Test Case Prioritization in Regression Testing using Various Metrics

Seema Sharma
M.Tech. Scholar, Dept. of CSE
SGI, Gurgaon.

Dr. Preeti Gera
Associate Professor, Dept. of CSE
SGI, Gurgaon, India.

Abstract—Regression testing is one of the most critical activitiesof software development and maintenance. Whenever software ismodified, a set of test cases are run and the comparison of newoutputs is done with the older one to avoid unwanted changes. Ifthe new output and old output match it implies that themodifications made in one part of the software don't affect theremaining software. 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. The problem of regression testcase selection can be solved by prioritizing the test cases.Regression test prioritization techniques reorder the execution ofa test suit in an attempt to ensure that faults are revealed at theearlier stage of the testing process. Test case prioritizationtechniques schedule test cases for execution so that those withhigher priority, according to some criterion are executed earlierthan those with lower priority to meet some performance goal. Inthis paper an algorithm is proposed to prioritize test cases basedon rate of fault detection and impact of fault. The proposedalgorithm identifies the severe fault at earlier stage of the testing process and the effectiveness of prioritized test case andcomparison of it with unprioritized ones with the help of APFD is done here in this paper.

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 [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.

II. EARLIER 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) .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 impact of fault value also has to be considered to prioritize the test cases.

III. MAIN RESULTS

This section discusses the proposed set of prioritization factors and the prioritization algorithm.
Factors to be considered for prioritization

(1) Rate of fault detection:

The average number of faults detected perminute 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.

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

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 ts identified for a module.

$$TSFD = \sum_{i=1}^{i=n} SM \text{ (severity measure)} \quad (2)$$

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

	Assign Severity Value									
Fault detected for some 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					

According to the equation 2 the severity value of ea each test case can be calculated as follows:

T1 = 6

T2 = 6

T3 = 6

T4 = 10

T5 = 8

T6 = 10

T7 = 4

T8 = 20

T9 = 12

T10 = 6

Equation (3) shows that the severity value of test case i, where t represent number of faults identified by the ith test case.

$$S_t = \sum_{j=1}^{t} SV \qquad (3)$$

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:

$$FIi = (Si / Maximum(S)) * 10 \qquad (4)$$

(3) Customer Assigned priority (CP):

It is a measure of the importance of customer requirement. So, the requirement with highest customer importance should be tested early to improve customer satisfaction. The customer assigns the values for each requirement ranging from 1 to 10 where 10 denote highest customer priority.

(4) Test Case Weightage

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

$$TCWi = RFTi * FIi + CP \qquad (5)$$

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

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+CP.

5. end for

6. Sort T in descending order on the value of test case weightage

7. Let T' be T

8. end

Test Case / Fault	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
F1								*	*	
F2		*	*		*					
F3				*		*				*
F4		*	*							
F5								*		
F6								*	*	
F7				*	*		*			
F8	*					*				
F9				*		*				*
F10	*							*		
No. 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
Customer Assigned priority (CP)	5	5	4	9	3	8	1	10	3	2

Table : Time taken to detect faults & fault severity& customer assigned priority

From Proposed Technique Rate of fault detection of test cases T1, T2….T10 respectively

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 Equation (4) test case weightage of test cases T1, T2….T10 respectively.

TCW1 = 11.66

TCW2 = 12.5

TCW3 = 8.284

TCW4 = 25.65

TCW5= 9.64

TCW6= 18.71

TCW7= 2.8

TCW8= 50

TCW9 = 15

TCW10= 6.614

Prioritize the test case according to decreasing order of their test case weightage (TCW), so the prioritized test case order is: T8, T4, T6, T9, T2, T1, T5, T3, T10, 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. Formally, the APFD can be computed according to equation (5).

APFD = 1 – (TF1 + TF2 + ……+ TFm)/nm + 1/2n …… (5)

where , 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 for Prioritized test case:

APFD = 1 – (1 + 5 + 2 + 2+ 1 + 1 + 2 + 6+ 2 + 9)/ 10*10 + 1/ 2* 10

APFD = 0.74

APFD for Non Prioritized test case:

APFD = 1-(1 + 8 + 2 + 4 + 2 + 8 + 8 + 4 + 1 + 4 + 1)/ 10* 10 + 1 / 2* 10

APFD = 0.63

IV.CONCLUSION

Thus a new prioritization technique to improve the rate of fault detection of severe faults for Regression testing is used. Here, two factors rate of fault detection and fault impact for prioritizing test cases are used. 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.

ACKNOWLEDGMENT

I would like to convey the sincerity and magnitude of my gratitude to my guide Dr. Preeti Gera, to whom I am really indebted, for her influence in my studies and life & for her inspiration, suggestion and cooperation during my research days. I have no hesitation to confess that she has enlightened a ray of hope in my heart for propelling towards my goal. Her brilliant guidance catalysed my research work and inspired me to face every difficult and challenging aspects pertaining to my studies. Her dedication towards my work is inexplicable. I shall follow her ideology for the rest of my life.

REFERENCES:

[1] Elbaum, S., Malishvesky, A.G., Rothermel, G., 2002. Test case prioritization: a family of empirical studies. IEEE Transactions on Software Engineering 28 (2), 159–182.
[2] 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.
[3] A. G. Malishevsky, G. Rothermel, and S. Elbaum. Modeling the cost-benefits tradeoffs for regression testing techniques. In proceedings of the International Conference on Software Maintenance, pages 204–213, Montreal, Quebec, Canada, October 2002.
[4] Alexey G. Malishevsky, Joseph R. Ruthruff, Gregg Rothermel, Sebastian Elbaum, Cost cognizant Test Case Prioritization, 2006
[5] 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.
[6] Amrita Jyoti, Yogesh Kumar Sharma, Ashish Bagla, D. Pandey “RECENT PRIORITY ALGORITHM IN REGRESSION TESTING” International Journal of Information Technology and Knowledge Management in July-December 2010, Volume 2.
[7] 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

[8] Bo Jiang & W. K. Chan "On the Integration of Test Adequacy, Test Case Prioritization, and Statistical Fault Localization", 2010.
[9] Bo Jiang, Zhenyu Zhang , W. K. Chan, T. H. Tse, "Adaptive Random Test Case Prioritization" IEEE/ACM International Conference on Automated Software Engineering,2009 .
[10] MaruanKhoury, "Cost Effective Regression Testing," October 5,2006.
[11] Dennis Jeffrey and Neelam Gupta, "Improving Fault Detection Capability by Selectively Retaining TestCases during Test Suite Reduction," IEEE Transactions on software Engineering, VOL. NO.2, February 2007.
[12] 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.
[13] G. Antoniol, M.D. Penta, and M. Harman, "Search-Based Techniques Applied to Optimization of Project Planning for a Massive Maintenance Project," Proc. 21st IEEE Int'l Conf. Software Maintenance (ICSM '05), pp. 240-249, 2005