Future Editions

In future editions of the test contest we strive to build upon the experiences from ICST 2017 and also expand upon the tools/techniques evaluated in the competition. As such, we expect the "theme" of the competition to vary from year to year and give all researchers, regardless of test-related interest, the opportunity to test their approach against similar approaches.

Acknowledgement

The organizational committee would like to, in particular, thank Peter M. Kruse and Berner & Mattner. Additionally, we wish to thank all who participated in the competition.

References

[1] U. Rueda, R. Just, J. P. Galeotti, and T. E. J. Vos, "Unit testing tool competition: Round four," in *Proceedings of the 9th International Workshop on Search-Based Software Testing*, ser. SBST '16. New York, NY, USA: ACM, 2016, pp. 19–28. [Online]. Available: http://doi.acm.org/10.1145/2897010.2897018

[2] A. M. Memon, "A comprehensive framework for testing graphical user interfaces," Ph.D. dissertation, University of Pittsburgh, 2001.

[3] M. Grechanik, Q. Xie, and C. Fu, "Maintaining and evolving GUI-directed test scripts," in *Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on*. IEEE, 2009, pp. 408–418.

[4] M. Finsterwalder, "Automating acceptance tests for GUI applications in an extreme programming environment," in *Proceedings of the 2nd International Conference on eXtreme Programming and Flexible Processes in Software Engineering*. Citeseer, 2001, pp. 114–117.

[5] T. E. Vos, P. M. Kruse, N. Condori-Fernández, S. Bauersfeld, and J. Wegener, "Testar: Tool support for test automation at the user interface level," *Int. J. Inf. Syst. Model. Des.*, vol. 6, no. 3, pp. 46–83, Jul. 2015. [Online]. Available: http://dx.doi.org/10.4018/IJISMD.2015070103

[6] E. Alegroth, M. Nass, and H. H. Olsson, "Jautomate: A tool for system- and acceptance-test automation," in *Software testing, verification and validation (icst), 2013 ieee sixth international conference on*. IEEE, 2013, pp. 439–446.

[7] T. Yeh, T.-H. Chang, and R. C. Miller, "Sikuli: using gui screenshots for search and automation," in *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. ACM, 2009, pp. 183–192.

[8] B. N. Nguyen, B. Robbins, I. Banerjee, and A. Memon, "Guitar: an innovative tool for automated testing of gui-driven software," *Automated Software Engineering*, vol. 21, no. 1, pp. 65–105, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10515-013-0128-9

[9] A. Holmes and M. Kellogg, "Automating functional tests using selenium," in *Agile Conference, 2006*. IEEE, 2006, pp. 6–pp.

[10] M. Grochtmann and K. Grimm, "Classification trees for partition testing," *Softw. Test., Verif. Reliab.*, vol. 3, no. 2, pp. 63–82, 1993.

[11] G. J. Myers, *The Art of Software Testing*. John Wiley and Sons, 1979.

[12] P. M. Kruse and M. Luniak, "Automated test case generation using classification trees," *Software Quality Professional Magazine*, vol. 13, no. 1, 2010.