Industrial Requirements to Benefit from Test Automation Tools for GUI Testing

Christof J. Budnik, Rajesh Subramanyan and Marlon Vieira

Software Engineering
Siemens Corporate Research, Inc.
755 College Road East
Princeton, NJ-08540
christof.budnik.ext@siemens.com
rajesh.subramanyan@siemens.com
marlon.vieira@siemens.com

Abstract: In addition to the growing complexity of software systems, test effort takes increasing amounts of time and correspondingly more money. Testing costs may be reduced without compromising on software quality by minimizing test sets through optimal selection of test cases and introducing more powerful test tools. Attaining high levels of test automation is the objective.
There are problems which make the introduction of test automation in industry quite difficult. Solution providers and tool developers often do not understand the requirements in industry for test automation. Otherwise introducing test automation could become counterproductive. This paper points out essential demands on GUI test tools for industrial purpose.

1 Introduction

Testing has become the most popular verification and validation method in industry. To meet the market demands of producing high quality systems at low costs, testing should become more efficient and faster. Nowadays, there exist numerous test tools for a wide variety of programming languages that promise to support test by automation. However, the desired goals are often not achieved by test automation. Test automation can be surprisingly more intricate than manual testing. This paper points out the specific needs of test automation in the software industry for testing graphical user interfaces (GUI) currently not met, so that the automation could become more beneficial. Furthermore, it proposes valuable hints to test tool vendors, as to what is desirable for the industry and how they can improve their test tools. Challenges in designing next generation test tools are outlined by combining current research with the essential needs of the industry. The next section gives a short review of past work and points out the differences with this paper. In Section 3, the known problems in test automation are addressed and explain how research in industry and academia are dealing with them. Section 4 clarifies practical issues in adopting test automation and provides suggestions for developing

beneficial test tools for the industry. The paper concludes with a summary and proposes areas for future work.

2 Existing Methods and Related Work

The term test automation for GUIs in industry implies automated test execution in most cases. Test execution itself is automated by *Capture/Replay* tools and is used for regression testing. Test automation is more than just regression testing. Automation should be attempted for as many stages of the entire test process as possible.

First, the test process has to be managed, which means that the test documentation, the different releases of test cases, and the test cases itself have to be maintained and kept consistent with the requirements they are associated with. In practice, this is the most neglected step. In the next step, the test cases have to be generated. This can be done automatically [BB05/2]. This differs from a simple test script generator, which only allows defining test inputs and their corresponding outputs. When a specification represented as a model is used to automatically generate executable test, the method is referred to as *model-based* test generation. In contrast, in *data-driven* test generation, existing test cases of just plain templates are parameterized with different data the test has to be run with. Both approaches are considered by the industry as too formal, which therefore concentrates on generating the test cases manually. Once the test cases are designed, tests have to be conducted on the System under test (SUT). Automation of this test step is supported by a wide variety of tools that support many popular programming languages. Finally, the test analysis step is left in which conducted tests and their outputs are evaluated. The results of the analysis are needed to fix the faults and to decide further tests. This paper concentrate on the entire test process by giving hints to what should be automated and in which test phase.

Thus, the test automation has to keep through to the complete product life cycle. In [Ke99] the life cycle is divided into six major stages and for each stage there is a checklist of questions. Depending on the responses it is decided whether to automate the corresponding stage or not. However, it does not get into detail how this can be improved to set up test automation during the entire life cycle. Another work dealing with the issue about when a test should be automated is given in [Ma98]. It assumes the intent of automated testing and does not take a decision on need for automation. This work discloses why test automation fails in so many cases in the industry. To overcome these drawbacks, test tool developers are called to improve their products and the users have to scale down their expectations. Beyond this it is still difficult to decide which tool fits best to the requirements. This is addressed in [He03].

3 Test Automation in Theory and Practice

This paper describes test automation of the entire test process, divided into 4 parts (Refer Figure 1), in the context of black-box testing of software applications containing a

graphical user interface (GUI). The objective of test automation research is to maximize automation in test process. This means that existing models from development are used to generate test cases (model-driven) using suitable algorithms. The models have to contain the test inputs and the outputs to derive a fault model. This ensures that the tests are observable and the outputs can be compared with the expected ones described in the model. It maybe observed in Figure 1 that the test process steps are coordinated, i.e., the test results of one stage can be used as an input of the following stage. Therefore the entire test process is automated as a single step.

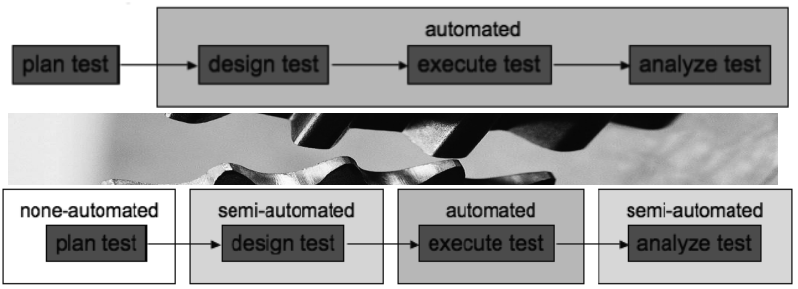

Figure 1: Gap between Test Automation in Theory and Practice

Insights from industrial projects trying to adopt test automation reveal that test automation is treated for each step separately. This differs from theory which treats test automation as a single step. In Figure 1, steps which still need some manual work are depicted as *semi-automated*. The test planning step is done manually and is accordingly depicted as *non-automated*. In addition automation is adopted in most cases too early which means without any preparation [Ba96, Ma98, Pe01]. The most reasonable cause to implement test automation in practice is for repeating test cases. That is also the reason why until now regression testing is so widely used and test automation is almost limited to the test execution in industry. Regression testing just confirms that despite changes to the software, the tests previously run provide identical results now. In fact, this means that no new functions have been tested. So regression testing is not supposed to find new faults in the code. In theory, it is the methodology that reveals fewest bugs. However, the most Capture/Replay tools are not able to provide expected benefits.

If the time gap between problems identified in the industry and solutions provided by academia increases, the industry could lose their interest in using test automation tools. There are a couple of additional neglected research fields that need more attention like usability, portability and visualization where the industry can gain from results of research problems. However, industry will not benefit from the research work until other practical problems as illustrated in the next section are still unsolved.

4 Test Tool Requirements in Industry

For testing software system powerful commercial Capture/Replay tools are available. Nevertheless, numerous time consuming and error-prone manual steps are still necessary

to complete the test process. Still, not all test tools allow constructing templates which can be run for different data from a database. Also, not all testing tools nowadays support the use of stubs or wrapping. From the industrial point of view there are many other problems concerning how to implement successful test automation. This section lists practical issues faced while attempting to introduce test automation for graphical user interfaces supported by Capture/Replay tools.

4.1 Dynamically Design Changes

Recorded objects of a GUI are identified by its unique properties. But these properties are static. This is an issue as only those objects which are available at the point of recording can be recognized by the tool. Otherwise, the objects have to be introduced into the tool afterwards. But, this can be very time consuming because the system has to set in the right state where the elements are active. Another problem arises due to the changes of different objects from release to release. Although, some test tools already provide to change the objects properties or to set regular expressions fitting the object, this has to be done manually and takes a lot of time to maintain. A significant problem is the case in which the object will change dynamically depending on the behaviour of the user. For example a conformed GUI depending on the user behaviour affected by knowledge based programs or learning software. There it is impossible to predefine all possible cases where objects will change. These vulnerabilities can be eliminated by a model-based test generation which maintains the model instead of repairing the test cases. Therefore approaches supposed by [BB05/1, ML+06] have to be added to commercial test tools.

4.2 Failure Treatment

Failure treatment is rarely supported by the test tools. One problem observed when testing the GUI is that while the GUI objects are checked, the underlying system operation is not. Indeed, one can write a test scenario which tests if all fields have to be filled out and if the submission button is disabled after sending once. This restriction is enforced to prevent multiple entries. However, this does not prove that the system has really connected with the database and stored or changed the values. Therefore, the test tools have to be extended by a functionality to compare database values with the expected values. Another kind of fault appears when the application crashes. Then a fault can be detected but the bug can not be fixed, because there is no information on which test cases failed. The only way to find that out is by conducting the test cases manually. This is analogous to finding a zero point numerically. Hence, the advantage of executing the test scripts without being present is lost. The same holds for faults which prevent the process from continuing like open windows. It is assumed that each test case starts at the main window, i.e., from the same starting point. Therefore the SUT has to be reset in its initial state from which each test case can be started. This is important if the test process should also provide test cases which are expected to fail [Be01]. Bret Pettichord mentions this special case of resetting the system in his work where he called it an "Error Recovering System" [Pe01]. For bug fixing, it is also recommend having a function which can set the system in predefined system state from which the test can

further run. Additionally, it may be necessary to have a memory map which is recorded during the test process. A worst case scenario occurs when a test case crashes not only the application, but also the testing tool or even the whole system. In this case the only chance to overcome the problem is to start the test execution from a remote machine which has to be provided by the test tool or even to support a test execution on a virtual machine.

5 Conclusion and Future Work

Nowadays, test research theory is well known in industry as well as to the developers of testing tools. The test criteria covered by each tool are limited to either the structure of the software or different data sets. The practical application of those test tools is missing. This paper points out the industrial requirements that need to be fulfilled in order to benefit from test automation. The issues and suggestions discussed go beyond recommendations including tools to test automation requirements in industry. This work should encourage the developers of test tools to go beyond providing more test methods or metrics, but also fill gaps by understand industrial requirements for test tools. Thus, first the missing of the outlined functionality has to provide to fulfill the industrial needs. The enhancement of interoperable test architectures and the testability of software system are planned for the future.

Bibliography

[Ba96] J. Bach. Test automation snake oil. Windows Technical Journal, pages 40–44, October 1996.

[Be01] F. Belli. Finite-state testing and analysis of graphical user interfaces. In ISSRE, pages 34–43, 2001.

[BB05/1] F. Belli, C.J. Budnik. Towards Minimization of Test Sets for Human-Computer Systems. IEA/AIE 2005, pages 300-309, 2005

[BB05/2] F. Belli, C.J. Budnik. Towards Self-Testing of Component-Based Software. COMPSAC (2) 2005, pages 205-210, 2005

[Du99] E. Dustin. Lessons in test automation. Software Testing & Quality Engineering Magazine, pages 16–21, September/Oktober 1999.

[FG99] M. Fewster and D. Graham. Software test automation: effective use of test execution tools. CM Press/Addison-Wesley Publishing Co., New York, NY, US, 1999.

[He03] E. Hendrickson. Making the right choice: The features you need in a gui test automation tool. STQE Magazine, pages 20–25, November/December 2003.

[KBP01] C. Kaner, J. Bach, and B. Pettichord. Lessons Learned in Software Testing. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[Ke98] D. Kelly. Software test automation and the product life cycle. MacTech, 13, 1999.

[Ma98] B. Marick. When should a test be automated, 1998.

[ML+06] M. Vieira, J. Leduc, B. Hasling, R. Subramanyan, J. Kazmeier. Automation of GUI testing using a model-driven approach. Proceedings of the 2006 international workshop on Automation of software test, ACM Press, pages 9-14, 2006

[Pe01] B. Pettichord. Success with test automation. Quality Week, May 2001.

[Po02] B. Posey. Just Enough Software Test Automation. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.