Using Test Recording for Test Automation

Martin FILIPSKY

Dept. of Computer Science and Engineering, Czech Technical University, Technicka 2, 166 27 Praha, Czech Republic

filipma2@fel.cvut.cz

Abstract. *This paper presents an approach for automation of functional tests based on Record and Playback techniques. Currently, our approach enhances capabilities of automation frameworks based on keywords, improves their robustness against test script ageing, and introduces a concept of reusable common components, which are automatically detected. It does not require programming skill from end users as the users work with commands of the framework similar to plain English. Tests to be automated are recorded using the Records & Play facility and to get the full power of frameworks, the recorded test scripts in standard programming languages like Java are converted to the language of the framework. Then, scanning algorithms seek common parts in the tests and create an efficient structure of reusable test parts, resulting in simpler the test maintenance and lower maintenance costs.*

Keywords

Automated testing, test recording, quality assurance, test efficiency.

1. Introduction

The increasing complexity of information systems and applications represents new challenges for software developers. Currently, technologies are rapidly changing, the market requires constantly. New approaches like Cloud computing [9], Virtualization [5], or Software as a Service (SaaS) [4] are coming. However, budgets are cut down for software projects and the software vendors are required to deliver more features for less money. Who wants to compete and be a market leader, has to cut down his costs and to optimize the development processes. Customers wish to be involved in the software development process and to change the product being developed according to their needs. They want to have the application in the hands from the beginning. From the costs point of view, the testing is an expensive part of the whole software development. The sooner is a bug introduced in the fixing of the bug is cheaper and the resultant quality of the product is higher. One of possibilities how to reduce costs for testing is to get involved test automation, which is also in the scope of our research.

Automated testing of software applications deals with a number of issues and challenges. Firstly, we see a problem of insufficient expertise to automate tests as testers are not usually programmers. Secondly, technological issues and special test requirements might cause the test automation will get very difficult. Finally, limited resources (short deadlines, a few skilled people) require a fast and effective test development as well as a test execution. The problem of test automation is that test scripts are getting out-of-date 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.

Partial solutions of some issues such as automated test development and/or debugging, or test maintenance [18], [13] may increase the efficiency of the process of test automation. Moreover, the resultant quality of the final application may be higher as the automated testing may cover more features and combinations than manual testing. When continuous integration is employed, the developers can have almost immediate results of the new build and do not have to wait until the manual testers carry out the regression testing.

There are some techniques used to speed up the process of test automation and to increase the efficiency like Mockups [20] or generating test cases from application models [23]. Mockups may be very useful when Quality Assurance (QA) teams can prepare and develop tests against an application prototype in advance. Agile software development methodologies [17] bring additional requirements on a development of automated tests. The problem here is short sprints, which do not allow the QA team to develop automated tests if the story is being currently developed and the team does not have time to prepare the automation. Test recording approaches are very often employed in such cases since they do not require a time-consuming preparation of test scripts. 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 may decrease significantly.

In our research, we are aiming at a definition and a validation of a meta-model among requirements on automated tests, an application under test (AUT) and test scripts. The workflow of our approach can be described in

two steps: the user records test cases using the defined meta-model. In the next step, we will automatically create a structure of tests having common parts of the recorded tests broken down into reusable pieces.

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

2. Previous Work

Test automation is a subject of interest of many teams. We distinguish among different kinds of testing like unit testing, functional testing, GUI or regression testing, and performance testing. Unlike functional testing, unit test automation is being successfully theoretically covered. There are some approaches focusing on a generation of input test data [8] or the generation of test cases based on models of the application under test.

Xu [23] introduced an approach based on high-level Petri nets as finite state test models for the automated test generation and the test execution. In addition, he developed a model-based Integration and System Test Automation tool. This might be drawback in projects driven by the agile methodology. In agile, specifications and models do not frequently exist at the moment when the functional test automation is required and 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. Furthermore, the generated test cases might not be possible to execute without additional pre-processing. Since dynamic objects might have different IDs every time when the web page is generated, the test harnessing tool is not able to identify the objects then. In [2], [16], and [22] were introduced approaches for modeling of web applications.

Koopman et al. [15] introduced 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 HTML (no Flex, and no JavaApplets). Koopman does not discuss a general solution for rich clients and/or other environments, or cross-environments test (imagine mainframe panels).

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 et al. [21] describes a specification-based approach for the testing of web applications based on a TTCN-3 language [6] and [19]. Another specification-based approach based on XML is presented by Jia and Liu [14]. The specification-based approaches introduced in [6], [10], [19], and [21] are close to our intentions, but they did not employ the test recording techniques.

The approaches are either limited by a technology or a prerequisite to have the model of AUT, or another detailed specification. In comparison to the mentioned approaches, we want 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 et al. introduced recently several approaches for automated functional testing. He build to solution on top of the navigation of web applications as described in [10], [11], and [12]. His concept is to automate functional tests using UML diagrams [7], which are used for an input of the automated test case generation driven by the navigation in the AUT. Unlike our intentions, a model of AUT is required before the testing actually starts. Hence, he allows skipping the phase of formal design, since they enable to record tests. However, they did not focus on a reusability to reduce maintenance costs.

3. Proposed Solution

Approaches for the functional tests automation can be divided into three groups: (i) a plain test recording of test cases, (ii) test scripting using a programming language, and (iii) an automation frameworks.

The test recording facility records the user activity while the tester is conducting a test. The activity is saved in a form of test script and represented as a sequence of commands of a programming language such as Java, VBScript or C#. There are some techniques for a simplification of the maintenance like object repositories but they are less efficient than scripting-based or framework-based approaches due to code redundancy, no code optimization, a limited parameterization, and object duplicity.

Scripting-based approaches are typically for experienced QA engineers, who are more programmers rather than testers. The test development is time-consuming. On the other hand, QA engineers can utilize object oriented programming languages to develop highly reusable pieces, which can be subsequently used in complex tests. If the design of the test infrastructure well done, the maintenance costs are quite low.

Last but not least are automation frameworks. We can divide them according to the way how the test execution is driven into several groups. First group is driven by data meaning the test flow is controlled by input data. Second kind of frameworks is driven by keywords. The flow usually does not depend on input data, but is controlled by keywords. Keywords are commands from a vocabulary of available actions that can be executed by the test harnessing solution. The last group is driven by a model. The model-driven frameworks might be worth in cases

where the system changes rapidly and new test scripts can be easily generated from the model. Hybrid approaches are common (for example, Data-Keyword frameworks) as they utilize the best features from both approaches.

Based on the research, our primary aim is a definition of a meta-model of test cases, which is one of the keys to the efficient testing based recording. The meta-model will be used in a process of building a smart structure of tests from the recorded test cases. Another aim of our interest is a reusability of recorded tests. We are working on solution that will identify common test parts in the structure of tests and reorganize them.

The idea is to integrate solutions of single problems in one solution, i.e. in the test automation framework among test requirements, testing tools and AUTs. The framework is designed to be built from components, which can be extended later on. We will employ both the recording and scripting approaches, since the recording offer the fast test automation and the scripting good maintenance costs.

The automation framework enables the users to automate test cases (Fig. 1) in two ways. 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 way, the automation framework records a new sequence of 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.

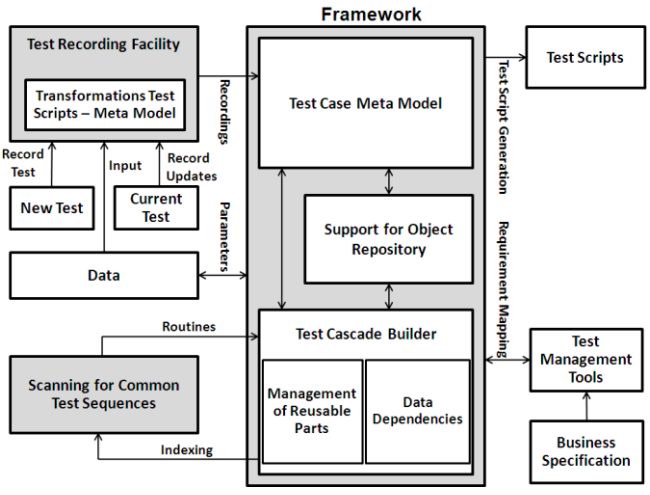

Fig. 1. The high level model of the framework.

The detected common test sequences can be excluded from the test suites subsequently and represented as new subunits of test cases as it is presented on Fig. 2.

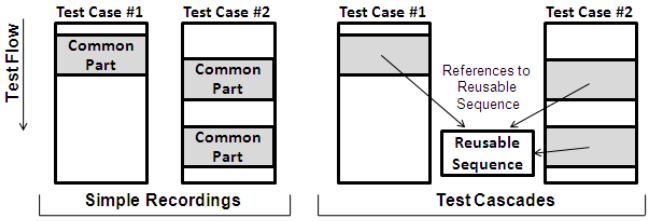

Fig. 2. Test Cascades. Gray parts identified by scanning algorithms represent the common parts in test cases.

For instance, if each test case contains a login to the application, the activity of logging in can be identified, separated and called from all test cases automatically. If such an activity needs to be changed due to an update in the application, the relevant common test sequence (reusable part) is updated and it takes effect in all tests using this common sequence.

An example of the concept of reusable parts is shown on Fig. 3. In order to create a structure of test cases with referenced reusable test parts for a better test maintainability, we need to find longest common subsequences of user activities (steps) in the whole 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. The problem might be to find an appropriate level of number of steps in the sequence.

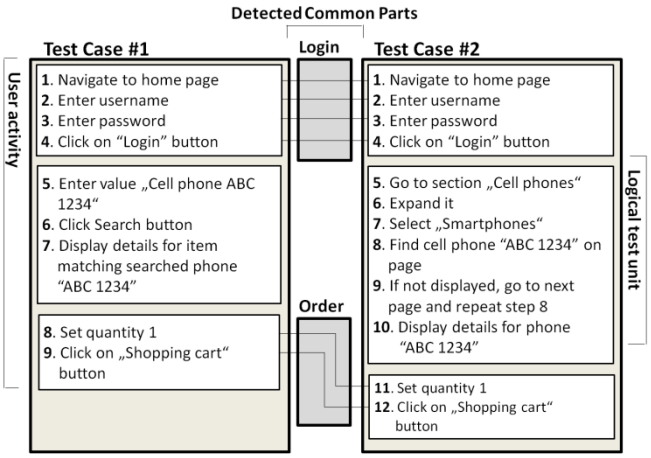

Fig. 3. The mapping of steps between two test cases.

Furthermore, the users have an option to design test cascades and script tests manually. Consider test cascades as a structure of test cases based on the reusability and resembling a program consisting of functions. Tests can be scripted 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.

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

Tab. 1. An example of the DSL used in the framework.

For non-experienced QA engineers, who do not have programming skills, is the proposed DSL easy to understand. Therefore, they can immediately start to alter the recorded test cases. Moreover, the resultant complexity of developing new test cases is lower in comparison to standard scripting approaches. Objects are stored in a central storage, which helps to increase the reusability. Built-in object repositories (for example the object repository of HP Unified Functional Testing tool) can be used as the central storage of such objects. An alternative is to use an internal object repository of the framework.

The concept of test cases represented by the meta-model with the test cascades provides a robust solution to changes in the application. Imagine what is going to happen if a new step is added into the current application workflow or if an object is changed in the AUT. The concept of test cascades supports the update in one place taking 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.

4. Conclusions

From our research of the current state-of-the-art, we have concluded that the problem of seeking reusable parts from the test recordings including the automated creation of test cascades is covered insufficiently. For example, in [1], [3], or [15] are presented either approaches for on-the-fly testing, which are actually limited by the technology or for testing based on models of AUT as it is presented in [6], [19], or [20]. A general concept that will employ the test recording technique is missing.

We have come out from the conclusions of the research and based on that, we have designed a framework for automation of functional tests. The approach does not suppose the model of the AUT on the input or another kind of detailed specification. The users can immediately start to automate the test cases. The recorded test scripts are subsequently processed in order to find common units and to create the test cascades. Both the approaches help to introduce the reusability into the concept of test recording and to reduce the test maintenance costs.

Currently, we are working on a detailed design of the meta-model and conducting a research of the longest common test step subsequences. Our goal is to create test cascades from the test recordings in a reasonable time if not on-the-fly while the user is recording the test. 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 will conduct experimental measurements on industrial projects and compare the test development costs as well as the test maintenance costs in comparison to manual testing and standard test automation approaches.

Acknowledgements

Research described in the paper was supervised by Assoc. prof. Ivan Jelinek and co-supervisor Miroslav Bures, both FEE CTU in Prague. This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS12/192/OHK3/3T/13.

References

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

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

[3] BESSON F. M., BEDER D. M., and CHAIM M. L., "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] BLOKDIJK G., "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] CAFARO M. and ALOISO G. (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] FOWLER M., "UML Distilled: A Brief Guide to the Standard Object Modeling Language". Addison-Wesley Professional, 3rd Edition, 2003. IBSN-13: 978-0321193681.

[8] FUJIWARA S., MUNUKATA K., MAEDA Y., KATAYAMA A. and UEHARA T., "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] FURHT B. and ESCALANTE A. (Eds.), "Handbook of Cloud Computing". 1st Edition., Springer, 2010. ISBN 978-1-4419-6524-0.

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

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

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

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

[14] JIA X. and LIU H., "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] KOOPMAN P., PLASMEIJER R., and ACHTEN P., "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] LANUBILE F. and MALLARDO T., "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] MARTIN R. C., "Agile Software Development, Principles, Patterns, and Practices". 1st Edition., Prentice Hall, 2002. ISBN: 978-0135974445.

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

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

[20] RIVERO J. M., ROSSI G., GRIGERA J., BURELLA J., LUNA E. R., and GORDILLO S., "From mockups to user interface models: an extensible model driven approach". In *Lecture Notes in Computer Science*, Volume 6385, Pages 13-24, 2010.

[21] STEPIEN B., PEYTON L., and XIONG P., "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] WU Y. and OFFUTT J., "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] XU D., "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.

About Authors...

Martin FILIPSKY was born in Prague. He studies PhD at the Czech Technical University in Prague and focuses on test automation and the efficiency of testing.