A NOVEL APPROACH FOR TEST CASE PRIORITIZATION

Sahil Gupta[1], Himanshi Raperia[2], Eshan Kapur[3], Harshpreet Singh[4] and Aseem Kumar[5]

Department of Computer Engineering, LPU, Jalandhar, India
sahil.gupta2688@gmail.com[1]
himanshi19.rpr@gmail.com[2]
eshankapur@gmail.com[3]
harshpreet.15790@lpu.co.in[4]
aseem.kumar6@gmail.com[5]

ABSTRACT

Test case prioritization techniques basically schedule the execution of test cases in a definite order such that to attain an objective function with greater efficiency. This scheduling of test cases improves the results of regression testing. Test case prioritization techniques order the test cases such that the most important ones are executed first encountering the faults first and thus makes the testing effective. In this paper an approach is presented which calculates the product of statement coverage and function calls. The results illustrate the effectiveness of formula computed with the help of APFD metric.

KEYWORDS

Test case prioritization, test cases, faults, APFD

1. INTRODUCTION

Software testing is an activity to assure the stakeholders and provide them with the information about the product quality or the service under test. Software developers often save the test suites they develop for their software, so that they can reuse those suites later as the software evolves [2]. The main aim of software testing is the detection of software failures so that they could be corrected after their discovery. It is not a trivial approach. The major establishment of testing is that a software product does not functions up to the mark or properly under specific conditions and not that it functions properly under all conditions. In software development life cycle test cases can be scheduled according to the priority using test case prioritization techniques so that ones with the higher priority can be executed earlier in the testing process. This improves the rate of fault detection in codes so that software engineers can address the faults earlier thereby reducing the development time. The main aim and scope of this is that by prioritizing the test cases in some definite order based on some factors should lead to decrement in the regression testing cost may be in terms of testing time or fault detection [6]. The advantage of the techniques is that they don't discard tests as they do in the test case reduction and non-safe regression test selection technique along with improvement in testing performance and earlier feedback on the system under test. It is suggested by these results that the relative cost-effectiveness of these various techniques for prioritization also varies across workloads. Workloads can be test suites, programs and different types of modifications.

In this paper in section 2 all the parameters are discussed and the benefits of new approach are described along with the use of prioritizing the test cases. In section 3 the proposed technique is

DOI : 10.5121/ijcsea.2012.2305

described along with all the inclusions i.e. the factors and the metrics used in the compilation. The algorithm is defined which illustrates the flow. In section 4 the results are analysed and calculated which were deduced. Further in section 5 the paper is concluded with an advantage of the technique.

2. TEST CASE PRIORITIZATION

Test case prioritization techniques are employed on the test cases in order to detect the defects in the software product and then remove them. The faults thus detected are a threat to the integrity of the software and they may hinder in the timely delivery of the finished product to the client. Software is test by executing various test cases under different conditions and environments. But the approach that all test cases are executed for similar modules changed or unchanged, it results in redundancy. Test case prioritization techniques schedule over test cases those results in increase in the performance of regression testing [3]. It is inefficient to repetitively execute every test case for every program function. So, test cases are scheduled in a queue for execution on the basis of priority depending upon different criterions and parameters.

Test case prioritization is important as it increases the efficiency of software testing as compared to the execution of test cases in a non-prioritized order. There are different approaches to prioritize test cases like based on statement coverage, number of lines covered, function coverage, optimal techniques, fault index, fault exposing potential, etc [5]. These can be implemented with the help of rate of fault detection like average percentage of faults detected per minute, fault impact, fault proneness of a module to be tested, testing impact, etc. As it is a very wide and broader topic we can pick any aspect of software testing used in test case prioritization such that by any chance the rate of fault detection can be increased and the cases are tested faster than previous methods employed. The proposed work comprises of an approach that prioritize test cases by increase in fault detection rate with the help of different parameters and metrics. Here as we work upon the analysis of different techniques and approaches for test case prioritization, it can be said that still there may be many different ways to prioritize test cases on various factors and parameters. But sometimes those techniques fail to give the efficient result due to some anomalies i.e. sometimes the effectiveness is not much in case where there are a number of test suites [4].

3. PROPOSED WORK: A NEW PRIORITIZATION APPROACH

3.1. Introduction

In the past it was very time consuming to run the test cases depending upon the size of the test suites. But now the test cases can be re ordered to find the increased rate of fault detection. The technique presented prioritizes the test cases in an order to maximize the number of faults. We introduce two criterions on the basis of which we determine a factor which is used to prioritize the test cases. After that the APFD metric is evaluated such that it is increased as compared to the non – prioritized order [1]. As the test case prioritization is done to improve the performance of regression testing so first time when the test cases are executed they are non – prioritized.

The requisites are that the number of code lines covered or statements covered should be known after the first test suite execution. In addition to that the number of function calls in the covered statements for respective test case should also be counted. The bind of the two mathematical

variables will yield to a metric that is in the form of a numeral. The value of this numeral will determine the order of the test cases. The test case which has the highest value will be executed first and then followed by the rest in the descending order. The importance of binding two variables is that it focuses two different criteria such that the result yielded is much more effective and efficient. The prioritization technique is dependent upon two factors so it becomes more complex.

3.2. Prioritization Factors

Here the objective is to define a new metric that form the basis of the priority order according to which test cases in a test suite are scheduled. It is stated that a new metric named product symbolized as P is proposed. It comprises of two varying factors for a test case. These are statement coverage which is symbolized as St and the number of function calls which is symbolized as Fc. When the product P is calculated

3.2.1. Statement Coverage

Statement coverage (St) is calculated for each test case that is executed in a test suite. It is a degree to which a program is tested. It is based on just the count of the number of statements traversed by the particular test case. Mathematically it is calculated as the number of statements covered upon the total number of statements per cent.

Statement coverage (St) = (No. of statements covered / total no. of statements) * 100

3.2.2. Function Calls

No. of Function calls (Fc) are calculated for each test case as it is looked into statements covered by the particular test case in a test suite. This variable is also a numeral as it includes the calls to a routine, procedure, method, function or subprogram. It is sum of both built in functions as well as user defined functions.

No. of Function calls (Fc) = Local functions + nested functions + private functions + overloaded methods [7]

Now we define a new metric that is the result of the integration of the above two variables that is the St and Fc. We multiply the above two to obtain a product for each test case. This can be denoted by P.

Product (P) = Statement coverage (St) * No. of Function calls (Fc)

After the test suite is executed first time for the software then it yields a result that when it is non – prioritized. Now as we will be able to keep an account for the re-test that is regression test that what is the statement coverage for each test case and how many function calls are occurring in the particular statement coverage for the particular test case.

As described above we will be calculating the product of the statement coverage and the no. of function calls. This becomes the software metric or the basis of the ordered set of test cases. As each test case has a value now so they will be ordered as the one with greatest value as the highest prioritized test case and followed in descending order.

For this an algorithm is proposed with the assumption that the statement coverage and no. of function calls are known, such that this is given as the input to the algorithm and the output is the prioritized test suite.

3.3. The Algorithm

Now we introduce a standard algorithm foe test case prioritization which is framed in a set pattern but within it only some changes in the calculation of the metric is done. The algorithm calculates the Product P metric for every test case given that the statement coverage St and Number of function calls Fc is known for every test case in advance. Then any sorting algorithm can be implemented to sort the product P values in descending order. These can be quick sort, merge sort or heap sort.

Input to the algorithm:

Test suite T, Statement coverage St, No. of function calls Fc

Output of the algorithm:

Prioritized test suite T'

1. **begin**
2. set T' empty
3. **for each** test case t € T **do**
4. calculate Product P as St * Fc
5. **end for**
6. sort T in descending order based on the value of P for each test case
7. let T' be T
8. **end** [1]

Now the test cases are arranged in descending order according to the value of P. But there can be a possibility that the product for the two or more test cases comes out to be same.

3.4. Possible Ambiguities and Solution

Then there is an ambiguity which test case would be executed first out of the two or which will be given a higher priority. Now here also this can be possible that is problem can be encountered in the two following ways:

3.4.1. The product P can be same of two or more test cases when the multiplication of both yields out to be same and their measure of the two variables is different respectively.

This can be made clearer with an example:

Variable / Test case	Statement coverage St (in percentage)	No. of function callsFc (in units)	Product P (in %units)
Test case T_2	10	4	40
Test case T_5	20	2	40

Here both the test cases yield the same product though both have different values of the variables St and Fc that integrate to give the value of the desired metric P.

3.4.2. The product P can also be same for two or more test cases when the multiplication of both yields out to be same but their measure of the two variables is same respectively.

This can be made clearer with an example:

Variable / Test case	Statement coverage St (in percentage)	No. of function calls Fc (in units)	Product P (in %units)
Test case T_2	10	4	40
Test case T_5	10	4	40

Here both the test cases yield the same product though both have same values of the variables St and Fc that integrate to give the value of the desired metric P.

Now these problems or ambiguity has to be resolved. If the first problem is encountered then we need to look into the no. of function calls (Fc) variable. Here if the product P comes out to be same for two or more test cases then their respective number of function calls is evaluated. The test case with the largest no. of function calls Fc is executed among them followed by the one with lesser Fc and so on. The Fc is given higher preference over St because there might be a chance that though the statement coverage by a test case may be more but it may not cover as many function calls. The more error prone or fault prone areas are where there is the involvement of as many function calls because the passing of arguments, data retrieval, etc is concentrated at the point of calling. So, it becomes more meaningful to consider the function calls. If we consider statement coverage with higher preference then it may yield worthless as statement coverage includes simple declarations, Print statements, etc also and those may result as a mask for error detection.

If the second problem encounters that is the Product P comes out to be same for two or more test cases and both the variables are same for the test cases then the test case which is encountered first will be executed first. This means this ambiguity is resolved by applying First Cum First Serve (FCFS) basis.

4. EXPERIMENTATION AND ANALYSIS

Now after we run the test suite for the first time then it is non – prioritized but after that the statement coverage for each test case is noted. Then for the respective statement coverage the no. of function calls are counted. On the basis of the two the test cases are given a priority in the test suite for regression test.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
St	40	25	30	30	26	40	45	25	30	35
Fc	5	8	3	8	5	4	4	4	5	4
P	200	200	90	240	130	160	180	100	150	140

Now the test cases map the faults that are exposed by them which are a pre requisite after the first test suite execution.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
F1						●			●	
F2		●						●		
F3							●			●
F4				●					●	
F5		●				●				
F6	●									
F7	●							●		
F8							●		●	
F9				●						●
F10		●					●			

Here comparison among the results of prioritised and non-prioritised 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 formula to calculate this is defined and illustrated below.

The prioritized order according to Fc 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

By using the APFD (average percentage of faults detected) formula:

$$APFD = 1- \{(TF_1 + TF_2 + TF_3 \ldots\ldots TF_m) / mn \} + (1/2n) \quad [1]$$

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**

Thus the prioritized test cases yield better fault detection than the non – prioritized test cases.

5. CONCLUSION

This paper proposed an approach for test case prioritization in order to improve regression testing. 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 and risk metrics.

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] Praveen Ranjan Srivastava, (2008) “Test Case Prioritization”, Journal of Theoretical and Applied Information Technology IEEE.
[2] Gregg Rothermel, Roland H. Untch, Chengyun Chu, Mary Jean Harrold, (1999) “Test Case Prioritization: An Empirical Study”, International Conference.
[3] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, (2001) “Prioritizing Test Cases for Regression Testing”, IEEE Transactions on Software Engineering.
[4] Siripong roongruangsuwan, Jirapun daengdej, (2010) “Test case prioritization techniques”, Journal of Theoretical and Applied Information Technology, IEEE.
[5] Sebastian Elbaum, Alexey G. Malishevsky and Gregg Rothermel, (2002) “Test case prioritization: A family of empirical studies,” IEEE Transactions on Software Engineering, Vol. 28, No.2, pp159-182.
[6] Gaurav Duggal, Mrs. Bharti Suri , “Understanding regression testing techniques”, Guru Gobind Singh Indraprastha University, Delhi, India.
[7] http://www.mathworks.in/help/techdoc/matlab_prog/f7-58170.html on 4.04.2012

Authors Biography

Mr. Sahil Gupta
M.Tech
Department of Computer Science
Lovely Professional University
Phagwara, Punjab

Ms. Himanshi Raperia
Asst. Prof.
Department of Computer Science
Lovely Professional University
Phagwara, Punjab

Mr.Eshan Kapur
M.Tech
Department of Computer Science
Lovely Professional University
Phagwara, Punjab

Mr. Harshpreet Singh
Asst. Prof
Department of Computer Science
Lovely Professional University
Phagwara, Punjab

Mr. Aseem Kumar
M.Tech
Department of Computer Science
Lovely Professional University
Phagwara, Punjab