ISSN 22773061

STUDY OF TEST CASE PRIORITIZATION TECHNIQUE USING APFD

Rahul Gupta, Akhilesh Kumar Yadav
E-Mail-rahulgupta468@gmail.com
Computer Science and Engineering
Kanpur Institute Of Technology, Kanpur

Abstract—

Regression testing is used to ensure that bugs are fixed and new functionality introduce in a new version of a software that don't adversely affect the original functionality inherited from the previous version.Regression testing is one of the most complaining activities of software development and maintenance.Unluckily, It may have feeble resources to allow for the re-execution of all test cases during regression testing. In this situation the use of test case prioritization is profitable because the best appropriate test cases are executed first. In this paper we are proposing an algorithm to prioritize test cases based on rate of fault detection and impact of fault.The proposed algorithm recognises the exhausting fault at earlier stage of the testing process.We are using an Average Percentage of Faults Detected (APFD) metric to determine the effectiveness of the new test case arrangements.

Keywords—Test case; regression testing; test case prioritization; fault impact; APFD(Average Percent of Fault Detected); fault detection

Council for Innovative Research

Peer Review Research Publishing System

Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY

Vol 10, No 3

editor@cirworld.com
www.cirworld.com, member.cirworld.com

INTRODUCTION

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

Test suits are often carry through by Software developers, so that they can reuse them, when software undergoes changes. Applying all test cases, an subsist test suite can consume massive amount of time. Lets an example product that contains approximately 50,000 lines of code running an entire test suits requires several weeks. The researchers have found various algorithms to reduce the cost of regression testing and also to increase the effectiveness of testing [9] Dennis Jeffrey and Neelam Gupta [DEN2007] have tested experimentally by selectively retaining test cases during test suite reduction.

In [JEN2004], they have through empirical observations evaluated several test case filtering techniques. These test cases are based on exercising information flows. Another way of testing is to maintain an order the test cases based on some criteria & heuristic to meet some performance goal. Testers may maintain the test cases according to some method so that the appropriate test cases run first. Here test case prioritization technique does not discard test cases. And they can avoid the drawback of test case minimization techniques. The software is successful, cost should be minimized and the product should be delivered to the customer in time.

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 it also has the different testing criteria which are useful for identifying test cases that exercise different structural and functional elements in a program. 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, we are proposing one new approach to prioritize the test cases at system level for *regression* test cases. This technique is more effective than earlier as it identifies more severe faults at an earlier stage of the testing process.

Following factors proposed to design algorithm:

1) Impact of Fault. Analyze the test cases by nourishing faults, invariant of the severity into any program.

2) Fault detection rate There are 2 assumptions on which The APFD (Average Percent of Fault Detected) metric relies:

(1) All faults have equal costs (hereafter referred to as fault severities),

(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. But in practice, however, there are cases in which these assumptions don't hold: cases in which faults vary in severity and test cases vary in cost. In such cases, the APFD metric can provide unsatisfactory results.

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 regression testing 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.

In the paper [9] (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, percentage of decision covered and percentage of statement covered .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 weight age 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.

ISSN 22773061

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.

PROPOSED 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) Fault detection rate:

The average number of faults detected per unit time by a test case is called fault detection rate. The fault detection rate 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:

Efficiency of testing 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 (ascending) 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

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

Table 1: CALCULATING SEVERITY OF FAULTS

ISSN 22773061

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

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

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

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

1. T1 = 7
2. T2 = 8
3. T3 = 2
4. T4 = 10
5. T5 = 5
6. T6 = 10
7. T7 = 4
8. T8 = 20
9. T9 = 10
10. T10 = 15

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

IFi = (TSFD/ Maximum (TSFD)) * 10..... (3)

3) Weightage of Test Cases

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

WTCi = RFTi * IFi .....(4)

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

Algorithm:

case fault	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
F1		*	*	*	*				*	
F2				*	*	*				
F3							*		*	
F4						*	*	*		*
F5									*	
F6					*	*			*	
F7								*		
F8										*
F9	*	*	*	*		*				
F10					*		*			
Total faults	1	2	2	3	4	4	3	2	4	2
Time	8	9	16	8	12	15	10	11	12	16
Severity	7	8	2	10	5	10	4	20	10	15

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 thee algorithm is prioritized test case order.

1) *Begin*
2) *Set T' empty*
3) *for each test case t ε T do Calculate test case weightage asWTC=RFT*IF.*
4) *end FOR*
5) *Sort T in descending order on the value of test case weightage.*
6) *Let T' be T*
7) *End*

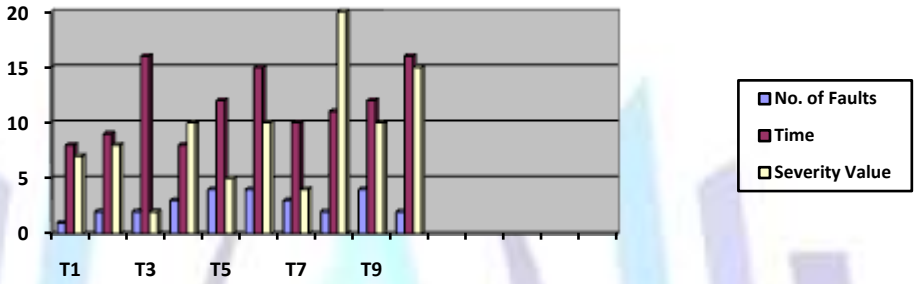

Chart 1: Showing the total time, no. of faults & severity measure values

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

- RFT1 = (1/8)*10=1.25
- RFT2 = (2/9)*10=2.22
- RFT3= (2/16)*10 =1.25
- RFT4= (3/8)*10=3.75
- RFT5= (4/12)*10=3.33
- RFT6= (4/15)*10=2.66
- RFT7= (3/10)*10=3.00
- RFT8 = (2/11)*10=1.81
- RFT9 = (4/12)*10=3.33
- RFT10= (2/16)*10=1.25

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

- IF1 = (7/20)*10 = 3.50
- IF2 = (80/20)*10 = 4.00
- IF3 = (20/20)*10 = 1.00
- IF4 = (100/20)*10 = 5.00
- IF5 = (50/20)*10 = 2.50
- IF6 = (100/20)*10 = 5.00
- IF7 = (40/20)*10 = 2.00
- IF8 = (200/20)*10 = 10.00
- IF9 = (100/20)*10 = 5.00
- IF10 = (100/20)*10 = 7.50

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

- WTC1 = 4.375
- WTC2 = 8.880
- WTC3 = 1.250
- WTC4 = 18.750
- WTC5= 8.325
- WTC6= 13.300
- WTC7= 6.000
- WTC8= 18.100
- WTC9 = 16.650
- WTC10= 9.375

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

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 + ……+ 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

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

APFD for Non Prioritized test case:

APFD = 1 - 2 + 4 + 7 + 6 + 9 + 5 + 8 + 10 + 1 + 5 / 10* 10 + 1 / 2* 10

APFD = 0.38

APFD for Prioritized test case

APFD = 1 – 6 + 1 + 8 + 4 + 3 + 7 + 2 + 5 + 9 + 8/ 10*10 + 1/ 2* 10

APFD = 0.42

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.

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.

The 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 the technique to a more cost and time efficient one.

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.