Test Case Prioritization for Regression Testing based on Severity of Fault

R. Kavitha
Assistant Professor/CSE
Velammal College of Engineering and Technology
Madurai, Tamilnadu, India

Dr. N. Sureshkumar
Principal
Velammal College of Engineering and Technology
Madurai, Tamilnadu, India

Abstract— **Regression testing is one of the most critical activities of software development and maintenance. Whenever software is modified, a set of 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 don't affect the remaining software. It is impractical to re-execute every test case for a program if changes occur. The problem of regression test case selection can be solved by prioritizing the test cases. Regression test prioritization techniques reorder the execution of a test suit in an attempt to ensure that faults are revealed at the earlier stage of the testing process. Test case prioritization techniques schedule test cases for execution so that those with higher priority, according to some criterion are executed earlier than those with lower priority to meet some performance goal. In this paper an algorithm is proposed to prioritize test cases based on rate of fault detection and fault impact. 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.**

Keywords- Regression Testing, Test Case, Average Percentage of Faults Detected (APFD).

I. INTRODUCTION

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 [1]. Dennis Jeffrey and Neelam Gupta [2] have tested experimentally by selectively retaining test cases during test suite reduction. In [3], [4], they have empirically evaluated several test case filtering techniques that are based on exercising information flows. Call-Stack coverage technique is also used to reduce the test suite [5].

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 [6], [7], [8].

In [9], [10], Gregg Rothermel 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) Fault Impact.

To determine the effectiveness of proposed algorithm, we tested two projects developed in a CCSQ at Chennai. We analyze the test cases by feeding faults, invariant of the severity into the projects. Section 2 presents the literature survey on coverage based test case prioritization. Section 3 and 4 describes the new proposed prioritization technique and case study conducted. Section 5 presents the summary and future work.

II. RELATED WORK

This section describes the code coverage based TCP Strategies and their benefits. Coverage based TCP done their prioritization based on their coverage of statements [1]. 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) [1]. 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. Our 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 [3,],[4]. 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. In this paper, severity value also is considered as one of the factors to prioritize the test cases where severity value ranging from 2 to 10 to the faults.

III. PROPOSE PRIORITIZATION TECHNIQUE

This section discusses the proposed set of prioritization factors and the prioritization algorithm.

A. *Factors to be considered for prioritization*

Two factors are proposed for system level prioritization. These two factors are discussed below, and the reasoning of why they were chosen for prioritization technique and their importance of software testing.

- Rate of fault detection

The average number of faults 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.

$$RFTi = ((\text{number of faults}) / \text{time}) * 10 \qquad (1)$$

Every factor is converted into 1 to 10 point scale. The reason being, earlier work [3], [9] 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 presented in this paper implemented 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.

- Fault Impact

Testing efficiency can be improved by focusing on the test case that is likely to contain 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 given below.

Very High Severe	: SV of 10
High Severe	: SV of 8
Medium Severe	: SV of 6
Less Severe	: SV of 4
Least Severe	: SV of 2

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

$$S_i = \sum_{j=1}^{t} SV \qquad (2)$$

If Max(S) is the high severity value of test case among all the test cases then fault impact of i^{th} test case is shown below

$$FIi = \left(Si/Max(S)\right)*10 \qquad (3)$$

B. *Test Case Weightage*

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

$$TCWi = RFTi + FIi \qquad (4)$$

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

C. *Proposed Prioritization 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.

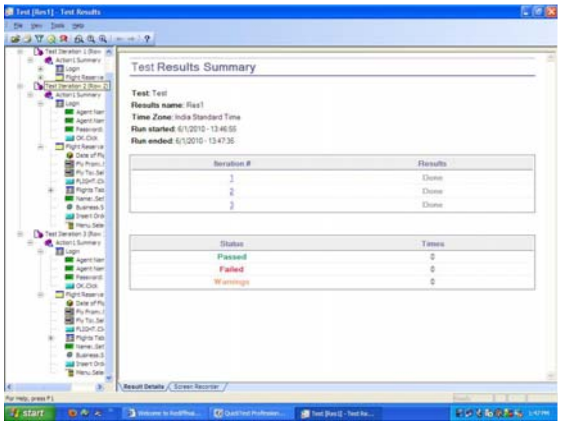
Algorithm:

```
1. Begin
2. Set T' empty
3. for each test case t ε T do
4. Calculate test case weightage as
           TCW = RFT + FI
5. end for
```

6. Sort T in descending order on the value of test case weightage
7. Let T' be T
8. end

IV. Experimentation and Analysis

The experiments were conducted on a PC with a 3GHz Intel Pentium 4 CPU and 8GB memory running the Windows XP operating systems. Two projects were tested using manual testing and testing tool QTP 9.5. The screen shot for the defect view is presented in Fig.1. We injected 10 faults, varying in severity level in each of the projects. To test the projects, we wrote 10 test cases for system level testing for each of the project. We have noted the time taken to find out the faults by each test case. The Table 1 shows the number of faults detected by each test case, the total time taken to detect the fault and severity values of faults for each test case.



Test Results Summary

Test: Test
Results name: Res1
Time Zone: India Standard Time
Run started: 6/1/2010 - 13:46:55
Run ended: 6/1/2010 - 13:47:36

Iteration #	Results
1	Done
2	Done
3	Done

Status	Times
Passed	0
Failed	0
Warnings	0

Figure 1. Defect view

Table I. Time taken to find out the fault and the severity value of first Project

Test case / Fault	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
F1								*	*	
F2		*	*		*					
F3				*		*				*
F4		*	*							
F5								*		
F6								*	*	
F7				*	*		*			
F8	*					*				
F9				*		*				*
F10	*							*		
Number of Faults	*2*	*2*	*2*	*3*	*2*	*3*	*1*	*4*	*2*	*2*
Time (ms)	*9*	*8*	*14*	*9*	*12*	*14*	*11*	*10*	*10*	*13*
Severity Value	*6*	*6*	*6*	*10*	*8*	*10*	*4*	*20*	*12*	*6*

From Proposed Technique Rate of fault detection of test cases $T_1, T_2 \dots T_{10}$ respectively.

$RFT_1 = (2/9)*10=2.22$
$RFT_2 = (2/8)*10=2.5$
$RFT_3= (2/14)*10 =1.428$
$RFT_4= (3/9)*10=3.33$
$RFT_5= (2/12)*10=1.66$
$RFT_6= (3/14)*10=2.142$
$RFT_7= (1/11)*10=0.9$
$RFT_8 = (4/10)*10=4.0$
$RFT_9 = (2/10)*10=2.0$
$RFT_{10}= (2/13)*10=1.538$

From Equation (3) Fault impact of test cases $T_1, T_2 \dots T_{10}$ 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 Equation (4) test case weightage of test cases $T_1, T_2 \dots T_{10}$ respectively.

$TCW_1 = 5.22$
$TCW_2 = 5.5$
$TCW_3 = 4.428$
$TCW_4 = 8.33$
$TCW_5= 5.66$
$TCW_6= 7.142$
$TCW_7= 2.9$
$TCW_8= 14.0$
$TCW_9 = 8.0$
$TCW_{10}= 4.538$

Prioritize the test case according to decreasing order of their test case weightage (TCW), so the prioritized test case order is: T_8, T_4, T_9, T_6, T_5, T_2, T_1, T_{10}, T_3, and T_7.

A. *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. It can be observed from Figure 2 that the new prioritization technique needs only 60% of test cases to find out all the faults. But 80% of test cases are needed to find out all the faults in the case of non prioritization, if test cases are executed in this order: T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, and T_{10}.APFD is the portion of area below the curve in Figure 2, plotting percentage of test cases executed against percentage of faults detected. Formally, the APFD can be computed according to equation (5).

$$ABFD = 1 - \frac{\sum_{k=1}^{m} Pos(Fk)}{nm} + \frac{1}{2n} \quad \textbf{(5)}$$

where n is the number of test cases, m is the number of revealed faults and pos(F_k) is the position of the first test case revealing the fault F_k in the prioritized test case.
APFD for Prioritized test case:

$$APFD = 1 - \frac{1+6+2+6+1+1+2+7+2+7}{10*10} + \frac{1}{2*10}$$

APFD = 0.70
APFD for Non Prioritized test case**:**

$$APFD = 1 - \frac{8+2+4+2+8|+8+4+1+4+1}{10*10} + \frac{1}{2*10}$$

APFD = 0.63

Ten random prioritization sets are generated to allow for comparison. And the results in Table 2 shows the new prioritization algorithm is better than randomized order. The results of the APFD for both prioritized and randomized order for the two projects are presented in graphical way in Fig. 2 and Fig. 3.

Project 1

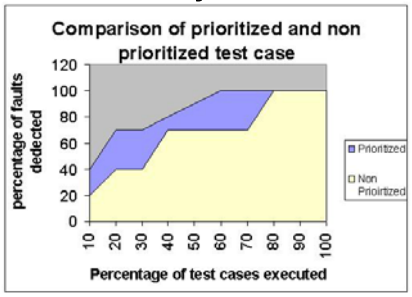


Figure 2. APFD is higher for prioritized test case order that reveal most faults early

Project 2

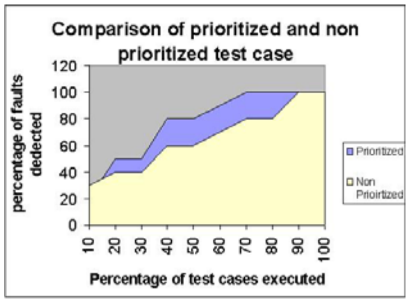


Figure 3. APFD is higher for prioritized test case order that reveal most faults early

Table II. Percentage of test cases executed to detect all the faults for prioritized and random order for Project 1

Random Test Case order	% of test case executed to detect all the faults
7,10,5,7,3,1,6,4,9,8	100
3,10,8,6,9,1,4,7,2,5	70
1,5,3,7,10,6,2,9,4,8	100
1,4,6,5,10,2,3,7,8,9	90
9,1,10,2,5,7,4,6,3,8	100
5,1,4,10,8,9,2,3,7,6	70
10,7,5,4,3,2,9,8,6,1	90
8,1,3,2,5,9,10,4,6,7	70
3,6,4,9,10,7,8,1,5,2	70
2,10,5,3,7,6,9,8,4,1	80
3,2,10,7,1,9,6,5,8,4	90
Prioritized Order	60

V. CONCLUSION

In this paper a new prioritization technique to improve the rate of fault detection of severe faults for Regression testing is proposed. Here, two factors rate of fault detection and fault impact for prioritizing test cases are proposed. The proposed algorithm is validated by analyzing two sets of industrial projects. Results indicate that the proposed technique lead to improved rate of detection of severe faults in comparison to random ordering of test cases. And also it is tested experimentally that the number of test cases runs to find the entire fault is less in case of proposed prioritization technique. The results prove that the proposed prioritization technique is effective. In future, test case prioritization over requirement analysis will be tried

APPENDIX A

In order to validate the effectiveness of the proposed technique, two VB projects of approximately 4500 LOC tested in CCSQ, Competency Centre for Software Quality, Chennai, India were chosen. Programs were thoroughly tested by manual testing and using QTP tool. The proposed prioritization algorithm was analyzed by seeding faults, invariant of the severity. On the entire faulty programs prioritized test cases are run and the execution of total number of test cases to find the faults are computed. Then ten different random orders of test cases using random number generation in 'C' are generated. The test cases are executed in this different random order of test cases and the total numbers of test cases run to find out all the faults are detected. The results of number of test cases executed to detect all faults in prioritized and non prioritized test cases are compared to show the effectiveness of the proposed test case prioritization. And Also Average Percentage of Fault Detected is higher for prioritized test cases.

REFERENCES

[1] Zheng Li, Mark Harman, and Robert M. Hierons, "Search algorithm for Regression Test Case Prioritization," IEEE Transactions on Software Engineering, Vol. 33, No.4, April 2007.

[2] Dennis Jeffrey and Neelam Gupta, "Improving Fault Detection Capability by Selectively Retaining Test Cases during Test Suite Reduction," IEEE Transactions on software Engineering, VOL. 33 NO.2, February 2007.

[3] Jennifer Black, Emanuel Melachrinoudis and David Kaeli, "Bi Criteria Models for All uses Test Suite-Reduction," 26th International Conference on Software Engineering (ICSE'04).

[4] Wes Masri, Andy Podgurski and David Leon, "An Emprical Studey of Test Case Filtering Techniques Based on Exercising Information Flows," IEEE Transactions on software Engineering, VOL. 33, NO.7, February 2007.

[5] Scott McMaster, Atif M. Memon, "Call Stack Coverage for GUI Test Suite Rdduction," IEEE Transactions on software Engineering, VOL. 34 NO.1, January/February 2008.

[6] J. Karlsson, K. Rayan , "A Cost value approach for Prioritizing requirements," IEEE Software Vol 14, NO 5, 1997.

[7] Maruan Khoury, "Cost Effective Regression Testing," October 5, 2006.

[8] Alexey G. Malishevsy, Gregg Rothermel, Sebastian Elbaum,"Modeling the Cost-Benefits Tradeoffs for Regression Testing Techniques," Proceedings of the International Conference on Software Maintenance ICSM'02), 2002 IEEE.

[9] Sebastian Elbaum, Alexey G. Malishevsky and Gregg Rothermel, "Test Case Prioritization: A Family of Emprical Studies," IEEE Transactions on software Engineering, VOL. 28, NO.2, February 2002.

[10] Gregg Rothermel, Roland H. Untch, Chentun Chu and Mary Jean Harrold, "Prioritizing Test Cases for Regression Testing," IEEE Transactions on software Engineering, VOL. 27 NO.10, October 2001.