REQUIREMENT BASED TEST CASE PRIORITIZATION TECHNIQUES FOR INCREASING EFFICIENCY OF REGRESSION TESTING

Meena Mehta
Assistant Professor

ABSTRACT

Software testing is the most important phase in Software development life Cycle. Failures during testing are probabilistic and their number and pattern can depend on many factors but the important one is test case selection. Test case, test suite prioritization's approaches are used in regression testing. Test case prioritization techniques schedule test cases in an execution order according to some criterion. These criteria can test case cost basis or time basis. Regression testing is the process of verification that previously functioning are remains after the changes. Prioritization concept increases the rate of fault detection or code in time and cost constraints. It also improves the functionality of a quality product. The work is to use a prioritization scheme with one main objective of minimizing the time of test case prioritization.
Key words: Regression testing, *Reliability*.

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 [13]. Software Reliability Engineering is a discipline that aims at ensuring failure free operation of software at the user end employs scientific tools and techniques during testing to remove the maximum number of faults. Software testing is process of executing software in a controlled manner. The objective of testing is to show incorrectness and testing is considered to success when error is detected (Myers 1979). 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. Software testing involves operation of a system or application under controlled conditions and evaluating results. The controlled condition should include both normal and abnormal conditions. The goal of testing is to find bugs and fixed them as early as possible. After all testing is potentially endless process. We cannot test till all the defects are unearthed and removed as it is simply impossible and expensive. At some point, we have to stop testing and ship the software.

When project build is completed it comes in testing phase, in this phase two types of testing are made for a project White Box Testing and Black Box Testing. White box testing is

concerned only with testing the software product as it cannot guarantee that the complete specification has been implemented. White box testing involves looking at the structure of the code. When the internal structure of a product is known, tests can be conducted to ensure that the internal operations performed according to the specification. And all internal components have been adequately exercised. White box testing can be completed by software developer. Because developer know the internal structure of project therefore he/she can find out more bugs Black box testing is concerned only with testing the specification as it cannot guarantee that all parts of the implementation have been tested. Black box is a test design method. Black box testing treats the system as a "Black Box ", it doesn't explicitly use Knowledge of the internal structure. Black box testing is sometimes referred to as functional testing or behavioral testing.

Regression Testing means that test cases from existing test suites to build confidence that software changes have no unintended side-effects. The ideal process would be to create an extensive test suite and tin it after each and every change. Select data for test case design and execution from academic and industrial environment. Define all possible test cases for selected test data problem. Analysis the all possible test suits under regression testing. Rearranging the test cases in test suit for finding the software problem with minimum test case and which takes minimum time to execute test case. Design methodology, which is applicable on different software application / software product for batter solution under regression testing with minimum time and minimum cost. Regression testing is an expensive process. It is compulsory for proper functioning of software product as per current market trend. It is an expensive, time consuming endless process in the software development life cycle. A test case prioritization, assigns each test case a selection probability according to its potential ability to achieve some certain testing goal. It selects prioritized test cases to run to get higher testing performance [6].

Test case prioritization techniques could be of great benefit to increasing the effectiveness of test suites in practice. It helps to increase the rate of fault detection or code whereas Regression test prioritization is often performed in a time constrained execution environment in which testing only occurs for a fixed time period. Some time it is complex and time consuming process due to there may be insufficient resources to allow for the re-execution of all test cases. On this stage test case prioritization techniques are used to handle the situation. Which is aim to improve the effectiveness of regression testing by ordering the test cases so that the most beneficial are executed first. Several test case prioritization techniques are derived to schedule the test case on priority basis for their minimum time execution, minimum cost or minimum weight. Test case prioritization techniques schedule test cases for regression testing in an order that increases their ability to meet some performance goal. One performance goal, rate of fault detection, measures how quickly faults are detected within the testing process.

BACKGROUND

Test case prioritization techniques schedule test cases for execution in an order that attempts to maximize some objective function [1]. They had discussed nine prioritization techniques. No Prioritization, Random Prioritization, Total Statement Coverage Prioritization, Total Branch Coverage Prioritization, Additional Branch Coverage Prioritization, Total Fault-Exposing-Potential Prioritization, Additional Fault-Exposing-Potential Prioritization, Optimal Prioritization and Additional Statement Coverage Prioritization. They have used a weighted average of the percentage of faults detected, or APFD on a test suite of five different test cases. They have arranged test cases in different sequence to find out percentage of maximum fault detection. According to them Test case prioritization techniques schedule test cases for execution in an order that attempts to Increase their effectiveness at meeting some performance goal.

Analyzing Regression Test Selection Techniques [2] has been proposed, Regression testing is a necessary but expensive maintenance activity aimed at showing that code has not been adversely affected by changes. Regression test selection techniques reuse tests from an existing test suite to test a modified program. Many regression test selection techniques have been proposed: however, it is difficult to compare and evaluate these techniques because they have different goals. The issues relevant to regression test selection techniques, and uses these issues as the basis for a framework within which to evaluate the techniques.

Efficient Regression Test Selection Technique [3] G. Rothermel and M. J. Harrold. has given that Regression testing is an expensive but necessary maintenance activity performed on modified software to provide confidence that changes are correct and do not adversely affects other portions of the software. A regression test selection technique chooses, from an existing test set , tests that are deemed necessary to validate modified software. They present a new technique for regression test selection. Their algorithms construct control flow graphs for a procedure or program and its modified version, and use these graphs to select tests that execute changed code from the original test suite. They proved that under certain conditions, the set of tests technique selects includes every test from the original test suite that can expose faults in the modified procedure or program. Under these conditions their algorithms are safe. Moreover, t h e i r algorithms may select some tests that cannot expose faults, they are at least as precise as other safe regression test selection algorithms, unlike many other regression test selection have implemented their algorithms. Initial empirical studies indicated that technique can significantly reduce the cost of regression testing modified software.

Two types of test case prioritization are distinguishes [6] [4] in general and version-specific. In general test case prioritization, given program P and test suite T, test cases in T are prioritized with the goal of finding a test case order that will be useful over a sequence of subsequent

modified versions of P. In contrast, in version-specific test case prioritization, given program P and test suite T, test cases in T are prioritized with the intent of finding an ordering that will be useful on a specific version P0 of P. Version specific prioritization is performed after a set of changes have been made to P and prior to regression testing P0.

Regression testing is an expensive process used to validate modified software. Test case prioritization techniques improve the cost effectiveness of regression testing by ordering test cases such that those that are more important are run earlier in the testing process.

Test case prioritization [21][13] is important in regression testing [21][22]. It schedules the test cases in a regression test suite with a view to maximizing certain objectives (such as revealing faults earlier), which help reduce the time and cost required to maintain service-oriented business applications test case prioritization techniques.

METHODOLOGY

Software Testing

It is an important phase of software development life cycle. The objective of software testing is to show incorrectness and when error is detected testing is considered to success. The purpose of software testing is to detect errors /faults/bugs, not to correct them in the software. Aim of software testing is to produce a software product that is economic, errorless and useful and safe for people. Goal of software testing is to find bugs and fixed them as early as possible.

Test case, Test suite

It is a set of test data and test programs (test scripts) and their expected results.
It is a collection of test scenarios and / or test cases that are related or that may cooperate with each other. In other words Test suit contains set of test cases, which are both related and unrelated. Test scenario is set of test cases that ensure that the business process flows are tested from end to end. Regression Testing means rerunning existing tests against the modified code to determine whether the changes break anything that worked prior to the change and by writing new tests where necessary. In other word Regression Testing means that test cases from existing test suites to build confidence that software changes have no unintended side- effects.

Test Case Prioritization Techniques

It provides another method for assisting with regression testing. These techniques let testers order their test cases so that those test cases with the highest priority, according to some criterion, are executed earlier in the regression testing process than lower priority test cases. For example: Testers might wish to schedule test cases in an order that achieves code coverage at the fastest rate

possible, exercises features in order of expected frequency of use, or exercises subsystems in an order that reflects their historically demonstrated propensity to fail.

Goal of Test Case Prioritization

Test case prioritization schedule test cases in order to increase their ability to meet some performance goal: Rate of fault detection Rate of code coverage, Rate of increase of confidence in reliability. **Rate of fault detection** is one of the goal to measure - How quickly faults are detected within the testing process.

Average Percentage Of Faults Detected (Apfd) Metric

To quantify the goal of increasing a subset of the test suite's rate of fault detection, used a metric called APFD developed by Elbaum et al. [1,2,4] that measures the average rate of fault detection per percentage of test suite execution. The APFD is calculated by taking the weighted average of the number of faults detected during the run of the test suite. APFD can be calculated using a notation m -> the number of faults contained in the program under test P n -> The total number of test cases and TFi -> The position of the first test in T that exposes fault i. APFD = 1 – TF1 + TF2+ ..... + TFm + 1 nm 2n So as the formula for APFD shows that calculating APFD is only possible when prior knowledge of faults is available. APFD calculations therefore are only used for evaluation.

Time based test case prioritization

There are many existing approaches to regression test prioritization that focus on the coverage of or modifications to the structural entities within the program under test [3], [4], [5]. None of these prioritization schemes explicitly consider the testing time budget like the time-aware technique presented in paper [6] Elbaum et al. and Rothermel et al. focus on general regression test prioritization and the identification of a single test case reordering that will increase the effectiveness of regression testing over many subsequent changes to the program. They had described a time-aware test suite prioritization technique. Experimental analysis demonstrates that their approach can create time-aware prioritizations that significantly outperform other prioritization techniques.

Measures of Time-Aware Test Suite Prioritization

Kristen R. Walcott et al [6] described other factors of test case prioritization i.e. Time-Aware Test Suite Prioritization. Problem description: Time aware test suit prioritization given:

(i) A test suite, T,

(ii) The collection of all permutations of elements of the power set of permutations of T, perms (2T)

(iii) The time budget, tmax, and

(iv) Two functions from perms (2T) to the real numbers, time and fit.

For Example, suppose that regression test suite T contains six test cases with the initial ordering for T that contains (T1; T2; T3; T4; T5; T6) as described in Table 1. For the purposes of motivation, this example assumes a priori knowledge of the faults detected by T in the program P. As shown in table1 (a), test case T1 can find seven faults, (F1, F2,,F4,F5,F6,F7,f8) in nine minutes, T2 finds one fault, (F1), in one minute, and T3 isolates two faults, (F1,F5), in three minutes. Test cases T4, T5 and T6 each find three faults in four minutes (F2,F3,F7), (F4,F6,F8) and (F2,F4,F6) respectively.

Table 1

	F1	F2	F3	F4	F5	F6	F7	F8
T1	X	X		X	X	X	X	X
T2	X							
T3	X				X			
T4		X	X				X	
T5				X		X		X
T6		X		X		X		

Table (a)

Test Case	No. of faults	Time Cost(mins)	Avg.Faults per Min.
T1	7	9	0.778
T2	1	1	1.0
T3	2	3	0.667
T4	3	4	0.75
T5	3	4	0.75
T6	3	4	0.75

Suppose that the time budget for regression testing is twelve minutes. Because we want to find as many faults as possible early on, it would seem intuitive to order the test cases by only considering the number of faults that they can detect. Without a time budget, the test tuple (T1; T4; T5; T6; T3; T2) would execute. Out of this, only the test tuple P1= T1 would have time to run when under a twelve minute time constraint and would only a total of seven faults, as noted in Table 1(b). Since time is a principal concern, it may also seem intuitive to order the test cases with regard to their execution time. In the time constrained environment, a time-based prioritization P2 = (T2;T3;T4;T5) could be executed and find eight defects, as described in

Table-1(b).

Another option would be to consider the time budget and fault information together. To do this, we could order the test cases according to the average percent of faults that they can detect per minute. Under the time constraint, the tuple P3 = (T2; T1) would be executed and find a total of seven faults. If the time budget and the fault information are both considered intelligently, that is, in a way that accounts for overlapping fault detection, the test cases could be better prioritized and thus increase the overall number of faults found in the desired time period. In this example, the test cases would be intelligently reordered so that the tuple P4 = (T5; T4; T3) would run, revealing eight errors in less time than P2.

Table 1(b) Time l imit 12 minutes

	Fault P1	**Time P2**	**APFD P3**	**Intelligent P4**
Test Suits	T1	T2 T3 T4	T2 T1	T5 T4 T3
Total faults	7	8	7	8
Total Time	9	12	10	11

Finally, it is important to note that the first two test cases of P2, T2 and T3, find a total of two faults in four minutes whereas the first test case in P4 T5, detects three defects in the same time period. Therefore, the intelligent prioritization, P4, is favored over P2 because it is able to detect more faults earlier in the execution of the tests.

CONCLUSION & RESULT

Test case prioritization techniques schedule test cases for execution in an order that attempts to Increase their effectiveness at meeting some performance goal. Arranged test cases in different sequence to find out percentage of maximum fault detection. Test case prioritization techniques schedule test cases for execution in an order that attempts to Increase their effectiveness at meeting some performance goal.

Regression testing is an expensive process used to validate modified software. Test case

prioritization techniques improve the cost effectiveness of regression testing by ordering test cases such that those that are more important are run earlier in the testing process. Test case prioritization techniques (e.g., [8, 9, 10, 11, 12, 13, 14, 15, 16, and 17]) offer an alternative approach to improving regression testing cost-effectiveness. Prioritization can improve cost effectiveness in two ways. First, prioritization can help engineers reveal faults early in testing, allowing them to begin debugging activities earlier in the testing cycle than might otherwise be possible. Second, in the case in which testing activities are cut short and test cases must be omitted, prioritization can improve the chances that important test cases will have been executed. In this case, cost savings related to early fault detection (by those test cases that are executed) still apply, and additional benefits accrue from lowering the number of faults that might otherwise be missed through less appropriate runs of partial test suites.

Test case prioritization [16] is important in regression testing [21][22]. It schedules the test cases in a regression test suite with a view to maximizing certain objectives (such as revealing faults earlier), which help reduce the time and cost required to maintain service-oriented business applications. Existing regression testing techniques for such applications focus on testing individual services [23] or workflow programs [20].

Regression testing is the verification that previously functioning software remains after a change. Regression testing is time consuming and expensive process. A large number of test case executions are expensive and time consuming during regression testing, where Test case prioritization is an effective and practical technique in regression testing to reduce it. It schedules test cases in order of precedence that increases their ability to meet some performance goals, such as code coverage, rate of fault detection. To schedule test case in test suit on priority basis, earlier, different metric are described: APFD, APFDc, time base test case prioritization, code coverage based prioritization. First I have reviewed importance of regression testing in software development and empirical approach of software testing along with testing models and techniques. After reviewing research papers based on software testing, regression testing. A prioritization of test case in regression testing is examined. There are so many factors and problems are exists to properly test a software. The existing gap between development and testing software can be eliminated only by using STLC in SDLC towards the industrial realities.

BIBLIOGRAPHY

[1] G. Rothermel et al “Test Case Prioritization: An Empirical Study” Proceedings of the International Conference on Software Maintenance, Oxford, UK, September, 1999

[2] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold. Test case prioritization: An empirical

study.In Proceedings of the International Conference on Software Maintenance, pages 179{188, August 1999.

[3] G. Rothermel and M. J. Harrold. A safe, efficient regression test selection technique. ACM Transactions on Software Engineering and Methodology, 6(2):173{210, April 1997.

[4] G. Rothermel, R.H. Untch, C. Chu, and M. J. Harrold. Prioritizing test cases for regression testing. IEEE Trans. Softw. Eng.

[5] K. Onoma, W.-T. Tsai, M. Poonawala, and H. Suganuma. Regression testing in an industrial environment. Comm. ACM,
41(5):81-86, May 1998.

[6] S. Elbaum, A. Malishevsky, and G. Rothermel. Prioritizing test cases for regression testing. Proc. Int'l Symp. Softw. Testing and Analysis, pages 102-112, Aug. 2000.

[7] R. C. Bryce and C. J. Colbourn. The density algorithm for pairwise interaction testing. Journal of Software Testing, Verification, and Reliability, to appear.

[8] Ren´ee C. Bryce & Atif M. Memon DoSTA'07, September 4, 2007, Dubrovnik, Croatia.ACM

[9] D. Jeffrey and N. Gupta. Test case prioritization using relevant slices. In Int'l.Comp. Softw. Appl. Conf., pages 411–420, Sept. 2006.

[10] J. Kim and A. Porter. A history-based test prioritization technique for regressiontesting in resource constrained environments. In Int'l. Conf. Softw. Eng., pages 119–129, May2002.

[11] D. Leon and A. Podgurski. A comparison of coverage- based and distribution-based techniques for filtering and prioritizing test cases. In Int'l. Symp. Softw. Rel. Eng., pages 442–453, Nov. 2003.

[12] Z. Li, M. Harman, and R. M. Hierons. Search algorithms for regression test case prioritization. IEEE Trans. Softw. Eng.,33(4):225–237, Apr. 2007.

[13] G. Rothermel, R. Untch, C. Chu, and M. J. Harrold. Test case prioritization. IEEE Trans. Softw. Eng., 27(10):929–948, Oct. 2001.

[14] F. Shull et al. What we have learned about fighting defects.In Int'l. Softw. Metrics Symp., pages 249–258, June 2002.

[15] A. Srivastava and J. Thiagarajan. Effectively prioritizing tests in development environment. In Int'l. Symp. Softw. Test. Anal., pages 97–106, July 2002.

[16] A. Walcott, M. L. Soffa, G. M. Kapfhammer, and R. Roos. Time-aware test suite prioritization. In Int'l. Symp. Softw. Test. Anal., pages 1–12, July 2006.

[17] W. Wong, J. Horgan, S. London, and H. Agrawal. A study of effective regression testing in practice. In Int'l. Symp. Softw. Rel. Engr., pages 230–238, Nov. 1997.

[18]Lijun Mei, Zhenyu Zhan et al WWW'09, April 20–24, 2009, Madrid, Spain.ACM 978-1-60558-487-4/09/04.

[19] S.-S. Hou, L. Zhang, T. Xie, and J.-S. Sun. Quota- constrained test-case prioritization for regression testing of service-centric systems. In Proceedings of the IEEE International Conference on Software Maintenance (ICSM2008), pages 257–266. 2008.

[20] Z. Li, M. Harman, and R. M. Hierons. Search algorithms for regression test case prioritization. IEEE TSE, 33 (4): 225–237, 2007.

[21] A. K. Onoma, W.-T. Tsai, M. Poonawala, and H. Suganuma. Regression testing in an industrial environment. Communications of the ACM, 41 (5): 81–86, 1998.

[22] M. E. Ruth and S. Tu. Towards automating regression test selection for Web services. In Proceedings of the 16th International Conference on World Wide Web (WWW 2007), pages 1265–1266. 2007.

[23] Web Services Description Language (WSDL) 1.1. W3C, 2001. Available at http://www.w3.org/TR/wsdl.