Framework for Better Efficiency of Automated Testing

Work-in-Progress Paper

Martin Filipsky, Miroslav Bures and Ivan Jelinek
Department of Computer Science and Engineering
Czech Technical University
Prague, Czech Republic
{filipma2, buresm3, jelinek}@fel.cvut.cz

***Abstract*—The paper introduces our design of a framework for test automation, which utilizes both recording and scripting approaches to help quality-assurance engineers with their test automation efforts. Characteristic problems of test automation are a low efficiency and/or high maintenance costs. The framework cuts down those drawbacks using a technique of abstraction, a clever structure of test cases, and reusable common test case retrieval from recorded tests. Finally, the approach is robust to a test script ageing and is technology-independent as well as testing-tools independent.**

Keywords-automated testing; functional tests; quality assurance; test efficiency; test recording

I. INTRODUCTION

With the increasing importance and complexity of information systems, software vendors and system integrators are facing numerous challenges. Current technologies are rapidly changing. New technologies and ideas like Cloud computing [9], Virtualization [5], or Software as a service [4] are coming; but, budgets intended for software products and IT projects are cut down. On the other hand, customers require more and more features, a better performance, a higher reliability, and a first-rate availability for less money. Who wants to compete and be a market leader, has to be more efficient than his competitors. Since testing costs represent a significant part of total costs of development, we focused on areas where we see a potential to improve a testing process using test automation.

The area of automated testing of software applications is facing a number of issues and challenges like an insufficient expertise to automate tests, technological issues, and/or demand on a fast and effective test development and execution. One of the most serious problems revolving around test automation is that test scripts are getting inaccurate and obsolete with changes in an application under test. As the result, automated testing is limited mostly to regression tests, smoke tests and performance testing. A utilization of automated testing in a sense of a replacement of manual testing is not quite often.

Even solutions of some issues (automated test development/debugging, or maintenance [18], [13]) may increase the efficiency of the process of test automation. Moreover, a number of defects would decrease in final applications released to production environments. Some techniques used to speed up the process of test automation and to increase the efficiency are already adopted by many quality-assurance (QA) teams, like using Mockups [20] or generating test cases from application models [23]. Mockups may be very useful when QA teams have enough time to prepare and develop tests against an application prototype. Agile software development methodologies [17] bring additional requirements on a development of automated tests. Test recording approaches are very useful in such cases because they do not require a time-consuming preparation. Testers can start to develop automated tests immediately. Furthermore, if those test recordings are transformed into abstract tests organized in test suites, the resultant effort and costs will decrease significantly.

The main goal of our research is a definition and a validation of a meta-model among requirements on automated tests, an application under test (AUT) and test scripts to record test cases using the defined meta-model and to create a structure of tests having common parts of the recorded tests broken down into reusable pieces for an easier and cheaper test maintenance.

This paper is structured as follows: We start by giving an overview on related research in Section 2. In Section 3, we describe the concept of the framework. In Section 4, we conclude with a summary and an outlook.

II. RELATED WORK

Many research teams are interested in test automation of various kinds like unit testing, functional, GUI or regression testing, and performance testing. Unit test automation is being successfully theoretically covered. Some approaches focus on a generation of test data [8] or a generation of test cases [23] from models of AUTs. For example, Xu [23] introduced an approach based on high-level Petri nets as finite state test models for an automated test generation and a test execution. On top of that, he developed a model-based Integration and System Test Automation tool. This premise may be seen as a drawback in agile software methodologies where specifications and models frequently do not exist at the moment when the functional test automation is required. For a generation of test cases, a detailed model of AUT including use case diagrams and user scenarios is required prior the testing process can start. Those generated test cases might not be possible to execute without additional pre-processing. Missing object IDs may prevent from that as the objects cannot be identified in the AUT. For example, if a development team uses dynamic objects with insufficient

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-230-1

unique object properties or with dynamically generated IDs. Approaches were introduced for web application modelling in [2], [16], and [22].

Koopman, Plasmeijer and Achten [15] introduce a model-based testing system for on-the-fly testing of thin-client web applications specified by Extended State Machines. The approach is proposed for plain HTML (no Flex, no JavaApplets). They do not discuss more general solutions for rich clients and/or other platforms, like mainframe panels (applications for terminal emulators), which leads to technological limitations (e.g., different objects, and application behavior) of the approach. Beek and Mauw [1] present an approach for a conformance, black-box testing of thin internet applications. Besson, Beder and Chaim [3] introduced an approach of test automation for acceptance testing. In general, approaches based on Finite State Machines (FSM) are quite often as presented Jia and Liu [14].

In comparison to test automation efforts based on model-based approaches, where test scripts are generated from models, Stepien, Peyton and Xiong [21] describe a testing of web applications using TTCN-3 language [6] and [19] which is a specification-based approach. An XML specification-based approach of testing of web applications is presented by Jia and Liu [14]. The specification-based approaches [6], [10], [19], and [21] are close to our intentions, but they did not utilize the test recording in their approaches.

The mentioned approaches are either limited by a technology or they require a model, or another detailed specification of AUT. In comparison to the mentioned approaches, we are focusing on a feature to build a structure of test cases on-the-fly while the user is recording tests. Furthermore, the proposed framework recognizes input data and uses them as test parameters. Both the features are a key to a reusability of tests or parts of tests, which might significantly decrease total costs of test maintenance.

García, Dueñas, and Parada introduced recently a couple of approaches for automated functional testing based on the navigation of web applications in [10], [11], and [12]. Their concept is to automate functional tests using UML diagrams [7] as an input for an automated test case generation driven by the navigation in the AUT. Compared with our proposed technique, a model or at least a part of the model of the AUT has to be available before the testing actually starts. They also presented an alternative to the test case generation for agile strategies. They can record tests in order to skip a phase of formal design. Unlike the presented approach, we lay emphasis on the feature of a detection of reusable test parts.

III. PROPOSED SOLUTION

Standard approaches of functional test automation are usually based on:

1. A plain test recording of test cases.
2. Test scripting using a programming language.
3. An automation framework.

A facility for test recording records a user activity while testing AUT and the resultant test is captured in any programming language, e.g., in Java. If an advanced technique is used like an object repository, the tests are relatively fine but the maintenance is getting more difficult in comparison to projects utilizing scripting and/or different automation frameworks (code redundancy, no code optimization, object duplicity, and usually no parameters among tests).

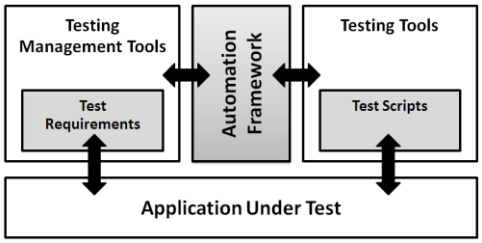

Figure 1. High-level architecture of the proposed solution.

Second approach is typically for experienced QA engineers. It is time-consuming and requires being well prepared. On the other hand, testers can fully utilize a power of object oriented programming and can develop highly reusable code (tests) with low maintenance costs.

The automation frameworks are driven by data (a flow is controlled by input data), by keywords (the flow usually does not depend on input data, test scripts completely control a test run) and by a model (the test flow depends on a model of AUT). The model-driven frameworks are worth in cases where the system changes dramatically and new test scripts can be easily regenerated from the model. Hybrid approaches are common (e.g., Data-Keyword, but not Recording-Scripting), and they combine the best features of single approaches.

The challenges described above lead us to several areas of our interest. We are working on a definition of a meta-model of test cases which is a key premise of an efficient test recording. The meta-model will be used in a process of building a smart structure of tests from the recorded test cases. Another field of our interest is a reusability of recorded tests. We are working on algorithms that will identify common test parts in the structure of tests and reorganize them.

Our intention is to integrate solutions of single problems in one test automation framework (Fig. 1) among test requirements, testing tools and AUTs. The framework is intended to be modular as well as platform-independent. We are focusing on a utilization of benefits of both the recording and scripting approaches (i.e., fast test automation and good maintenance costs). There are two options how the automation framework should support user efforts to automate test cases (Fig. 2). Either the user does not have a test case automated yet or the user has already automated a test case. In the first case, the user records the test case, which is converted on-the-fly into the meta-model, using the recording facility. All relevant objects (buttons, links, data, strings etc.) are currently captured into an object repository without additional duplicities. In the second case, the automation framework records a new sequence of

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-230-1

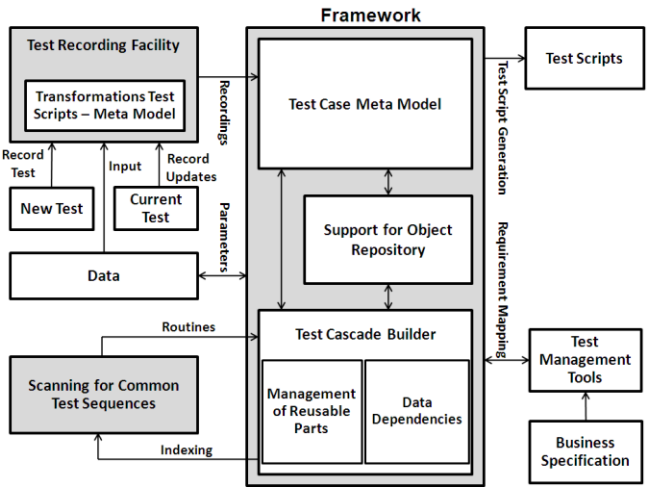

Figure 2. The concept of the framework based on test recording.

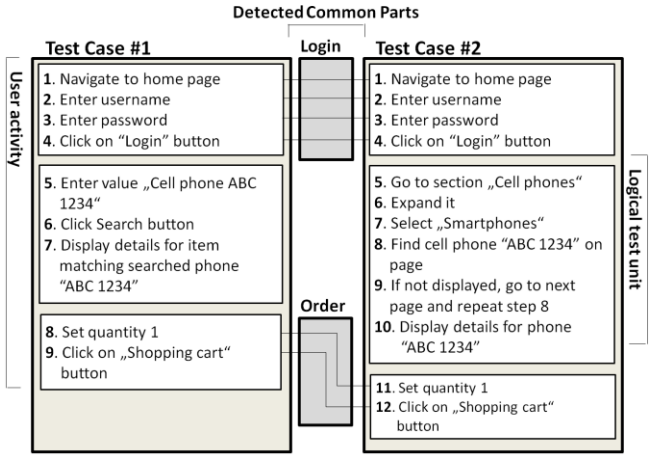

Figure 4. An example of mapping steps in common test sequences between two test cases.

user activities while the user is executing altered parts of the test case. The user selects a relevant part of the test case to be updated and the new sequence of recorded steps is mapped into the current test and all relevant parts of other recorded tests, which use the updated part. If the user records more than two test cases in the test suite, the framework detects common parts, which are typically frequent and repeating activities like a login to the AUT. Therefore, we let the user record the complete test suite in order to run scanning algorithms, which will detect common test sequences in the recorded test cases. The detected common test sequences can be excluded from the test suites afterwards and represented as new subunits of test cases as it is presented on Fig. 3.

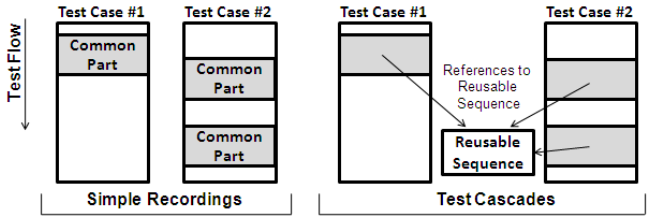

Figure 3. Test Cascades. Gray parts identified by scanning algorithms represent the common parts in test cases.

For instance, if each test case contains a login to a system, the procedure of logging in can be extracted and called from all test cases automatically. One change in the common test sequence (reusable part) takes effect in all calling tests. An example is shown on Fig. 4. Since we need to get an organized structure of test cases with reusable test parts for a better test maintainability, we have to find longest common subsequences of user activities in the test suite. In other words, we have to find a mapping of steps among all tests in the test suite. If such a mapping exists, those steps can be excluded from tests and the new reusable part, which is referenced from these tests, can be created. Obviously, a problem might to find a suitable level of a step count in the sequence. Remaining problems to be solved are questions of handling data dependencies, a problem of a test parameterization and a problem of recursive calls to reusable units.

In addition to the recording, the user has still an option to design test cascades manually (consider test cascades as an automatically generated structure of test cases from test recordings) and to design and script test cascades in a domain-specific language (DSL). In DSL, we specify the test cases from the end user's point of view and it enables to work with the meta-model of test cases. Syntax of DSL comes out from simple English, but we are also planning a graphical version of the DSL, which will be identical to the standard text version represented by Tab. 1.

TABLE I. AN EXAMPLE OF THE DOMAIN-SPECIFIC LANGUAGE FOR A TEST CASE REPRESENTATION

Object Type	Object Name	Action	Parameters	Recovery Scenario
Browser	MyBrowser	Open	www.cvut.cz	CloseIfNot Available
Button	Submit	Click		StopTest
Table	MyTable	Validate		StopTest
TextBox	Search	Set	Test automation	SkipStep

A requirement on simple English is beneficial for non-experienced QA engineers who do not know standard programming languages like Java. Thus, they can immediately start to alter their recorded test cases. The second benefit of the DSL is a fact that a complexity of the solution will be hidden for end users. Objects are stored in an object repository. For this object repository, storage available in testing tools like, e.g., HP QuickTest Professional can be used. An alternative is to use an internal object repository of the framework.

The automation framework gives the QA engineers an option of inspection that helps them to detect common mistakes in the design of automated tests. For example, web applications may have different responses depending on system loads, which lead usually to synchronization issues.

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-230-1

The concept of test cases represented by the meta-model with the test cascades provides a robust solution to changes in the application. For example, if a new step is added into a current business process or if an object is altered in the AUT, a modular structure of test cascades supports an update in one place, which will take effect in all places. The higher level of abstraction of test cases together with a support of generation of test scripts for different testing tools helps to prevent tests from test script ageing.

IV. CONCLUSIONS AND FUTURE WORK

We carried out a research of the current state-of-the-art and we have found out that the problem of building reusable test cascades from test recordings is covered insufficiently. There are either approaches for on-the-fly testing (usually limited by a technology like plain HTML), e.g., [1], [3] or [15] or approaches requiring to prepare a model of AUT like in [6], [19] or [20]. A more general concept utilizing the test recording is missing. Based on that, we have designed a framework for automation of functional tests with no need to have a model of the AUT. We have designed the structure of the meta-model, used in the proposed solution.

Currently, we are working on a detailed design of the meta-model. In parallel with that, we are conducting a research of a problem of transformation test recordings i.e., the transformation of test scripts from standard programming languages to the meta-model. Besides that, we are dealing with a problem of seeking out the longest common test sequences. The goal is to build test cascades from recordings with no need of a post-processing. As a consequent task, we are going to solve a problem of test parameterization i.e., to detect input parameters and dependencies among them.

Finally, we are preparing to conduct experimental observations on real industrial projects with comparisons of costs and results of the manual testing, the conventional automated testing using the test recording and/or the plain test scripting, and the automated testing using the proposed automation framework.

REFERENCES

[1] H. M. A. van Beek and S. Mauw, "Automatic Conformance Testing of Internet Applications". In *Lecture Notes in Computer Science*, 2004, Volume 2931, Formal Approaches to Software Testing, Page 1106.

[2] H. M. A. van Beek, "Specification and Analysis of Internet Applications". In *PhD thesis*, Technical University Eindhoven, The Netherlands, 2005. ISBN 90-386-0564-1.

[3] F. M. Besson, D. M. Beder, and M. L. Chaim, "An Automated Approach for Acceptance Web Test Case Modeling and Executing". *Lecture Notes in Business Information Processing*, 1, Volume 48, Agile Processes in Software Engineering and Extreme Programming, Part 2, Pages 160-165.

[4] G. Blokdijk, "SaaS 100 Success Secrets - How Companies Successfully Buy, Manage, Host and Deliver Software As a Service (SaaS)", Emereo Pty Ltd. USA, 2008. ISBN 978-0-9804-7164-9.

[5] M. Cafaro and G. Aloisio (Eds.), "Grids, Clouds and Virtualization". 1st Edition., Springer, 2011. ISBN 978-0-85729-049-6.

[6] ETSI ES 201 873-1: "The Testing and Test ControlNotation version 3", Part1: TTCN-3 Core notation, V3.2.1, February 2007.

[7] M. Fowler, "UML Distilled: A Brief Guide to the Standard Object Modeling Language". Addison-Wesley Professional, 3rd Edition, 2003. IBSN-13: 978-0321193681.

[8] S. Fujiwara, K. Munukata, Y. Maeda, A. Katayama and T. Uehara, "Test data generation for web application using a UML class diagram with OCL constraints". In *Innovations in Systems and Software Engineering*, 2011, volume 7, Number 4, Pages 275-282.

[9] B. Furht and A. Escalante (Eds.), "Handbook of Cloud Computing". 1st Edition., Springer, 2010. ISBN 978-1-4419-6524-0.

[10] B. García, "Contribution to the Automation of Software Quality Control of Web Applications". In *PhD thesis*, Universidad Politécnica de Madrid, Spain, 2011. ID code 9017.

[11] B. García, J. C. Dueñas, "Automated Functional Testing based on the Navigation of Web Applications". WWV 2011, Reykjavik, Iceland, June 2011.

[12] B. García, J. C. Dueñas, H. A. Parada, "Functional Testing based on Web Navigation with Contracts". IADIS International Conference (WWW/INTERNET09). Rome, Italy. Nov. 2009.

[13] D. Hoffman,"Cost Benefits Analysis of Test Automation". STAR West, 1999.

[14] X. Jia and H. Liu, "Rigorous and Automatic Testing of Web Applications". In *Proceedings of the 6th IASTED International Conference on Software Engineering and Applications (SEA 2002)*, pages 280–285, Cambridge, MA, USA, Nov. 2002.

[15] P. Koopman, R. Plasmeijer, and P. Achten, "Model-Based Testing of Thin-Client Web Applications". *Lecture Notes in Computer Science*, 2006, Volume 4262, Formal Approaches to Software Testing and Runtime Verification, Pages 115-132.

[16] F. Lanubile and T. Mallardo, "Inspecting Automated Test Code: A Preliminary Study". In *Lecture Notes in Computer Science*, 2007, Volume 4536, Agile Processes in Software Engineering and Extreme Programming, Pages 115-122.

[17] R. C. Martin, "Agile Software Development, Principles, Patterns, and Practices". 1st Edition., Prentice Hall, 2002. ISBN: 978-0135974445.

[18] B. Pettichord, "Seven Steps to Test Automation Success". STAR West, 1999.

[19] R. L. Probert, B. Stepien, and P. Xiong, "Formal testing of web content using TTCN-3". In *TTCN-3 User Conference 2005*, June 2005.

[20] J. M. Rivero, G. Rossi, J. Grigera, J. Burella, E. R. Luna, and S. Gordillo, "From mockups to user interface models: an extensible model driven approach". In *Lecture Notes in Computer Science*, Volume 6385, Pages 13-24, 2010.

[21] B. Stepien, L. Peyton, and P. Xiong, "Framework testing of web applications using TTCN-3". In *International Journal on Software Tools for Technology Transfer (STTT)*, 2008, Volume 10, Number 4, Pages 371-381.

[22] Y. Wu and J. Offutt, "Modeling and Testing Web-based Applications". GMU ISE Technical ISE-TR-02-08, Information and Software Engineering Department, George Mason University, Fairfax, USA, Nov. 2002.

[23] D. Xu, "A Tool for Automated Test Code Generation from High-Level Petri Nets". In *Lecture Notes in Computer Science*, 2011, Volume 6709, Applications and Theory of Petri Nets, Pages 308-317.

Copyright (c) IARIA, 2012. ISBN: 978-1-61208-230-1