An Empirical Approach for Test Case Prioritization

Himanshi Raperia, Saurabh Srivastava

***Abstract*—** Regression testing is one of the most critical activities of software development and maintenance. Whenever software is modified, a set of or all the test cases are run and the comparison of new outputs is done with the older one to avoid unwanted changes. If the new output and old output match it implies that the modifications made in one part of the software didn't affect the remaining software. It is impractical to re-execute every test case for a program if changes occur. In this paper an algorithm is proposed to prioritize test cases based on rate of fault detection and impact of fault. The proposed algorithm identifies the severe fault at earlier stage of the testing process and the effectiveness of prioritized test case and comparison of it with unprioritized ones with the help of APFD.

Index Terms —software testing;regression testing; test case; test case prioritization; fault detection; fault impact; APFD

1 INTRODUCTION

Software Testing forms one of the indispensable parts of the software development life cycle and heavily depends upon the design of the software. It also depends upon the software tester, guidelines of the organization, which developed the software and various other factors. Regression testing is a necessary but expensive process in software lifecycle. One of regression testing approach, test case prioritization, aims at sorting and executing test case in the order of their potential abilities to achieve certain testing objective.

Software developers often save the test suites, so that they can reuse them, when software undergoes changes. Running all test cases in an existing test suite can consume enormous amount of time. For example a product that contains approximately 20,000 lines of code running an entire test suits requires seven weeks. Researchers have found various algorithms to reduce the cost of regression testing and also to increase the effectiveness of testing [ZHE2007] Dennis Jeffrey and Neelam Gupta [DEN2007] have tested experimentally by selectively retaining test cases during test suite reduction.

In [JEN2004], they have empirically evaluated several test case filtering techniques that are based on exercising information flows. The other way of testing is to order the test case based on some criteria to meet some performance goal. Testers may want to order their test cases so that those test cases with the highest priority according to some criterion are run first. So test case prioritization technique do not discard test cases, they can avoid the drawback of test case minimization techniques. The software is successful when Quality of software is maximized, cost should be minimized and the product should be delivered to the customer in time [JKA1997], [MAR2006], [ALE2002].

In [SEB2002], Sebastian Elbaum et. Al. investigated several prioritization techniques such as total statement coverage prioritization and additional statement coverage, to improve the rate of fault detection. There are varieties of testing criteria that have been discussed and the different testing criteria are useful for identifying test cases that exercise different structural and functional elements in a program, and therefore the use of multiple

testing criteria can be effective at identifying test cases that are likely to expose different faults in a program. In this paper, one new approach to prioritize the test cases at system level for regression test cases is proposed. This technique identifies more severe faults at an earlier stage of the testing process.

Factors proposed to design algorithm are:

1) Rate of faults detection (how quickly the faults are identified
2) Impact of Fault. We can analyze the test cases by feeding faults, invariant of the severity into any program.

The APFD metric relies on two assumptions: (1) all faults have equal costs (hereafter referred to as fault severities), and (2) all test cases have equal costs (hereafter referred to as test costs). Earlier empirical results 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.

2 RELATED WORK

Test case prioritization is an effective and practical technique that helps to increase the rate of regression fault detection then software evolves. Defect free software increases the confidence of customer in the software, so it is important for any software organization to develop techniques that helps software testers to detect faults within specified time and cost. Test case prioritization is technique which prioritizes test cases for

- *Himanshi Raperia is currently teaching as an assistant professor in computer science department at lovely professional University, India, PH-8960097227. E-mail: er.raperia@gmail.com*
- *Saurabh Srivastava is currently pursuing masters degree program in computer science and engineering in lovely professional University, India, PH-8699499269. E-mail: iam100rabh@gmail.com*

regression testin so that maximum number of faults can be detected without compromising with cost and time. Earlier work describes the code coverage based TCP Strategies and their benefits. Coverage based TCP done their prioritization based on their coverage of statements.

Zheng Li, Mark Harman, and Robert M. Hierons in their research paper "Search Algorithms for Regression Test Case Prioritization" published in 2007 that for prioritizing statement coverage the test cases are ordered based on the number of statements executed or covered by the test case such that the test cases covering maximum number of statements would be executed first. Some of the other techniques are branch coverage and function coverage. In this method test cases are prioritized based on their number of branch or function coverage by test case respectively.

The benefits of the code coverage strategies were measured using weighted average of the percentage of branch covered (APBC), percentage of decision covered (APDC) and percentage of statement covered (APSC) .APBC is the rate of coverage of blocks during testing process, APDC is a measure of rate of coverage of decisions for a test suite and the APSC is a measure of rate of coverage of statements during test suite. The disadvantage of the above method is that no importance for fault. My aim is to give equal weightage of rate of fault detection and also identification of severe faults at the earlier stages of the testing process.

Several case studies demonstrate the benefits of code coverage based TCP strategies. Researchers have used various prioritization techniques to measure APFD values and found statistically significant results. The APFD value is a measure that shows how quickly the faults are identified for a given test suite set. The APFD values range from 0 to 100 and the area under the curve by plotting percentage of fault detected against percentage of test cases executed.

The code coverage-based TCP strategies were shown to improve the rate of fault detection, allowing the testing team to start debugging activities earlier in the software process and resulting in faster software release to the customer. If all the faults are not equally severe, then APFD leads misleading information. The fault impact value also has to be considered to prioritize the test cases.

3 PROPOSED PRIORITIZATION TECHNIQUE:

A. Factors to be considered for prioritization

Two factors are used for prioritization. These two factors are discussed below, and the reasoning of why they were chosen for prioritization technique:

1) Rate of fault detection:

The average number of faults detected per minute by a test case is called rate of fault detection. The rate of fault detection of test case i have been calculated using the number of faults detected and the time taken to find out those faults for each test case of test suite.

*RFi = ((number of faults) / time) * 10 …… (1)*

Every factor is converted into 1 to 10 point scale. The reason being, earlier work may take long time (may be several months or a year) depending on the size of the test suite and how long each test case takes to run. The technique implements a new test case prioritization technique that prioritize the test cases with the goal of giving importance of test case which have higher value for rate of fault detection and severity value.

2) Impact of fault:

Testing efficiency can be improved by focusing on the test case that is likely to cover high number of severe faults. So, for each fault severity value was assigned based on impact of the fault on the product. Severity value has been assigned based on a 10 point scale as shown below:

- Complex (Severity 1): SM value of 9-10
- Moderate (Severity 2): SM of 6
- Low (Severity 3): SM of 4
- Very Low (Severity 4): SM of 2

Once the fault has been detected then we assign some severity measure to each fault according to the type of the fault. For example we can assign the severities to the faults in decreasing order of the severity as follows.

Timing/ serialization → 10

Function → 9

Unknown →8

Assignment → 7

Environment → 6

Interface → 5

Algorithm→ 4

Data→ 3

Checking → 2

GUI →1

Total Severity of Faults Detected (TSFD) in the module is the summation of severity measures of all faults identified for a module.

$$TSFD = \sum_{i=1}^{i=n} SM \text{ (severity measure)}$$

This equation shows TSFD for a module where n represents total number of faults identified for the module.

According to the equation (5) the severity value of each test case can be calculated as follows:

1. T1 = 6
2. T2 = 6
3. T3 = 6
4. T4 = 10
5. T5 = 8
6. T6 = 10
7. T7 = 4
8. T8 = 20
9. T9 = 12
10. T10 = 6

Equation (2) shows that the severity value of test case i, where t represent number of faults identified by the ith test case.

$$S_i = \sum_{f=1}^{t} SV$$

If Maximum(S) is the high severity value of test case among all the test cases then fault impact of ith test case is shown below:

*FIi = (Si/ Maximum(S)) * 10 .....(3)*

	Assign Severity Value									
Fault detect test case	1	2	3	4	5	6	7	8	9	10
T1		x		x						
T2	x				x					
T3	x				x					
T4	x		x			x				
T5		x				x				
T6	x	x					x			
T7				x						
T8	x			x	x					x
T9			x						x	
T10	x				x					

Table 1: CALCULATING SEVERITY OF FAULTS

3) Test Case Weightage

Test case weight of ith test case is computed as follows.

*TCWi = RFTi * FIi .....(4)*

Test cases are sorted for execution based on the descending order of TCW, such that test case with highest TCW runs first.

4 ALGORITHM:

THE PROPOSED PRIORITIZATION TECHNIQUE IS PRESENTED IN AN ALGORITHMIC FORM HERE UNDER: THE INPUT OF THE ALGORITHM IS TEST SUITE T, TEST CASE WEIGHTAGE OF EACH TEST CASE IS COMPUTED USING THE EQUATION (4) AND THE OUTPUT OF THE ALGORITHM IS PRIORITIZED TEST CASE ORDER.

Algorithm:

4) Begin
5) Set T' empty
6) for each test case t ε T do
7) Calculate test case weightage as TCW = RFT * FI.
8) end for
9) Sort T in descending order on the value of test case weightage
10) Let T' be T
11) End

case fault	T1	T2	T3	T4	T5	T6	T7	T8	T9	T 10
F1								*	*	
F2		*	*		*					
F3				*		*				*
F4		*	*							
F5								*		
F6								*	*	
F7				*	*		*			
F8	*					*				
F9				*		*				*
F10	*							*		
faults	2	2	2	3	2	3	1	4	2	2
Time	9	8	14	9	12	14	11	10	10	13
Severity	6	6	6	10	8	10	4	20	12	6

Table 2: TEST CASES DETECTING NUMBER OF FAULTS

Rate of fault can be calculated using eqn. (1)

- RFT1 = (2/9)*10=2.22
- RFT2 = (2/8)*10=2.5
- RFT3= (2/14)*10 =1.428
- RFT4= (3/9)*10=3.33
- RFT5= (2/12)*10=1.66
- RFT6= (3/14)*10=2.142
- RFT7= (1/11)*10=0.9
- RFT8 = (4/10)*10=4.0
- RFT9 = (2/10)*10=2.0
- RFT10= (2/13)*10=1.538

From Equation (3) Fault impact of test cases T1, T2….T10 respectively.

- FI1 = (6/20)*10 = 3.0
- FI2 = (6/20)*10 = 3.0
- FI3 = (6/20)*10 = 3.0
- FI4 = (10/20)*10 = 5.0
- FI5 = (8/20)*10 = 4.0
- FI6 = (10/20)*10 = 5.0
- FI7 = (4/20)*10 = 2.0
- FI8 = (20/20)*10 = 10.0
- FI9 = (12/20)*10 = 6.0
- FI10 = (6/20)*10 = 3.0

From Eq. (4) test case weightage of test cases T1, T2, T10 respectively.

- TCW1 = 5.22
- TCW2 = 5.5
- TCW3 = 4.428
- TCW4 = 8.33
- TCW5= 5.66
- TCW6= 7.142
- TCW7= 2.9
- TCW8= 14.0
- TCW9 = 8.0
- TCW10= 4.538

Prioritize the test case according to decreasing order of their test case weightage (TCW), so the prioritized test case order is: T8, T4, T9, T6, T5, T2, T1, T10, T3, and T7.

Comparison between prioritized and non prioritized test case:

The comparison is drawn between prioritized and non prioritized test case, which shows that number of test cases needed to find out all faults are less in the case of prioritized test case compared to non prioritized test case the APFD can be computed according to equation (5).

$$APFD = 1 - TF1 + TF2 + \ldots\ldots + TFm / nm + 1/2n \quad \ldots\ldots (5)$$

where , m = the number of faults contained in the program under test P

n = the total number of test cases

TFi = The position of the first test in T that exposes fault i.

APFD for Prioritized test case:

$$APFD = 1 - 1 + 6 + 2 + 6 + 1 + 1 + 2 + 7 + 2 + 7/ 10*10 + 1/ 2* 10$$

$$APFD = 0.70$$

APFD for Non Prioritized test case:

$$APFD = 1 + 8 + 2 + 4 + 2 + 8 + 8 + 4 + 1 + 4 + 1 / 10* 10 + 1 / 2* 10$$

$$APFD = 0.63$$

Results indicate that the Average percentage of fault detected is better in case of prioritized test cases as compared to random ordering of test cases.
The results prove that the proposed prioritization technique is effective.

Lots of research has been done and techniques are developed on test case prioritization for regression testing. Wong et al proposed a hybrid technique which combines modification, minimization and prioritization based selection using source code changes and test history [3]. They also suggested that prioritization approach could be especially useful when testers can only afford to rerun a few regression tests.
Elbaum et al proposed an improved test case prioritization technique by incorporating varying test costs and fault severities [2].
Kim et al modeled regression testing as an ordered sequence of testing sessions and presented a history based prioritization technique which utilizes the information from previous testing [4].
Our proposed algorithm processes the number of faults and edges in individual independent paths of any software module calculated using control flow graph and gives priority to the set of paths. The cost and time required by this algorithm is lower as compared to the other techniques mentioned above. This proposal makes use of the control flow graph, so that it is hard to apply this technique when source code is not available.

5 CONCLUSION

In this paper, we proposed a general process of test case prioritization in regression testing. To implement it, we also propose an algorithm which prioritizes the test cases on the basis of the rate of fault detection and impact of fault.
Our results suggest that this technique can improve the rate of fault detection of test suites. The test case prioritization technique that we have generated can be described as “general prioritization techniques” in the sense that they are applied to a base version of a program, with no knowledge of the location of modifications to the software, in the hope of producing a test case ordering that will be effective over subsequent versions of the software.
Future work will be based on extension of this proposed algorithm and comparison of this technique with already

existing techniques and modifying our technique to a more cost and time efficient one.

6 References

[1] Marius Nita David Notkin "White-Box Approaches for Improved Testing and Analysis of Configurable Software Systems" IEEE 2009.

[2] Girish Janardhanudu"White Box Testing" in 2009.

[3] Gregory M. Kapfhammer "Software Testing" ACM 2008

[4] Jon Oltsik"Black Box Testing and Codenomicon DEFENSICS" in 2008.

[5] Hyunsook Do, Siavash Mirarab, Ladan Tahvildari "An Empirical Study of the Effect of Time Constraints on the Cost-Benefits of Regression Testing" ACM 2008.

[6] Goutam Kumar Saha "Understanding Software Testing Concepts" ACM 2008.

[7] R. Pressman, Software Engineering: A Practitioner's Approach. Boston: McGraw Hill, 2001.

[8] Bo Jiang, Zhenyu Zhang , W. K. Chan, T. H. Tse, "Adaptive Random Test Case Prioritization" IEEE/ACM International Conference on Automated Software Engineering, 2009 .

[9] Dongjiang You, Zhenyu Chen, Baowen Xu1, Bin Luo and Chen Zhang "An Empirical Study on the Effectiveness of Time-Aware Test Case Prioritization Techniques" , 2011.

[10] Ashwin G. Raiyani & Sheetal S. Pandya "Prioritization technique for minimizing number of test cases" International Journal of Software Engineering Research & Practices Vol.1, Issue 1, Jan, 2011.

[11] R. Kavitha & Dr. N. Sureshkumar "Test Case Prioritization for Regression Testing based on Severity of Fault" International Journal on Computer Science and Engineering Vol. 02, No. 05, 2010.

[12] R.Krishnamoorthi and S.A.Sahaaya Arul Mary "Regression Test Suite Prioritization using Genetic Algorithms" International Journal of Hybrid Information Technology Vol.2, No.3, July, 2009.

[13] Gregg Rothermel_ Roland H. Untchy Chengyun Chuz Mary Jean Harrold "Prioritizing Test Cases For Regression Testing" 2001.