Effectiveness of Testcase Prioritization using APFD Metric: Survey

R. Pradeepa
Assistant Professor,
Department of Computer Applications,
Sri Ramakrishna Engineering College,
Coimbatore, Tamilnadu, India

K. VimalaDevi, Ph.D.
Professor,
Department of Computer Science,
Kalasalingam University,
Krishnankovil, Tamilnadu, India

ABSTRACT

Test case prioritization techniques involve scheduling over test cases in an order that improves the performance of regression testing. It is inefficient to re execute every test cases for every program function if once change occurs. Test case prioritization is to be scheduled based on higher priority than lower priority to meet some performance goal (i.e. increase in the effectiveness of testing). The performance goals are 1. rate of fault detection (how quickly faults are detected) 2. Rate of code coverage at fastest rate,3) Rate of increase of confidence in reliability during the testing process to improve the software quality. The problem of test case selection can be solved by prioritizing the test case. The main aim of my paper is to determine the effectiveness of prioritized and non-prioritized test case with the help of APFD(Average Percentage Faults Detected).

Keywords

Test Case Prioritization, Regression Testing, Average Percentage of Faults Detected (APFD), Test Cases.

1. INTRODUCTION

Regression means retesting the unchanged parts of the application. Test cases are re-executed in order to check whether previous functionality of application is working fine and new changes have not introduced any new bugs. This test can be performed on a new build when there is significant change in original functionality or even a single bug fix. This is the method of verification. Verifying that the bugs are fixed and the newly added features have not created in problem in previous working version of software. Testers perform functional testing when new build is available for verification. The intend of this test is to verify the changes made in the existing functionality and newly added functionality. When this test is done , the tester should verify if the existing functionality is working as expected and new changes have not introduced any defect in functionality that was working before this change. Regression test should be the part of release cycle and must be considered in test estimation.

Regression testing is usually performed after verification of changes or new functionality. But this is not the case always. For the release which is taking months to complete, regression tests must be incorporated in the daily test cycle.

For weekly releases regression tests can be performed when functional testing is over for the changes. By reducing the cost of regression testing and increasing the likely effectiveness of running the test suite in a time-constrained execution environment, developers can afford higher levels of verification. When experimenting with prioritization techniques, regression faults can be obtained in two ways : by locating natural faults and by the seeding faults[20].

Test case prioritization techniques could be of great benefit to increasing the effectiveness of test suites in practice. Test case prioritization is a technique helps to increase the rate of fault detection. In an empirical evaluation of regression test suite prioritization technique ordering was measured using an evaluation metric called APFD(Average Percentage Faults Detected) and PTR (Problem Tracking Report)

2. A SURVEY OF RECENT RESEARCH IN THE FIELD USING APFD METRIC

For test case prioritization process in regression testing using APFD, various researchers have proposed several researches contributions . A brief review of some important contributions from the existing literature is presented in this section.

Rothermel et al.[10] compared the proposed prioritisation techniques like random prioritisation, optimal prioritisation, and no prioritisation, using the Siemens suite programs. Optimal prioritisation is possible because the experiment was performed in a controlled environment, i.e. the faults were already known. The results show that all the proposed techniques produce higher APFD values than random prioritization or no prioritisation. The surrogate with the highest APFD value differes between programs, suggesting that there is no single best surrogate.

Do and Rothermel applied coverage-based prioritisation techniques to the JUnit testing environment, a popular unit testing framework [23]. The results showed that prioritised execution of JUnit test cases improved the fault detection rate. One interesting finding is that the random prioritisation sometimes resulted in an APFD value higher than the untreated ordering, i.e. the order of creation. When executed in the order of creation, newer unit tests will be executed later. However, newer unit tests will never have a higher chance of detecting faults. The empirical results showed that random prioritisation could exploit this weakness of untreated ordering in some cases.

Praveen Ranjan Srivastava [12] has presented a new test case prioritization algorithm to compute average faults discovered per minute. Using APFD metric results, he has demonstrated the effectiveness of the algorithm and

presented. Calculating the effectiveness of prioritized and non-prioritized cases by means of APFD has been his main objective.

R.Krishnamoorthi et al. [21] have proposed a Genetic Algorithm (GA) based new test case prioritization method. A superior rate of fault detection when compared to rates of randomly prioritized test suites has been obtained, when the new suite that consists of subsequences of the original test suite prioritized by the proposed technique is executed within a time-constrained execution environment. Test cases have been prioritized utilizing structurally based criterion by the experiment and the genetic algorithm has been analyzed with regard to effectiveness and time overhead. The effectiveness of the new test case orderings have been calculated using an Average Percentage of Faults Detected (APFD) metric.

R. Kavitha et al [5] have proposed an algorithm that performs rate of fault detection and fault impact based prioritization of test cases. Experimental results using APFD have demonstrated that more effective severe fault identification at earlier stage of the testing process could be obtained by the proposed algorithm for prioritized test cases compared to unprioritized ones.

Zheng Li *et al.* [22] have tested experimentally that genetic algorithms perform well for test case prioritization. The benefits of code coverage based prioritization techniques are measured using a weighted average of the percentage of faults detected (APFD) average percentage block coverage (APBC), average percentage decision coverage (APDC) and average percentage statement coverage (APSC).

3.PRIORITIZED TEST SUITE EFFECTIVENESS

The performance of the prioritization technique used in this paper, it is must to assess effectiveness of the sequence/ordering of the test suite. Effectiveness will be measured by the rate of faults detected. The following metric is used to calculate the level of effectiveness.

3.1 Average Percentage Of Faults Detected (APFD) Metric

To quantify the goal of increasing a subset of the test suite's rate of fault detection, we use a metric called APFD developed by Elbaum et al.[6] that measures the rate of fault detection per percentage of test suite execution. The APFD is calculated by taking the weighted average of the percentage of faults detected during the execution of the test suite. APFD values range from 0 to 100; higher values imply faster (better) fault detection rates. APFD can be calculated as follows:

$$APFD=1-\{(Tf1+Tf2+\ldots.+Tfm)/mn\}+(1/2n) \quad (1)$$

Where n be the no. of test cases and m be the no. of faults. (Tf1,….,Tfm) are the position of first test T that exposes the fault.

Table 1. Fault Matrix

Faults	Test Cases									
	T 1	T 2	T 3	T 4	T 5	T 6	T 7	T 8	T 9	T 10
f 1						X			X	
f 2		X						X		
f 3							X			X
f 4				X					X	
f 5		X				X				
f 6	X									
f 7	X							X		
f 8							X		X	
f 9				X						X
f 10		X					X			

To illustrate this measure, consider the program with 10 faults and a suite of 10 test cases 1 through 10 as shown in Table 1.

Here comparison among the results of prioritized and non-prioritized suite is done based on the results of the APFD metric. This is average percentage of faults detected. APFD is a standardized metric that is used to find the degree of faults detected.

The prioritized order according to fi is:
T4 T2 T1 T7 T6 T9 T10 T5 T8 T3
No. of test cases (n) = 10
No. of faults (m) = 10
The position of the first test in T that exposes fault i. = Tfi

Applying APFD w.r.t. the prioritized test cases:
APFD = 1 – {(5 + 2 + 4 + 1 + 2 + 3 + 3 + 4 + 1 + 2) / (10*10)} + {1/(2*10)}
= 1 – { 27 / 100} + { 1 / 20}
= 1 – 0.27 + 0.0
= **0.78**

Now APFD value for non – prioritized test cases:
APFD = 1 – {(6 + 2 + 7 + 4 + 2 + 1 + 1 + 7 + 4 + 2) / (10*10)} + {1/(2*10)}
= 1 – { 36 / 100} + { 1 / 20}
= 1 – 0.36 + 0.05
= **0.69**

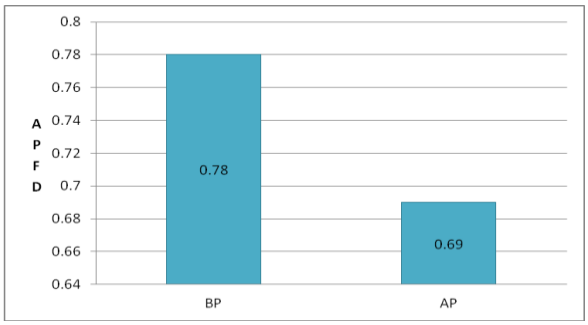

Figure 1 : APFD is higher for prioritized test case order that reveal most faults early.

Thus the prioritized test cases yield better fault detection than the non – prioritized test cases as shown in the chart.

3.1.1 Limitations of the APFD Metric

The APFD metric just presented relies on two assumptions: (1) all faults have equal severity, and (2) all test cases have equal costs. (These assumptions are manifested in the fact that the metric simply plots the percentage of faults detected against the fraction of the test suite run.) Our previous empirical results [6, 7] suggest that when these assumptions hold, the metric operates well. In practice, however, there are cases in which these assumptions do not hold: cases in which faults vary in severity and test cases vary in cost. In such cases, the APFD metric can provide unsatisfactory results.

3.2 Average Percentage Block Coverage (APBC).

This measures the rate at which a prioritized test suite covers the blocks.

3.3. Average Percentage Decision Coverage (APDC).

This measures the rate at which a prioritized test suite covers the decisions (branches).

3.4. Average Percentage Statement Coverage (APSC).

This measures the rate at which a prioritized test suite covers the statements.

3.5. Average Percentage Loop Coverage (APLC).

This measures the rate at which a prioritized test suite covers the loops.

3.6. Average Percentage Condition Coverage (APCC).

This measures the rate at which a prioritized test suite covers the conditions.

3.7. Problem Tracking Reports (PTR) Metric

The PTR metric is another way that the effectiveness of a test prioritization may be analyzed. Recall that an effective prioritization technique would place test cases that are most likely to detect faults at the beginning of the test sequence. It would be beneficial to calculate the percentage of test cases that must be run before all faults have been revealed. PTR is calculated as follows:

$$Ptr(t,p) = nd/n$$

Let t - be the test suite under evaluation, n - the total number of test cases in the total number of test cases needed to detect all faults in the program under test p

3.7.1 Limitations of the PTR Metric

However, the numerator of the PTR equation requires the knowledge of the minimal number of test cases needed to detect all faults. While it is easy to calculate the maximum number of tests needed, test set size minimization is equivalent to the NP-complete minimal set covering problem.

4. REGRESSION TESTING TECHNIQUES

There are number of available regression testing techniques. Here we are representing all these techniques in basic 3 categories as defined .

(i) Retest All: As the name suggest in this testing technique we perform whole testing cycle again after the inclusion of new code and component and related test cases into it. Again the test cases will be generated, sequence reset etc. This type of technique is not feasible in most of time, as it requires much time and cost. But in smaller software where a small change in code impact on whole software at that time regression testing is used.

(ii) Regression Test Selection: This approach is a modification over the existing retest all approaches. In this approach instead of testing all cases a selection on the test cases is performed. To perform this selection a test cases categorization is performed. According to this rest table cases are separated from whole test cases such as the requirement based testing is generally need not to be performed again. The code based test cases and the system based test cases are selected to perform the testing process. In this technique instead of rerunning the whole test suite, we select a part of test suite to rerun if the cost of selecting a part of test suite is less than the cost of running the tests that RTS allows us to omit. RTS divides the existing test suite into (1) Reusable test cases; (2) Re-testable test cases; (3) Obsolete test cases.

(iii) Test Case Prioritization: All the test cases used in a testing approach or the sequence are not alike. It means each kind of test cases have there on values called the basic prioritization of the test cases. Generally the prioritization process is defined on the bases of state space diagram of the cases. The test cases that exist on initial stage of the test cases or the development process have the lower priority and the test cases that affect the whole system or tested repeatedly over the whole process having the higher priority. Besides this the prioritization process is further divided in number of sub techniques to assign the priorities

a)The easiest type of assigning priorities is the random prioritization but in most of the cases it does perform the complete justification with the test cases selection. Because of this such type of technique is never recommended to generate the test cases.

(b) Optimal ordering: in which the test cases are prioritized to optimize rate of fault detection. As faults are determined by respective test cases and we have programs with known faults, so test cases can be prioritized optimally. It is one of the dynamic prioritization approach in which decision is affected because of types of occurred faults and there frequency.

(c) Total statement coverage prioritization: in which test cases are prioritized in terms of total number of statements by sorting them in order of coverage achieved. If test cases are having same number of statements they can be ordered pseudo randomly.

(d)Additional statement coverage prioritization: which is similar to total coverage prioritization, but depends upon feedback about coverage attained to focus on statements not yet covered. This technique greedily selects a test case that has the greatest statement coverage and then iterates

until all statements are covered by at least one test case. The moment all statements are covered the remaining test cases undergo Additional statement coverage prioritization by resetting all statements to "not covered".

5.CONCLUSION

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 (TCP) is an effective and practical technique in regression testing to reduce it. It schedules test cases in the order of precedence that increases their ability to meet some performance goals, such as code coverage, rate of fault detection through APFD metric. Analysis is done for prioritized and non-prioritized cases with the help of APFD (average percentage fault detection) metric. It is proven that when the prioritized cases are run then result is more efficient. In future, test case prioritization can be done by using more factors and evaluate by PTR, Weighted Defect Density (WDD), Defect Removal Efficiency (DRE), Defect Removable Efficiency (DRE), Weighted Percentage Based on Fault Severity (WPFS) and risk metrics. We conclude that prioritization of test case or test suits have different aspects of fault detection. On the basis of prioritization techniques, functionality of regression testing can be improved in minimum time and recourses. This can support to make a better software product.

6. ACKNOWLEDGEMENTS

We would like to express my heartiest gratitude to all the people who poured their efforts in compilation of this work. We would like to thank almighty for giving us strength to pull through this task and to all the individuals who gave their best contribution in the related field of research.

7. REFERENCES

[1] Sanjukta Mohanty, Arup Abhinna Acharya and Durga Prasad Mohapatra, (2011), "A Survey On Model Based Test Case Prioritization". International Journal of Computer Science and Information Technologies.

[2] Sahil Gupta, Himanshi Raperia, Eshan Kapur, Harshpreet Singh and Aseem Kumar, (2012), "A Novel Approach For Test Case Prioritization". International Journal of Computer Science, Engineering and Applications.

[3] Prakash Srivastava, "Performance Evaluation of Cost-cognizant Test Case Prioritization". International Journal of Computer Science and its Applications.

[4] Sanjukta Mohanty , Arup Abhinna Acharya and Durga Prasad Mohapatra, (2011), A Survey On Model Based Test Case Prioritization. International Journal of Computer Science and Information Technologies.

[5] R. Kavitha and N. Sureshkumar, (2011) , "Factors Oriented Test Case Prioritization Technique in Regression Testing". European Journal of Scientific Research.

[6] S. Elbaum, A. Malishevsky, and G. Rothermel.(2000), "Prioritizing test cases for regression testing".

[7] G. Rothermel, R. Untch, C. Chu, and M. J. Harrold. "Test case prioritization: an empirical study".

[8] S. Yoo, M. Harman, (2007) Regression Testing Minimisation, Selection and Prioritisation : A Survey. Software Testing, Verification And Reliability .

[9] Alexey G. Malishevsky, Joseph R. Ruthruff, Gregg Rothermel, Sebastian Elbaum, (2006) , Costcognizant Test Case Prioritization.

[10] S.Elbaum, A.Malishevsky, and G.Rothermel, (2002), Test case prioritization: A family of empirical studies. IEEE Transactions on Software Engineering.

[11] Dr.Varun Kumar, Sujata and Mohit Kumar ,(2011), "Testcase Prioritization Using Fault Severity", IJCST.

[12] Praveen Ranjan Srivastava, (2008) "Test Case Prioritization", Journal of Theoretical and Applied Information Technology IEEE.

[13] Gregg Rothermel, Roland H. Untch, Chengyun Chu, Mary Jean Harrold, (1999) "Test Case Prioritization: An Empirical Study", International Conference.

[14] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, (2001) "Prioritizing Test Cases for Regression Testing", IEEE Transactions on Software Engineering.

[15] Siripong roongruangsuwan, Jirapun daengdej, (2010) "Test case prioritization techniques", Journal of Theoretical and Applied Information Technology, IEEE.

[16] Sebastian Elbaum, Alexey G. Malishevsky and Gregg Rothermel, (2002) "Test case prioritization: A family of empirical studies," IEEE Transactions on Software Engineering.

[17] Gaurav Duggal, Mrs. Bharti Suri , "Understanding regression testing techniques", Guru Gobind Singh Indraprastha University, Delhi, India.

[18] Sebastian Elbaum , Alexey Malishevsky , Gregg Rothermel, (2001) "Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization" , Proceedings of the 23rd International Conference on Software Engineering.

[19] Srinivasan Desikan,(2006),"A test methodology for an effective regression testing".