An Automated Testing Strategy Targeted for Efficient Use in the Consulting Domain

Annakkodi P. S[1], Mohanapriya. V[2]

[1]Research Supervisor, Head, Department of Information Technology, Sri Ramalinga Sowdambigai College of Science & Commerce, Coimbatore-109, Tamil Nadu, India

[2]Research Scholar, Department of Computer Science, Sri Ramalinga Sowdambigai College of Science & Commerce, Coimbatore-109, Tamil Nadu, India

Abstract: ***Test automation can decrease release cycle time for software systems compared to manual test execution. Manual test execution is also considered inefficient and error-prone. However, few companies have gotten far within the field of test automation. This thesis investigates how testing and test automation is conducted in a test consulting setting. It has been recognized that low test process maturity is common in customer projects and this has led to equally low system testability and stability. The study started with a literature survey which summarized the current state within the field of automated testing. This was followed by a consulting case study. In the case study it was investigated how the identified test process maturity problems affect the test consulting services. The consulting automated testing strategy (CATS) been developed to meet the current identified challenges in the domain. Customer guidelines which aim to increase the test process maturity in the customer organization have also been developed as a support to the strategy. Furthermore, the study has included both industrial and academic validation which has been conducted through interviews with consultant practitioners and researchers.***

Keywords: Consulting, Testing, Requirements, Process Improvement.

1. Introduction

Software testing is a practice that is neglected in many development projects due to budget and time constrains. In the test consulting domain, the testers and test managers change domains frequently due to large sets of customers involved. This chapter will present the motivation for this thesis project followed by the aims and objectives and research questions. The research methodology will be briefly introduced followed by an outline for the rest of the report.

1.1 Back Ground of the Study

Executing manual test cases several times is inefficient and error-prone and by automating these, the tests can be improved in later development phases, resources may be freed and the release cycle time may be decreased. Acting as a consultant in the test consulting domain infers some special issues that need to be handled in regards to the automation of the manual test cases in the customer development projects. The development process maturity often differ between the customers and with this in mind, the automated test procedures, methods and approaches used by the consulting firms must be adapted to suit the different customer domains and the distinct projects within these domains.

If automated testing is not considered in the architecture and design, it will be decrease the possibilities of automating the test cases in the later phases. This can pose problems for a test consultant that arrives in late phases of development where these items are hard to change for the sake of automating the test cases. As mentioned by Keller, the success of the automated tests are dependent on the test automation strategy that describes which test types that are to be performed, such as for example, integration tests, reliability tests and functional tests.

There are development methodologies that support automated testing, such as test driven development. Such practices can in fact reduce the defects in the software products and this is partly because it enables automated test cases to be written before the actual problem solution implementation. However, the consulting domain differs from traditional software development in the sense the consultants arrive in various phases of development depending on the contract with the given customer. It would hence be an advantage if the consultant could guide the early development phases in a direction which would facilitate automated testing in the later phases when the consultant arrives

Automated testing is not the best verification technique for every single scenario, many other factors needs to be considered before making the decision to automate the test case such as what artifact that are to be tested, how many times the test are to be run and how long time it will take to implement the test suite. However, having them gives the advantage of being able to run them more frequent and improves the quality of the test cases.

1.2 Aims and Objectives

This aim of this thesis project was to report on the difficulties within the test consulting domain in regards to

the automated test methods and processes used. With this information in mind, an automated testing strategy and customer guidelines has been constructed with the aim of making these methods and processes more adaptable between different customer domains. The objectives which were formed prior to the study are primarily described in the list below:

1. Identify which automated testing methods, approaches and strategies that are used in the consulting domain.
2. Identify how these automated testing methods, approaches and strategies differ from the corresponding ones used by standard development companies and the ones considered state-of-the-art.
3. Construct a theoretical hybrid strategy for automated testing, targeted for efficient adaptation in the consulting domain, with guidelines for easier adoption.
4. Validate the adaptation efficiency of the strategy in the consulting domain.
5. Validate the feasibility and cost effectiveness of the proposed strategy in the consulting domain

2. Automated Software Testing

In every large software development project, there exist several defects in artifacts such as requirements, architecture, design in addition to the source code, each of which decrease the quality of the product. Software testing practices are used to ensure quality of software items by finding these defects. The overall development cost can be decreased by finding these defects early in the development process rather than later. For example, consider performing a bug fix to a set of requirements after the implementation has been completed. When performing such change, the already implemented source code may now be based on an incorrect set of requirements. This means that the existing functionality may not be needed after all, rendering the development effort useless

Software testing can roughly be divided into several methods and levels each of which has distinct responsibility of testing. The methods include black-box and white-box testing which is discussed below. Software levels include unit, integration, system and validation testing each of which will be introduced in Section 2.2

Unit Testing: This level verifies if the implementation of the individual modules described by the detailed design behaves in an acceptable manner. However, it could also be used to ensure the correct behavior of the units by using a black-box approach.

Integration Testing: The integration testing level focus on the high level design which usually contains cooperating architectural artifacts. This means that this level verifies if the implemented interactions between modules are correct.

System Testing: The system testing level ensures that the complete system is behaving in acceptable manner. It acts with the system specification as the basis and the input source to this test level comes from the developers.

Acceptance Testing: This testing is usually done by the end-user or customer and verifies if the requirements are fulfilled by the implementation with the requirements specification as a basis. The main difference between this level and the system testing level is that the source of input comes from the customer instead of the developers.

3. Automated Testing Opportunities

Manual execution of test cases is considered inefficient and error-prone and it is often possible to increase the efficiency by automating these which also relives the workload of the testers. By introducing automated test cases to the development process, the testing cost also decrease and some of the tedious manual labor is avoided. However, in addition to the opportunities it provides, there are several challenges as well. It does take some time to develop these automated test cases and several considerations should be taken before their implementation. If test cases are to run several times which is the case in for example regression testing, it may prove beneficial to automate them so that the resources needed for the re-run can be put to better use.

Even with the introduction of automation it is most often impossible to achieve full test coverage due to the large amount of different states and branches that a software product may enter. This introduces the issue that handles which artifacts that are important enough to be considered for coverage of the automated test cases. However, it should be noted that striving for full coverage is not always the most appropriate measure for fault detection. This is due to the fact that the defects often have different severity while the test cases differ in terms of cost.

3.1 Reuse

In most development stages, there has been a focus of component reuse which has several advantages. First of all, the component can be written once and used many times which saves development effort. It also has quality benefits because the component may be refined and improved over time. This practice can be used for requirements, design artifacts and source code components and it can also be applied to the automated test cases. With this kind of reuse, the benefits discussed such as quality refinement is transferred to the test cases as well and first-class test cases is very important in testing. For example, with poor quality, false positives may be found instead of real defects which can lead to unnecessary manual labor. This is an issue that can be remedied with sound reuse.

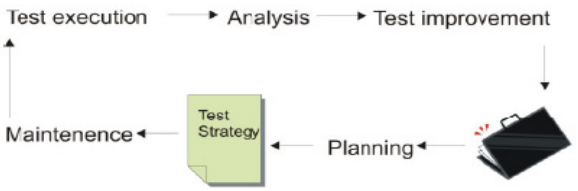


Figure 1: Reuse Strategy Example

To get a reusable quality test suite it could be appropriate to extend the normal test case development process briefly described by Keller et al. Figure 2 gives an example of how the test suite can be improved along sides the ordinary development. It contains the following stages.

Planning: This phase includes consulting the test strategy to see if the test case chosen from the test suite corresponds to the current testing goals.

Maintenance: Often, when test cases are brought from the test suite, they need some maintenance so that it can be adapted to the current setting. This state takes care of the possible modifications needed.

Test Execution: In this stage, the test is executed in order to find eventual defects and more importantly for the reuse issue, return test data to the next stage.

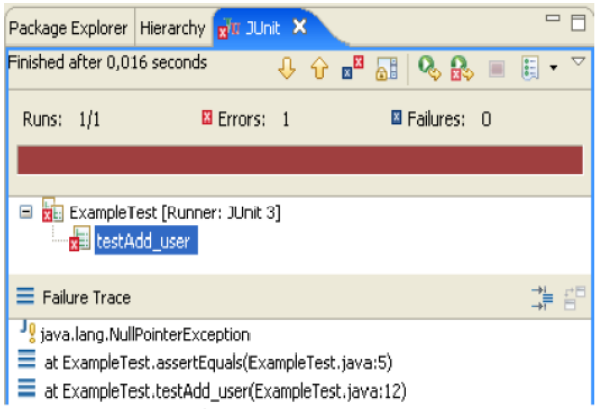
Analysis: Analysis in regards to test reuse is concerned with how the test case performed, if it fulfilled its purpose. Some measurements may be needed, depending on the current goals of the test strategy

4. Relevant Methods

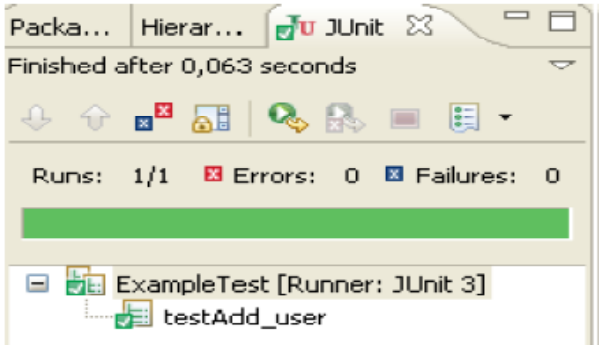
There are numerous frameworks available that covers different criterions. Several testing frameworks are available which supports the automation of test cases, not only to automate the test cases themselves but also to adapt other frameworks to fit several application domains. This section will introduce frameworks, methods and strategies that are considered to be useful primarily in the test consulting domain where the testing criteria often change

4.1 JUnit

JUnit is an executable testing framework that enables developers to write automated test cases for classes, methods and packages they have written using the Java programming language. In organizations where Unit testing is adapted, the JUnit and other xUnit frameworks are the ones that are primarily used. This can be traced to their early arrival to the development community but also to the simple structure of the frameworks. The benefits imposed by unit testing frameworks have been recognized by several IDE vendors. Net beans and Eclipse for example, has built-in support for JUnit which makes test case creation for particular classes, packages and individual methods a couple of clicks away.



As mentioned, the test aims for validation of the expected output for a unit and a failing test case is showed in Example 5 where the test described in Example 4 has failed. As can be seen in the JUnit GUI illustration in the example, a failure trace is given so that the test case can be examined and thereby the failing source code unit. In this case, the code returned null instead of the expected User instance



5. Consulting Automated Testing Strategy (CATS)

The automated testing strategy is developed for use by consultants that primarily deal with test automation in software development projects but some parts of the strategy may be useful for manual testing as well.

5.1 Strategy concepts

Strategy pointers in the following sections are distinct tips that can be applied in different phases of the testing project with the intent to increase the efficiency of the testing practices. The pointers can be applied independently of each other, depending on the parameters of the current development project and organization. As for the different phases of the strategy, these are not to be confused with the phases of the used development methodology since the

strategy phases are independent of the development methodology. The main concepts of these phases are to increase software testability and stability, increase the effectiveness of the test execution and to improve the strategy and customer guidelines with the execution results as the input source. The pointers are structured so that the test consultant can assess the pointers independently and choose which pointers that applies in the current development phase.

6. Customer Guidelines

As a complement to the automated testing strategy developed for use by test consultants, the following guidelines is intended for the customers and provide directions towards more testable and stable software applications in the customer projects. The guidelines are divided into so called pointers and each of these gives a specific tip of what should be done to facilitate the system and acceptance test. These pointers are intended to be implemented by the developers in the customer projects and be motivated by the consultant test manager by using the motivation sections for each pointer. As previously described, without system testability and stability in the release, the lead time for the consultant testers will increase which in turn decrease the efficiency of the system and acceptance test. The guidelines has-been designed based on consultant experience and empirically evaluated studies which prove their usefulness in development projects. Adoption of the pointers will increase the system testability and stability in the development projects and this will maximize the return of investment of the consulting services when the contract has been signed for the system and acceptance test.

7. Discussion

This section will provide a discussion of the perceived applicability of the proposed strategy and guidelines in a live consulting setting. Since only static validation has been collected, the discussion will be based on the opinions of the consultants at Testway, the opinions of one of their customers and the opinions of the thesis author.

8. Conclusion

The consulting automated testing strategy (CATS) along with its supporting customer guidelines was developed for consulting domains where the practitioners act in changing application environments. CATS are divided into three steps where the first step targets the system testability and stability, a step which should be done prior to the actual test automation. The second step handles issues that should be taken care of in the test execution phase. As for the final step of CATS, it is focused on strategy and customer guideline improvements. Both CATS and the guidelines was developed in cooperation with a test consulting firm where it was recognized that the most common challenges is related to requirements engineering practices and early verification activities in the customer projects.

9. Future Work

Only static validation has been performed through interviews within the consulting firm, relevant customer of this firm and researchers in academia. It would be appropriate for future researchers to assess the strategy and guidelines through dynamic validation by letting consulting customers use the guidelines before the consultant starts the assignment using the automated strategy. This way, an eventual increase in quality could be monitored and documented which would prove the worth of this study as well.

Reference

[1] Hayes, L. (1995). The Automated Testing Handbook. The Software Testing Institute, Texas
[2] Aurum, A., Peterson, H., &Wohlin, C. (September 2002). State-of-the-Art: Software Inspections after 25 Years. Software Testing Verification and Reliability
[3] Bach, J. (2001). James Bach on Explaining Testing to Them. Software Testing &Quality Engineering
[4] Borland. (2007). Automated Software Regression Testing & Functional Software Testing - from Borland. Retrieved May 22, 2007
[5] Edwards, S. H. (2001). A Framework for Practical, Automated Black-Box Testing of Component-Based Software. Software Testing, Verification and Reliability.

Author Profile

Mohanapriya. V received her BCA and MCA from Vivekanandha College of Engineering Affiliated to Anna University, Chennai, Tamil Nadu, India in 2007 and 2009 respectively. Currently pursuing her M.Phil in Computer Science at Sri Ramalinga Sowdambigai College of Science & Commerce, Coimbatore affiliated to Bharathiar University, Coimbatore, Tamil Nadu, and India.

Annakkodi P. S is working as Assistant Professor in the Department of Information Technology, Sri Ramalinga Sowdambigai College of Science & Commerce, Coimbatore. She has 12 years of teaching experience in the specializations Computer Science and Information Technology and guided several PG and M.Phil Projects.