Advanced Performance analysis of Prioritization Techniques Through Regression

[1]Ayaz Ahmed Shariff,[2] Dr.R.P.Singh
[1]Department of Computer Science
SSSUTMS, Sehore

Abstract— Due to dynamic changing business and operating environment, software evolution is inevitable. As software evolves, there is a need for a retest of the system in order to ensure its validity and thus regression testing assures that this happens. Regression Testing has several research categories, one of the most popular areas is Test Case Prioritization which orders test cases based on a particular criteria to be tested in accordance to available resources, and thus ensuring that the most critical test cases are tested first. In this study we examine the different types of test case prioritization and a description of the techniques presented alongside the issue raised in each study. This will give an insight into the main problems currently plaguing test case prioritization and thus pave way for further research areas to tap into.

IndexTerms — Regression Testing; Test Case; Prioritization Techniques; Test Case Selection; Test Suit Augmentation; Test Suit Reduction;

I. INTRODUCTION

As software systems change either through a change in environment, platform or simply evolving into a higher version, they must be re-tested to ensure quality, however since the system has changed, the same test cases cannot be applied to the same system, some test cases are obsolete while others are more important to be tested first. Regression Testing concentrates on the aspect of change in software and performs testing on the modified version of the software [1]. The main goal of Regression Testing is to make sure that these modification and changes that happen to the software do not have a negative impact on applications [2]. As much as 80% of the testing budget has been estimated to be used by Regression Testing [3]. It can also take as much as 50% of the software maintenance costs [4].

One of the most expensive and repetitive processes in software testing or software process overall is regression testing. To ensure quality, systems have to be constantly retested as they evolve, each modification or change requires the software to be rested as a result [5]. The cost is the major concern of regression testing. In some cases regression testing may consumes 50% of testing [5]. The second major concern is the efficiency of regression testing method [6]. The efficiency of test suites is assessed in terms of execution time [7]of test cases and their ability to validate the software under test [8]. The most desirable of test suits are those that retain a high fault detection while remaining small at the same time [9]. The main goal of regression testing is to increase fault detection while at the same time decrease their cost of execution or production of the aforementioned test suit [2].

These objectives are achieved by test case selection methods [10], test case prioritization [11], test suite reduction [12] and test suite augmentation [13]. When test cases are selected, they are either prioritized (ordered) [4], reduced or augmented which increases the likelihood of faults being discovered early in the testing process [5]. Detecting faults early is needed because we bug fixes should start as soon as possible [16]. If by any chance the resources exhaust or deplete during the testing process, the consequences would be much less severe as the critical tests have already either been retested or remade. Much of regression testing research has been focused on test case selection and prioritization methods to reduce the execution time and improve the bug detection capability [2, 17, 18]. In order to achieve these objectives the original test suit designed for the first iteration of the software is considered imperative in achieving these objectives [9]. The quality of these test suites is very important as it defines the quality of the selected test cases, and thus it also impact the effective prioritization or ordering of test cases for successive test suits made for modified version of the software [2].

This study is comprised of several parts. It the second section, the background of the study is elaborated. It starts by expanding on the idea of Regression Testing and its purpose within software engineering. It also expands on the components of Regression Testing. The third section does a critical analysis of various regression testing techniques and elaborates on the gaps and the significance of Test Case Prioritization. The fourth section expands on Test Case Prioritization most popular techniques. This is followed up by a critical analysis of these techniques. In the last section a conclusion is made based on the information elaborated.

Background

As software systems change either through a change in environment, platform or simply evolving into a higher version, they must be re-tested to ensure quality, however since the system has changed, the same test cases cannot be applied to the same system, some test cases are obsolete while others are more important to be tested first. Regression Testing is a solution created to tackle this particular issue.

To define regression testing as a whole, consider a program P and its modified version P', with test suite T generated for program P (the original version). Engineers use regression testing to test the P'. As elaborated earlier, regression testing is an expensive

process and the need to adopt cost effective Regression Testing is much required. The one most basic and straightforward approach is to retest-all, in which the tester executes all the test cases again on the next version of a program P'. But retest-all is both expensive and impractical due to the fact that new version of software under test is not changed 100%. There is only few parts of that software may change, therefore it is not required to test the whole software again. Thus the wise approach is to test only the parts which are changed and are in need of a retest. However, as the software evolves, the test suite also grows and it is difficult to rerun the test suite again on the changed software. These limitations forces to consider selection, prioritization or reduction of the test cases in use. Regression Testing has several subtypes and subcategories. The four main types of regression testing are Test Case Selection, Test Case Prioritization, Test Suit Reduction (also known as Test Suit Minimization) and Test Suit Augmentation. Figure 1 visualizes the subtypes and also correlates the several methods that are shared among all the different regression testing types. Such as Code Based Methods, Test Tube, Graph Based Methods, Model Based Methods, Program Slicing Techniques, Regression Testing for web based applications, Regression Testing for GUI, and Regression Testing for embedded software. All of which is demonstrated in the diagram on Figure 1.

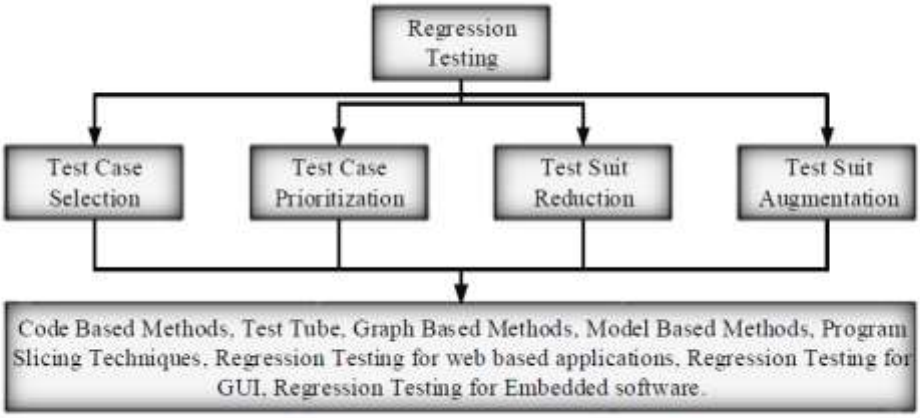

Figure 1. Sub-types of Regression Testing

II. Test Case Selection

One of the main types of regression testing, which is also considered its biggest field is Test Case Selection. The concept of Test Case Selection comes in tandem with the definition of Regression Testing. As elaborated earlier it is both expensive and impractical to simply retest all the test cases after software or program has updated. Thus a method is required in order to identify and select these test cases that are relevant based on a particular criterion [21].

To defined Test Case Selection clearly, take P as the first build and iteration of a program, with T as the test suit designed for the program P. P' is the modified version of program P. In Regression Test Selection (RTS) the goal is to create T' which is a subset of T to be used for P'. An effective RTS technique would selects all test cases that can reveal faults or bugs in the new modified version of the program which is P'. As it is not possible to find all fault-revealing test cases by simply selecting them, among the most effective methods is to simply select test cases that are affected by the modification – which under the right circumstances it includes all fault revealing test cases [2].

Table 1. Research Growth of TCP Methods

No.	Research Title	Issues Raised	Proposed Solution	Year
1	A Study of Effective Regression Testing in Practice [34].	Selection techniques at the time ignored the coverage of tests in modified programs.	Hybrid modification, minimization and prioritization selection technique.	1997
2	Test Case Prioritization: An Empirical Study [35].	The tradeoffs of various prioritization techniques based on rate of fault detection.	Optimize prioritization techniques by improving rate of fault detection.	1999
3	Prioritizing Test Cases for Regression Testing [33].	Fourteen techniques were examined. Tradeoffs between statement and function level techniques were examined.	Version-specific test case prioritization has shown to improve the rate of fault detection.	2000
4	Effectively Prioritizing Test Development Environment [5].	Prioritizing at a statement level in large scale software development environment is very slow and expensive.	Introduced prioritization based on binary code approach and built a test prioritization system (Echelon) that prioritized based on the changes made to the program.	2002
5	Empirical Studies of Test	Does Test Case Prioritization	In C based prioritization	2004

III. Test Case Prioritization Sub-categories

There are many TCP methods reported in literature [8, 3]. There is difference of opinion in classification TCP techniques reported in literature. A detailed survey [8], investigated 47 TCP studies published between 1999 and 2009. This study classifies the existing research into the following groups.

Coverage Based TCP: These TCP methods use coverage information to reorder the test cases. The assumption is that maximum coverage means maximum number of faults may be identified during testing.

Distribution Based TCP: These methods based on the distribution of profiles of test cases. Similar test case can identify the same faults and considered redundant test cases.

Human-Based TCP: This TCP class used human comparisons of test case results and usage scenarios to prioritize the test cases.

Probability Based TCP: These methods use probability theory to prioritize the test cases. The test case history with fault detection probability is calculated for each test case. Then this probability score is used to prioritize the test cases. Bayesian network based TCP methods are classified in this category.

History Based TCP: The test cases execution history can be used in this method to prioritize the test cases.

Requirement Based TCP: Requirement properties can be used to prioritize the test cases. These properties are like customer priority, change impact analysis, requirement volatility and developer perceived complexity of requirements.

Model Based TCP: Different models and design artefacts such as use case diagrams, activity diagrams, sequence diagrams and other design artifacts are used to prioritize the test cases.

Cost-Aware TCP: Cost of test cases is considered to prioritize the test cases. The fact is that cost of each test case is not equal.

Other TCP Methods: There are many other approaches like call tree, program slicing, contextual difference methods and interface contact shown in this category.

IV. CONCLUSION

In this study regression testing techniques are under review specially test case prioritization techniques. The main purpose of this review is to identify most basic techniques and major contributions in the test case prioritization domain. There is also a discussion how prioritization techniques are classified. The three major classes of prioritization techniques are type of code coverage data, type of use of feedback and type of code modification data. The most reported types of prioritization techniques are coverage based, fault based, code modification based, code analysis based, use of feedback collected from tester or program data, history based, cost aware, model based and greedy approaches. The common combination observed in these techniques is coverage with

fault detection ability. The primary focus in prioritization techniques is shifted from code analysis to history based due to the fact that fault detection is basic proxy for all prioritization techniques.

When looking at most of the researches performed in the area of Test Case Prioritization, it is observed that the industry has a bias or desire towards artificial prioritization techniques rather than coverage based ones, and the researches done in the area show that the industry yields less positive results from coverage based techniques than artificial ones especially when it is on smaller scale programs and artificial seeds regarding them. Further research in the area is needed to compare both artificial based techniques and coverage based techniques on a larger scale programs or large scale software development in order emphasize the gaps between them more clearly. As seen, in some instances some artificial techniques have proved to drastically loose effectiveness once the size of the program exceeds a certain amount, and the larger it grows the lower the effectiveness becomes.

V. REFERENCES

[1] K. K. Ranga, "Analysis and Design of Test Case Prioritization Technique for Regression Testing", International Journal for Innovative Research in Science and Technology, vol. 2, (2015), pp. 248-252.

[2] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey", Software Testing, Verification and Reliability, vol. 22, (2012), pp. 67-120.

[3] T. Hall, S. Beecham, D. Bowes, D. Gray and S. Counsell, "A systematic literature review on fault prediction performance in software engineering", Software Engineering, IEEE Transactions on, vol. 38, (2012), pp. 1276-1304.

[4] D. C. Kung, J. Gao, P. Hsia, Y. Toyoshima and C. Chen, "On regression testing of object-oriented programs", Journal of Systems and Software, vol. 32, (1996), pp. 21-40.

[5] A. Srivastava and J. Thiagarajan, "Effectively prioritizing tests in development environment", in ACM SIGSOFT Software Engineering Notes, (2002), pp. 97-106.

[6] T. Yu, W. Srisa-an and G. Rothermel, "SimRT: an automated framework to support regression testing for data races", in Proceedings of the 36th International Conference on Software Engineering, (2014), pp. 48-59.

[7] Vengatesan K., and S. Selvarajan"Improved T-Cluster based scheme for combination gene scale expression data" International Conference on Radar, Communication and Computing (ICRCC), pp. 131-136. IEEE (2012).

[8] Kalaivanan M., and K. Vengatesan." Recommendation system based on statistical analysis of ranking from user. International Conference on Information Communication and Embedded Systems (ICICES), pp.479-484, IEEE, (2013).

[9] K. Vengatesan, S. Selvarajan: The performance Analysis of Microarray Data using Occurrence Clustering. International Journal of Mathematical Science and Engineering, Vol.3 (2) .pp 69-75 (2014).

[10] K Vengatesan, V Karuppuchamy, S Pragadeeswaran, A Selvaraj," FAST Clustering Algorithm for Maximizing the Feature Selection in High Dimensional Data", Volume – 4, Issue-2, International Journal of Mathematical Sciences and Engineering (IJMSE), December 2015