International Journal of Information Technology and Management

Vol. VI, Issue No. II, May-2014, ISSN 2249-4510

TESTING, MINIMIZATION, TRANSPARENCY AND PERFORMANCE OF SYSTEM TEST CASE PRIORITIZATION: A CASE STUDY OF REGRESSION TESTING

AN INTERNATIONALLY INDEXED PEER REVIEWED & REFEREED JOURNAL

www.ignited.in

International Journal of Information Technology and Management
Vol. VI, Issue No. II, May-2014, ISSN 2249-4510

Testing, Minimization, Transparency and Performance of System Test Case Prioritization: A Case Study of Regression Testing

Meena Mehta

Assistant Professor (Adhoc) Maharaja Agarsen College, University of Delhi

Abstract – In a software testing domain, different techniques and approaches are used to support the process of regression testing in an effective way. The main approaches are test suite minimization, test case prioritization and test case selection. Test case prioritization techniques improve the performance of regression testing, and arrange test cases in order to obtain maximum available fault that is going to be detected in a shorter time.

User-sessions and cookies are unique features of web applications that are useful in regression testing because they have precious information about the application state before and after any change that is made to the software code. The main challenge is the effectiveness of average percentage fault detection rate and time constraint in the existing techniques. Thus, in this research the priority is given to clustering test cases that are performed based on some criteria related to http requests which collected from the database of server side. To verify the new technique some fault will be seeded in subject application then applying the prioritization criteria on test cases for comparing the effectiveness of average percentage fault detection rate and time with existing techniques.

Prioritization of test cases is generally done to reduce the cost of regression testing. We prioritize our test cases so that those which are more important, by some measure, are made to run earlier in the testing phase. There exists a large variety of prioritization techniques in the literature, we have basically used coverage-based prioritization techniques (i.e., prioritization in terms of the number of statements, path coverage, branch coverage and fault coverage) controlling the field.

A prioritized test suite which covers more than one coverage criteria is considered to be a stronger than those which cover only single coverage. The proposed method was empirically studied for bank application and the results show that the proposed work is more effective than the existing method.

----------♦----------

INTRODUCTION

In today's changing business environment, time to market is a key factor to achieving project success. For a project to be most successful, quality must be maximized while minimizing cost and keeping delivery time short.

Quality can be measured by the customer satisfaction with the resulting system based on the requirements that are incorporated successfully in the system. Boehm proposes a value-based approach to software engineering that measures the value the system provides to the prospective customers. Boehm suggests that currently most of the software engineering research and practice is done in a valueneutral setting whereby all requirements, use cases, test cases, and defects are treated as equally important without considering the business value provided to the customer.

Value-based engineering involves the prioritization of development activities, keeping in mind stakeholder value propositions. Value-based software engineering practices are believed to improve user-perceived software quality. In this paper, we explore a value-driven approach to prioritizing software system test with the objective of improving userperceived software quality. Software testing is a strenuous and expensive process. Research has shown that at least 50% of the total software cost is comprised of testing activities.

Companies are often faced with lack of time and resources, which limits their ability to effectively complete testing efforts. Often, the engineering team is compelled to stop their testing efforts abruptly due to schedule pressures and is forced to deliver the system with compromised software quality. To optimize the time and cost spent on testing, prioritization of test cases in a test suite can be beneficial. Test case prioritization (TCP) involves the

explicit planning of the execution of test cases in a specific order with the intention of increasing the effectiveness of software testing activities by improving the rate of fault detection earlier in the software process.

TCP has been primarily applied to improve regression testing efforts. Regression testing is the process of retesting of a system or component to verify that changes made to the system code have not caused unintended effects and that the system is still compliant with the specified requirements. Software engineers save test cases and re run these test cases as regression tests in later versions. Running the entire set of test cases on a new and/or revised version could be expensive. Currently, regression TCP techniques use structural coverage criteria to select the test cases. Structural coverage techniques, such as statement or branch coverage, are applicable at the code level.

CASE DESCRIPTION

At Sony Ericsson the software verification is carried out at different levels by different departments, i.e. unit, integration, system and acceptance test. The software is divided into different functional areas and for each area there is a group of developers and a group of testers. The integration test is carried out at a test department by a test group for each function. The testing at this level is performed on temporary builds of the software or on the main software branch, usually in a hardware prototype. The system test is carried out by another test department and is performed on the main branch. Finally a release candidate of the software is sent for acceptance test to the carrier service providers.

The software development process is an incremental process, where each component is developed and integrated in small iterations. However, to ensure that the overall quality of the software is maintained, after several iterations, a range of regression test sessions are performed by each function test group on an integrated system. For every new feature, several new test cases are created based on the feature requirements. All the test cases are written for black box testing and are not connected to a specific part of the code.

The test cases are added to the test database which contains all the test cases relevant to a specific product. The amount of features increases in each project and therefore the total amount of test cases available is too large to re-test all during RT on a regular basis. All test case descriptions and the execution data are stored in a commercial tool, HP's Quality Center (QC). QC is a web based test database that supports essential aspects of test management. The defect reports are stored in a defect management system (DMS) and linked via an id number, to the revealing test case in QC and also to other test cases that are affected by the defect. Sony Ericsson uses QC as its main test management tool. Test planning, test design and test execution is performed in the QC environment. Each executed test case should contain the following information: Software version, hardware version, test status (Blocked, Failed, N/A, No Run, Not Completed, Passed), execution date and time, tester id and if the status is Failed a DMS number from the defect report in the DMS.

BACKGROUND

This section introduces the basic concepts and definitions that form a nomenclature of regression testing and minimisation, selection and prioritisation techniques.

Regression Testing - Regression testing is performed between two different versions of software in order to provide confidence that the newly introduced features of the System Under Test (SUT) do not interfere with the existing features. While the exact details of the modifications made to SUT will often be available, they may not be easily available in some cases. For example, when the new version is written in a different programming language or when the source code is unavailable, modi_cation data will be unavailable.

The following notations are used to describe concepts in the context of regression testing. Let P be the current version of the program under test, and P0 be the next version of P. Let S be the current set of specifications for P, and S0 be the set of specifications for P0. T is the existing test suite. Individual test cases will be denoted by lower case: t. P(t) stands for the execution of P using t as input.

Distinction between Classes of Techniques - It is necessary at this point to establish a clear terminological distinction between the different classes of techniques described in the paper. Test suite minimisation techniques seek to reduce the size of a test suite by eliminating redundant test cases from the test suite. Minimisation is sometimes also called `test suite reduction', meaning that the elimination is permanent. However, these two concepts are essentially interchangeable because all reduction techniques can be used to produce a temporary subset of the test suite, whereas any minimisation techniques can be used to permanently eliminate test cases.

RESEARCH PROCESS

In this paper our research procedure is based on the five main phases as shows in figure 1. In the first phase we have introduced literature review. The second phase is to find the problem by systematic mapping on web application testing and determine the problem statement in this domain. The third phase is conducted to propose solution then we check the rate of fault detection in phase fourth. Finally, the last phase is evaluated with three case

studies that compare the average fault detection rate with the results of previous techniques.

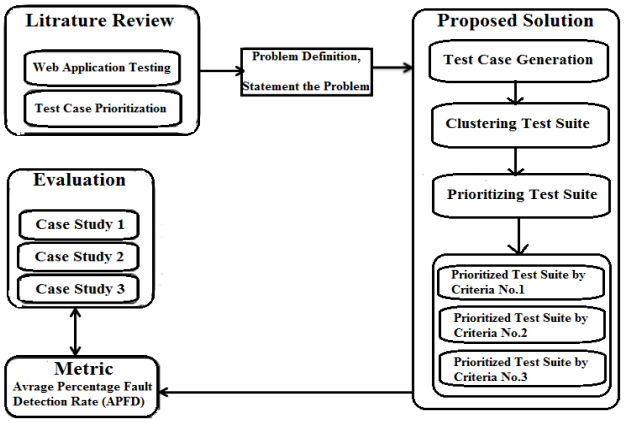

Figure 1. Research process

Literature Review - The focus of research is on two approaches in web application regression testing which are clustering test suite and prioritization test suite in case of effectiveness time and fault detection rate. This step is aimed to search about the current state of the art and challenges in web application test considering prioritization test case technique for better fault detection in less time. The drawbacks of these approaches are explored thoroughly in literature review step of the research process.

Problem Definition - The analysis of the problem, in test cases prioritization and web application testing is performed making the current literature review as a base for the issue or problem.

Proposed Solution - The goal of this research is to propose a new technique in test case prioritization for improving effectiveness of fault detection rate and time, due to these objectives the first step of the proposed solution is generating test cases base on the user session data which is using logs file in server side.

TEST CASES DEVELOPMENT AND PRIORITIZATION

Test case is a set of conditions using which a tester determines whether an application or software system is working correctly or not. The basic objective of writing test cases is to validate the testing coverage of the application. Test cases should be written after understanding the function or technology that is to be tested, and each test case should be submitted for peer review. Test case is designed based on random input data.

Descriptive test cases should be prepared rather than detailed test cases because determining pass or fail criteria is usually easier with this type of case. Moreover, detailed test cases are more time-consuming to develop and maintain. When planning the test cases, we should keep in mind that it is not feasible to test everything. Instead of trying to test every combination, we prioritize our testing so that we can perform the most important tests — those that focus on areas that present the greatest risk or have the greatest probability of occurring.

Test case prioritization technique improves regression testing by ordering test cases such that the more important test case runs earlier in the testing process. This is inefficient to re execute all the test cases in regression testing following the software modifications. Using information obtained from execution of previous test cases, prioritization technique is used to order the test cases for regression testing so that most beneficial are executed first thus allows an improved effectiveness of testing using minimum time and resources.

Effective test case prioritization technique for regression testing is necessary to ensure optimum utility and no side effect in the software after modification. Test case prioritization is rearranging the order of the test case based on certain constraints so that the most beneficial test cases may be executed first. Most of the existing research works on test case prioritization methods are based on single coverage criterion which is time consuming and more expensive. A prioritized test suite which covers more than one coverage criteria is considered to be a stronger coverage goal than a test suite which covers single coverage criteria.

Test case prioritisation seeks to find the ideal ordering of test cases for testing, so that the tester obtains maximum benefit, even if the testing is prematurely halted at some arbitrary point. The approach was first mentioned by Wong et al.. However, in that work it was only applied to test cases that were already selected by a test case selection technique. Harrold and Rothermel proposed and evaluated the approach in a more general context.

For example, consider the test suite described in Table 3. Note that the example depicts an ideal situation in which fault detection information is known. The goal of prioritisation is to maximise early fault detection. It is obvious that the ordering A-B-C-D-E is inferior to B-A-C-D-E. In fact, any ordering that starts with the execution of C-E is superior to those that do not, because the subsequence C-E detects faults as early as possible; should testing be stopped prematurely, this ensures that the maximum possible fault coverage will have been achieved.

PRIORITIZATIONS OF REQUIREMENTS FOR TESTING

Building on this work, we propose a multi-faceted, system-level prioritization technique called PORT. In this section, we explain the current set of PORT prioritization factors, the PORT prioritization factor collection process, and the algorithm underlying PORT. The algorithm underlying PORT Version 1.1 prioritizes based upon four factors:

(1) customer-assigned priority on requirements,

(2) Requirement volatility,

(3) Developer perceived implementation complexity, and

(4) Fault proneness of requirements. We discuss below these four factors, the reasoning of why they were chosen in our prioritization technique, and their importance to software testing.

Customer-assigned priority (CP**) :** is a measure of the importance of a requirement to the customer. The customer assigns a value for each requirement ranging from 1 to 10 where 10 is the requirement with the highest customer priority.

***Reasoning*:** Approximately 45% of the software functions are never used, 19% are rarely used, and only 36% of the software functions are sometimes or always used.

A fault that lies along the path of normal execution results in frequent failures, and the majority of the effort should be spent in finding these faults. A focus on customer requirements for development has been shown to improve customer-perceived value and satisfaction. By identifying and thoroughly testing the fraction of requirements that would be of highest importance to the customer, we aim to increase the business value generated to the customer. Additionally, if the testing efforts were stopped abruptly due to schedule pressures, the requirements of highest value to the customer would have been tested early and thoroughly.

CONCLUSION

In this paper we report on a case study of the implementation of history-based regression testing for the purpose of improving transparency and test efficiency at function test level in a large software development organization. Current practices are investigated and a semi-automated tool, combining the concepts of history-based regression testing in literature with good practices in the current process as well as practitioners' opinions, are implemented. Three different strategies, the current experienced based approach and two systematic approaches are empirically compared through a post hoc quasi experiment. The outcome of the tool is further assessed through manual reviews made by the practitioners.

The web application domain has an advantage that actual user-sessions can be recorded and used for regression testing. While these tests are indicative of the user's interactions with the system, both clustering and prioritizing user-sessions has not been thoroughly studied. In this paper we examined a new technique for using cluster based test case prioritization of such user-sessions for web applications. By applying several new prioritization criteria to these test suites to identify whether they can be used to increase the rate of fault detection. Prioritization by frequency metrics and systematic coverage of parameter-value interactions may increase the rate of fault detection for web applications.

The analysis of trends reported in the paper reveals some interesting properties. There is evidence to suggest that the topic of test case prioritisation is of increasing importance, judging by the shift in emphasis towards it that is evident in the literature. It is also clear that the research community is moving towards assessment of the complex trade and balances between different concerns, with an increase in work that considers the best way in which to incorporate multiple concerns (cost and value for instance) and to fully evaluate regression testing improvement techniques.

This focus on empirical methodology is one tentative sign that the field is beginning to mature. The trend analysis also indicates a rising profile of publication, providing evidence to support the claim that the field continues to attract growing attention from the wider research community, which is a positive finding for those working on regression test case selection and minimisation and, in particular those working on prioritisation problems.

REFERENCES

- A.Sajeev and B. Wibowo, "Regression test selection based on version changes of components," in Tenth Asia-Pacific Software Engineering Conference, 2003, pp. 78–85.
- B.Boehm and L. Huang, "Value-Based Software Engineering: A Case Study," *IEEE Computer*, vol. 36, no. 3, pp. 33-41, March 2003.
- B.Boehm, "Value-Based Software Engineering," *ACM Software Engineering Notes*, vol. 28, no. 2, pp. 1-12, March 2003.
- B.Qu, C. Nie, B. Xu, and X. Zhang, "Test case prioritization for black box testing," in Annual International Computer Software and Applications Conference, vol. 1. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 465–474.
- J.H.Andrews, L. C. Briand, and Y. Labiche. Is mutation an appropriate tool for testing

experiments? In Proceedings of the 27th International Conference on Software Engineering (ICSE 2005), pages 402{411. ACM Press, May 2005.

- L.C.Briand, Y. Labiche, and S. He. Automating regression test selection based on UML designs. Journal of Information and Software Technology, 51(1):16{30, 2009.

- Md.Imrul Kayes, “Test Case Prioritization for Regression Testing based on Fault Dependency”, IEEE, pp- 48-52.

- N.Alshahwan and M.Harman. Automated session data repair for web application regression testing. In Proceedings of 2008 International Conference on Software Testing, Veri_cation, and Validation, pages 298{307, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

- P.Runeson and M. H¨ost, “Guidelines for conducting and reporting case study research in software engineering,” Empirical Software Engineering, vol. 14, no. 2, pp. 131–164, 2009.

- R.C.Bryce and C. J. Colbourn. Test prioritization for pairwise interaction coverage. In Proceedings of the ACM workshop on Advances in Model-Based Testing (A-MOST 2005), pages 1{7. ACM Press, 2005.

- Y.Fazlalizadeh, A. Khalilian, M. Azgomi, and S. Parsa, “Prioritizing test cases for resource constraint environments using historical test case performance data,” 2009 2nd IEEE International Conference on Computer Science and Information Technology, pp. 190–195, 2009.

- Zheng Li, Mark Harman, and Robert M. Hierons, 2007. “*Search Algorithms for Regression Test Case Prioritization*”, IEEE Transactions On Software Engineering, Vol. 33, No. 4, pp 225 – 237.