ISSN: 2321-9653

INTERNATIONAL JOURNAL FOR RESEARCH IN APPLIED SCIENCE AND ENGINEERING TECHNOLOGY (IJRASET)

Regression Testing with Multiple criteria based Test Case Prioritization

Navleen Kaur[1], Manish Mahajan[2]

[1]Assistant Professor, Chandigarh Engineering College, Landran.
[2]Associate Professor, Chandigarh Engineering College, Landran.

Abstract*—*Test Case Prioritization is rearranging the test case order based on certain constraints so that the most beneficial test cases may be executed first. This is inefficient to re execute all the test cases in regression testing following the software modifications. Using information obtained from previous test case execution, prioritization techniques order the test cases for regression testing so that most beneficial are executed first thus allows an improved effectiveness of testing. There are different varieties of techniques for prioritization in the literature, we have basically used coverage-based prioritization techniques. A prioritized test suite which covers more than one coverage criteria is considered to be stronger than those which cover only single coverage. In In the empirical study, test cases are prioritized based on statement coverage, fault coverage and path coverage using the proposed method, so that the single regression process is performed with multiple coverage criterion. The proposed method was empirically studied for bank application and the results show that the proposed work is more effective than the existing method. The main aim of my paper is to determine the effectiveness of prioritized and non-prioritized case with the help of APFD, APSD, APBD and APPD.

Keywords* —*Test Cases, Prioritization, RegressionTesting, Software Testing, Multiple coverage criterion, Test Case Prioritization.

I. INTRODUCTION

Software systems are maintained by the developers by doing regression testing periodically in hope to find errors caused by changes and provide confidence that modifications made in the software are correct. Developers generally create an initial test suite and then reuse it for regression testing [1]. Software testing is a set of activities conducted with the intent of finding errors in software. It also verifies and validate whether the program is working correctly with no bugs or not. [2] The main purpose of software testing is to achieve quality assurance, verification, validation, and reliability estimation. Software Testing is done by executing a program or system repeatedly and then finding the errors in it. But, in practical, even after completion of the testing phase, it is not possible to guarantee that the final program is error free. This is because the input data domainof most programs is very large, and it is not practical to test the program exhaustively with respect to each value that the input can assume. But even with this limitation of testing, we should not underestimate the importance of testing.

We know that software testing can portray most of the defects that are in a program and therefore software testing provides a useful way of reducing defects in a system [3].

II. DEVELOPMENT OF TEST CASES AND PRIORITIZATION

ISSN: 2321-9653

Test case is a set of conditions or variables under which a tester will determine if a requirement upon an application is partially or fully satisfied. It may take many test cases to determine that a requirement is fully satisfied. In order to fully test that all the requirements of an application are met, there must be at least one test case for each requirement unless a requirement has sub requirements. In that situation, each sub requirement must have at least one test case.Test cases include a description of the functionality to be tested taken from either the requirements or use cases, and the preparation required to ensure that the test can be conducted. [4]

The basic objective of writing test cases is to validate the testing coverage of the application.Test cases should be written after understanding the function or technology that is to be tested, and each test case should be submitted for peer review. Test case is designed based on random input data.The process of developing test cases can also help find problems in the requirements or design of an application.

Regression test suits are often simply test cases that software engineers have previously developed, and that have been saved so that they can be used later to perform regression testing [5].Re executing all the test cases require enormous amount of time thus make the testing process inefficient.
Prioritizing test cases provide the opportunity tomaximize some performance goals or effectiveness. One of the performance goals may be rate of dependency detection among faults[6].

When designing the test cases, we should keep in mind that it is not possible to test everything. As a substitutefor trying to test every combination, we prioritize our test cases so that we can perform the most important tests — those that focus on areas that present the greatest risk or have the greatest probability of occurring.

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.

Test case prioritization is rearranging the test case order based on certain constraints so thatthe most beneficial test cases may be executed first.Test case prioritization techniqueimproves regression testing by ordering test cases such that the more important test case runs earlier in the testing process [7]. This is inefficient to re execute all the test cases in regression testing following the software modifications [8].
Effective test case prioritization technique for regression testing is necessary to ensure optimum utility and no side effect in the software after modification [9]. In this work, a multiple criterion based merging technique for test case prioritization method is proposed. The main objective of this work is to conduct regression test with minimum number of test cases to cover more than one coverage criterion such as fault coverage, statement coverage, path coverage and branch coverage.

Most of the existing research works on test case prioritization methods are based on single coverage criterion which is time consuming and more expensive [10]. A prioritized test suite which coversmore than one coverage criteria is considered to be astronger coverage goal than a test suite which covers single coverage criteria [11].

III. PROPOSED WORK

The main focus of the proposed work is to provide solution for software testing while testing efficiency of the software component. In our research we have focused on improving test case prioritization by adding multiple coverage criteria. The selection of test cases for testing the software is done by decreasing the overhead of regression testing. Bank application (BA) based on Java language and C++ language has been implemented according to optimized proposed work for software testing. We have tried topresent more testing dependencies. In our regression testing, we have added branch coverage, statementcoverage, path coverage and fault coverage dependencies test which are supportive in discovering issues in completephase of testing. Software testing has earlier been done by various tools and we have related to proposed

theory. Test case prioritization hasbeen applied to the software testing work of Industrial applications and is compared with traditional method.

IV. IMPLEMENTATION

Various steps which have been used for implementing are as follows:-

1. Extensive literature survey is done to understand different testing techniques which are available.
2. Software testing data is fetched from a program based on bank applications in JAVA and C++ language.
3. Test cases are developed.
4. Different modules are developed based upon statement coverage, path coverage, fault coverage and branch coverage.
5. The program is checked for different test cases and the bugs are found out.
6. Merge all the coverage criteria.
7. Apply the prioritization algorithm.

In the final step we get the prioritized test cases with a new coverage criteria included i.e., branch coverage which is used to validate that all branches in the code are reached and ensured that no branches lead to abnormal program's operation. Branch coverage is the easiest way to perform testing of workflow of program.

Table 1: Proposed Criterion Based Test Case Prioritization [6]

Algorithm for Test Case Prioritization Technique for Regression testing:

Input: Test cases with different coverage criterion covered

Output: Prioritized test cases

1. Merge the entire coverage criteria tables (statement, fault, path, and branch) with Test cases Ti.
2. Select the test case Ti, covering maximum number of criteria modules fj.
3. Put the test case Ti in a separate priority list (TCAL- Test Case Array List) with the coverage modules fj associated with it.
4. Repeat step 2 & 3 with rest of the test cases until all the coverage modules get covered up.
5. Arrange the test cases Ti of priority list in descending order based on maximum coverage modules.

In the proposed algorithm, Test Case Array List (TCAL) is used to store the prioritized test cases. Test suite Ti and its associated coverage criteria are given as input to the prioritization algorithm. In the first step of the algorithm, test cases and its associated coverage criteria are combined together. The test case and its associated coverage criteria whichhasmaximum coverage is removed and added in the TCAL in step 2. Then all the coverage criteria, associated with that test case is removed from the list. The next step is to find out the test case and their associated coverage criterion which has maximum coverage is removed and added in TCAL. This step is repeated until no test case has association with the coverage criteria. Finally, the remaining test cases are prioritized [6].

A. Regression Testing with JAVA

To exemplify the proposed algorithm, consider Bank Application in JAVA language with a test suite of twelve test cases, T1 through T12, such that the program contains twelve faults detected by those test cases, as shown in Table 2. For the same program test cases T1

through T12 covers 20 statements are shown in Table 3. For the same program test cases T1 through T12 covers 6 branches as shown in Table 4. Same program test cases T1 through T12 covers 6 paths as shown in Table 5. The main objective of the proposed multiple criterion based test case prioritization is to cover more than one criterion by executing single set of test cases. Therefore by merging all criterion based on its associated test cases the objective will be met. The test cases and its associated coverage criteria are merged as shown in Table 6.

Each row in the matrix corresponds to a test case and each column represents coverage criteria. The 'X' in a row denotes that the pattern covers the corresponding coverage criteria, while the empty denotes that it does not. The main aim is to reduce the number of test cases so that there would be as many columns with at least one 'X' as in the initial matrix.

To elucidate the proposed work test case T1 will be removed from the table, because the test case T1 is associated with maximum coverage criteria. Hence the test case T1 and its associated coverage criteria S8, S12, B5, B6, P5, P6 and F8 will be removed. The test case T1 and the coverage criteria that it covers denoted by 'X' will be removed from the Table 5 and T1 will be added in TCAL. This step is continued till there is no relationship between the test cases and coverage criteria. After applying the proposed algorithm final test cases will be T1, T4, T6 and T8.

Table 2: Test Cases and its associated faults Coverage

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
T1								X			
T2											
T3											
T4									X		
T5									X		
T6										X	
T7											
T8											X
T9											
T10											
T11									X		
T12										X	

Table 3: Test Cases and its associated statement Coverage

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17	s18	s19	s20
T1								X				X								
T2								X												
T3								X												
T4								X					X							
T5								X					X							
T6								X						X						
T7								X												
T8								X								X				
T9																				
T10																				
T11								X					X							
T12								X						X						

Table 4: Test Cases and its associated branch Coverage

	B1	B2	B3	B4	B5	B6
T1					X	X
T2					X	
T3					X	X
T4					X	X
T5					X	X
T6					X	X
T7					X	
T8					X	X
T9						
T10						
T11					X	X
T12					X	X

Table 5: Test Cases and its associated path Coverage

	P1	P2	P3	P4	P5	P6
T1					X	X
T2					X	
T3					X	X
T4					X	X
T5					X	X
T6					X	X
T7					X	
T8					X	X
T9						
T10						
T11					X	X
T12					X	X

Table 6: Test Cases of Bank Application with its associated Faults, Statements, Branches and Paths Coverage in JAVA.

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15	s16	s17	s18	s19	s20	B1	B2	B3	B4	B5	B6	P1	P2	P3	P4	P5	P6	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
T1								X				X													X	X					X	X								X				
T2								X																	X						X													
T3								X																	X	X					X	X												
T4								X					X												X	X					X	X								X				
T5								X					X												X	X					X	X								X				
T6								X						X											X	X					X	X									X			
T7								X																	X						X													
T8								X								X									X	X					X	X										X		
T9																																													
T10																																													
T11								X					X												X	X					X	X									X			
T12								X						X											X	X					X	X										X		

The test case 1 is associated with maximum of seven criteria. Then the test case 1 and its coverage criteria S8, S12, B5, B6, P5, P6 and F8 are removed from the graph.

Also the row one will be removed from the table and added to the TCAL.This step is continued till there is no relationship between the test cases and coverage criteria. After applying the proposed algorithm final test cases in TCAL will be T1, T4, T6 and T8.Then the algorithm prioritizes the test cases in TCAL based on the number of criteria covered by each test case. The prioritized test case order based on the proposed new Algorithm is T1, T4, T6, and T8.

B. Regression Testing with C++

The second testing is done in regression testing with C++ as primary language.

Table 7: Test Cases and its associated faults Coverage

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
T1	X										
T2	X										
T3	X										
T4	X								X		
T5											
T6										X	
T7											X
T8		X									
T9											
T10											

Table 8: Test Cases and its associated statement Coverage

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
T1	x	x		x									X
T2	x	x											X
T3	x	x					X						X
T4	x	x							x				X
T5	x	x										X	X
T6		x								x			X
T7		x									X		X
T8													X
T9		x					X						X
T10		x											X

Table 9: Test Cases and its associated branch Coverage

	B1	B2	B3	B4	B5	B6
T1	X		X			X
T2	X		X			X
T3	X		X			X
T4	X		X		X	X
T5	X		X			X
T6	X		X		X	X
T7	X		X		X	X
T8	X					X
T9	X		X			X
T10	X			x		X

Table 10: Test Cases and its associated path Coverage

	P1	P2	P3	P4	P5	P6
T1	X		X			
T2	X		X			
T3	X		X			
T4	X				X	
T5	X					
T6	X				X	
T7	X				X	
T8	X					
T9	X		X			
T10	X			X		

The test case 4 is associated with maximum of twelve criteria. Then the test case 4 and its coverage criteria S1, S2, S9, S13, B1, B3, B5, B6, P1, P5, F1 and F9 are removed from the graph.Also the row four will be removed from the table and added to the TCAL.This step is continued till there is no relationship between the test cases and coverage criteria. After applying the proposed algorithm final test cases in TCAL will be T1, T3, T4, T5, T6, T7, T8 and T10.Then the algorithm prioritizes the testcases in TCAL based on the number of criteria covered by each test case. The prioritized test case order based on the proposed new Algorithm is T4, T6, T1, T7, T3, T5, T10 and T8.

Table 11: Test Cases of Bank Application with its associated Faults, Statements, Branches and Paths Coverage in C++

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	B1	B2	B3	B4	B5	B6	P1	P2	P3	P4	P5	P6	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
T1	x	x		x									x	x		x			x	X		X				X											
T2	x	x											x	x		x			x	X		X				X											
T3	x	x					x						x	x		x			x	X		X				X											
T4	x	x						x					x	x		x		x	x	X				X		X								X			
T5	x	x									x		x	x		x			x	X																	
T6		x							x				x	x		x		x	x	X				X											X		
T7		x								x			x	x		x		x	x	X				X												X	
T8													x	x					x	x							X										
T9		x					x						x	x		x			x	x		x															
T10		x											x	x			x		x	X			X														

V. RESULTS ANALYSIS

Software testing process is basically used to save testing resources and to find maximum bugs from limited resources. To provide solution for it we have done some changes in regression testing by introducing multiple coverage criterions for testing.In our research we have mainlydone ad hoc regression testing, it is a testing in which test cases are made only if any bug is found in the application. Bank application (BA) isconsideredto verify the practicality of the proposed method. Testing is done in two programming languages JAVA and C++ for checking compatibility and then prioritizing the test cases for both the languages for same Bank application.The conducted empirical study of experiment is given in the table 12.

Table 12

Program	Language	No. Of Test Cases	No of Modules	No. of faults
Bank application (BA)	JAVA	12	9	13
Bank application (BA)	C++	10	6	23

To measure the effectiveness of the proposed method a modified Average Percentage of Fault Detection (APFD) [21] metrics is used. APFD measures the average of the percentage of faults detected over the life of the suite. Let T be a test suite containing n test cases and let F be a set of m faults revealed by T. Let TFi be the first test case in the recorded test suit T of T that reveals fault$i$. The APFD value for T is given by the following equation from [21]

$$APFD = 1 - \frac{TF1 + TF2 + \ldots + TFm}{nm} + \frac{1}{2n} \quad (1)$$

In this work, statement coverage, fault coverage, path coverage and branch coverage criteria are considered. In addition to APFD metric, Average Percentage of Statement Detection (APSD), Average Percentage of Branch Detection (APBD), Average Percentage of Path detection (APPD), and Average Percentage of Multiple Criterion Detection (APMCD) are used.

$$APSD = 1 - \frac{TS1 + TS2 + \ldots + TSm}{nm} + \frac{1}{2n} \quad (2)$$

$$APPD = 1 - \frac{TP1 + TP2 + \ldots + TPm}{nm} + \frac{1}{2n} \quad (3)$$

$$APBD = 1 - \frac{TB1 + TB2 + \ldots + TBm}{nm} + \frac{1}{2n} \quad (4)$$

$$APMCD = 1 - \frac{TMC1 + TMC22 + \ldots + TMCm}{nm} \cdot \frac{1}{2n}$$

The results for before prioritization and after prioritization are analyzed using normal prioritization method and multiple criterion based test case prioritization method for the criterion fault coverage, statement coverage, branch coverage and branch coverage. The summary of average percentage fault detected value for normal test case prioritization and criterion based test case prioritization for the software Bank Application is shown in Table 13. Our results show that the criterion based test case prioritization yield better results than the normal test case prioritization technique.

Table 13: Summary of APFD values, APSD values, APPD values, APBD values and APMCD values for Normal and Criterion Based Test Case Prioritization

ISSN: 2321-9653

Program		Language	Normal Test Case Prioritization					Multiple Criterion Based Test Case Priorit				
			Before Prioritization					After Prioritization				
			APFD	APSD	APPD	APBD	APMCD	APFD	APSD	APPD	APBD	A
Bank Application	Proposed Work	JAVA	91.36%	84.93%	84.29%	84.29%	98.39%	79.80%	72.11%	72.11%	72.11%	9
	Existing Work	JAVA	72.26%	68.45%	75.24%	-	-	80.19%	82.44%	82.44%	-	8
Bank Application	Proposed Work	C++	91.52%	81.52%	87.17%	81.08%	98%	89.94%	80.16%	86.14%	79.61%	9
	Existing Work	C++	63.32%	67.52%	69.13%	-	-	76.32%	74.89%	76.45%	-	8

The results before prioritization and after prioritization are analyzed using normal test case prioritization method and multiple criterion based test case prioritization method for the criterion fault coverage, statement coverage path coverage and branch coverage. The summary of average percentage fault detected value for normal test case prioritization and criterion based test case prioritization for the software Bank Application is shown in Table 13.

Figure 1 shows the values for APFD, APSD and APPD for Bank application in Java. The graphs shows the difference before prioritization of test cases and after multiple criterion prioritization of test cases for proposed work and existing work. The values for APFD, APSD and APPD is found better for the work done as compared to the proposed work.

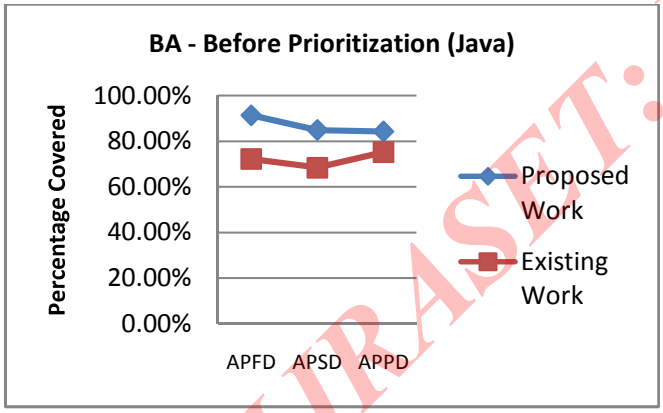
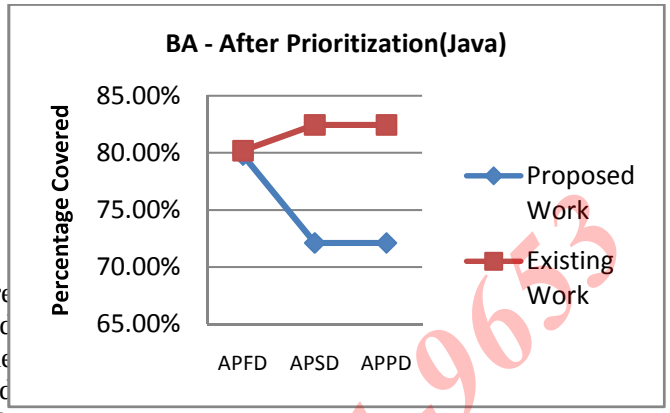


Figure 1: Comparative Analysis for Bank Application in JAVA

a) Before Prioritization



b) After Prioritization

Figure 2 shows the values for APFD, APSD and APPD for Bank application in C++. The graph shows the difference before prioritization of test cases and after multiple criterion prioritization of test cases for proposed work and existing work. The values for APFD, APSD and APPD is found better for the work done as compared to the proposed work.

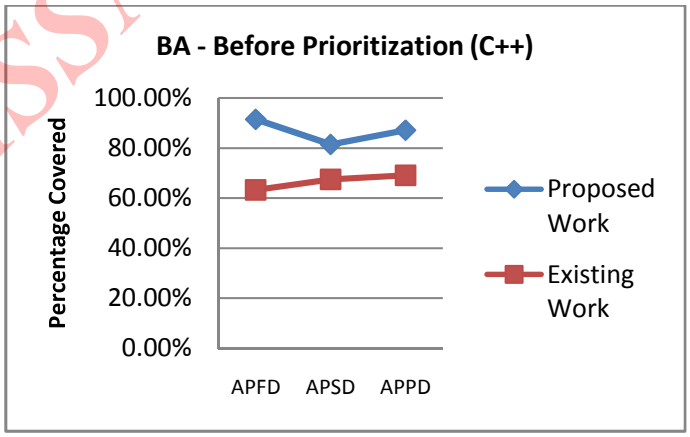
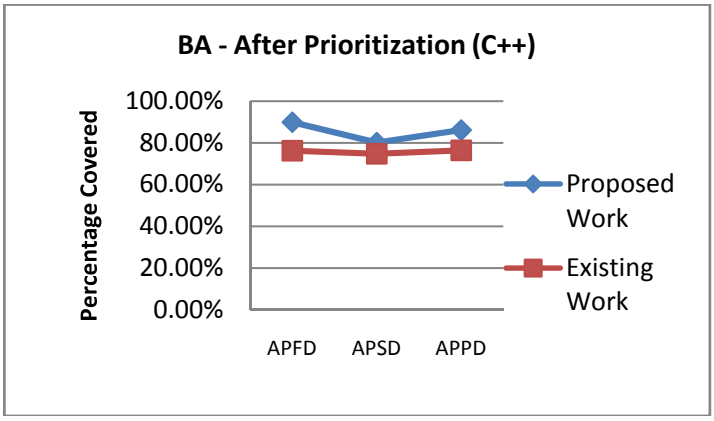


Figure 2: Comparative Analysis for Bank Application in C++

a) Before Prioritization



b) After Prioritization

Figure 3 shows the values for APFD, APSD, APPD, APBD and APMCD for Bank application in JAVA. The graphs show the difference before prioritization of test cases and after multiple criterion prioritization of test cases. Average Percentage for multiple criterion detection for BA is 99.23% which is 25.19% higher than the average of APFD, APSD, APPD and APBD. Also the value for APMCD is found 9.46 % better in the proposed work as compared to the existing work.

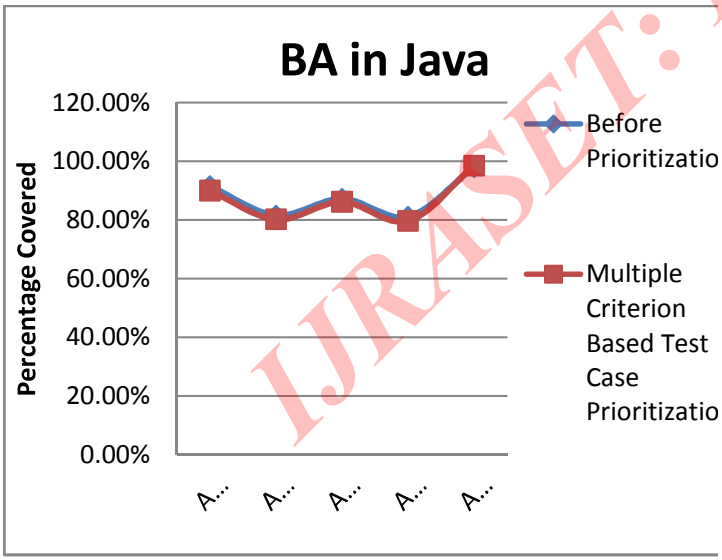


Figure 3: Graphical Representation of before prioritization and after multiple criterion based prioritization for BA in JAVA

Figure 4 shows the values for APFD, APSD, APPD, APBD and APMCD for Bank application in C++. The graphs show the difference before prioritization of test cases and after multiple criterion prioritization of test cases. Average Percentage for multiple criterion detection for BA is 98.43% which is 14.46% higher than the average of APFD, APSD, APPD and APBD. Also the value for APMCD is found 10.01% better in the proposed work as compared to the existing work.

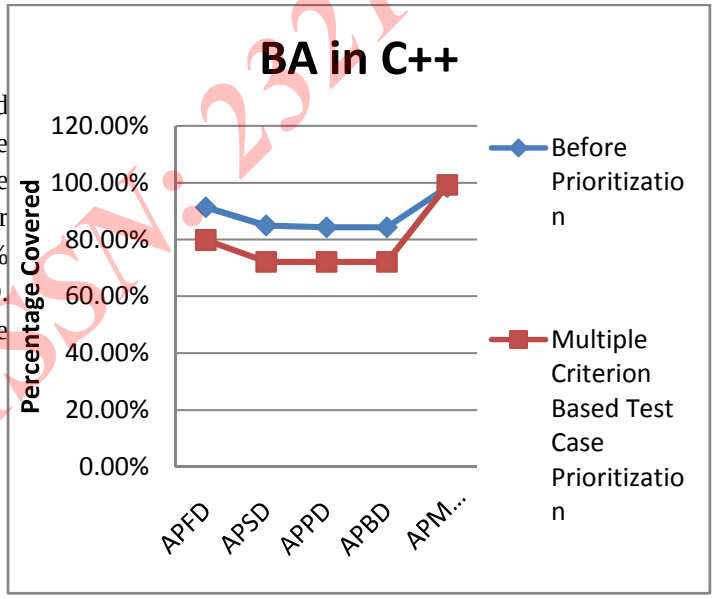


Figure 4: Graphical Representation of before prioritization and after multiple criterion based prioritization for BA in C++

VI. CONCLUSION

In this research we have used multiple criteria parameters for selection of test case priority process with applications built in JAVA platform and C++ platform for providing the optimized testing for software testing process. We have branch coverage,

statement coverage, path coverage and fault coverage parameters for better selection of the test cases. Such test model selection and test volume evaluation method had been applied to the software testing work of Java and C++ language base on same bank application, and compared with traditional software testing approach given in related study [6]. The proposed work shows better result with 25 % more bug finding with execution time which is 17 % less than traditional method. The resultant test provide better level of software testing by increases the efficiency of the testing by 35 % as compared to traditional work. From the experimental test effect and workload, the previous method is rational but the proposed optimized testing method is more effective, especially when testing small scale applications.

REFERENCES

[1] Todd L. Graves, Mary Jean Harrold, Jung-Min Kim, Adam Porter, and Gregg Rothermel. 2011. "An empirical study of regression test selection techniquies", ACM Trans. Softw. Eng. Methodology. 10, 2 (April 2011).

[2] Introduction to software testing available at http://www.onestoptesting.com/test-cases/designing.asp

[3] Hyunsook Do, SiavashMirarab, LadanTahvildari, and Gregg Rothermel, 2010. "*The Effects of Time Constraints on Test Case Prioritization: A Series of Controlled Experiments"*, IEEE Trans. on Software Engineering, Vol. 36, No.54, pp. 593-617.

[4] Software Testing available at http://testingsoftware.blogspot.in/2006/02/test-case.html

[5] Alexey G. Malishevsky, Joseph R. Ruthruff, Gregg Rothermel, Sebastian Elbaum, "Cost-cognizant Test Case Prioritization," Technical Report TR-UNL-CSE-2006-004, Department of Computer Science and Engineering, University of Nebraska–Lincoln, Lincoln, Nebraska, U.S.A., March 2006.

[6] N. Prakash, T. R. Rangaswamy," *Multiple Criteria Based Test Case Prioritization for Regression Testing*", European Journal of Scientific Research, Vol. 84, No.1, February 2012, pp.36 - 45.

[7] Rothermel G, Mary Jean Harrold and JeinayDedhia, 2000. "*Regression test selection for C++ Software*," Research Article Software Testing, Verification and Reliability. Vol. 10, Issue 2, pp 77-109.

[8] Srinivasan Desikan, Gopalaswamy Ramesh, 2006. "*Software testing principles and practices*", Pearson Education, 1st Edition.

[9] Mohd. Ehmer Khan, "*Different Forms of Software Testing Techniques for Finding Errors*", IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 1, May 2010.

[10] SahilBatra, Dr. Rahul Rishi, "*Improving Quality Using Testing Strategies*", Journal of Global Research in Computer Science, Volume 2, No. 6, June 2011.

[11]Fangchun Jiang, Yunfan Lu, "*Software testing model selection research based on Yin-Yang testing theory*", International Conference on Computer Science and Information Processing (CSIP), IEEE, Vol.9, 2012, pp.11-15.

[12] Abhijit A. Sawant, Pranit H. Bari and P. M. Chawan, "*Software Testing Techniques and Strategies*", International Journal of Engineering Research and Applications (IJERA), pp. 980-986, Vol. 2, Issue 3, May-Jun 2012.

[13]Zheng Li, Mark Harman, and Robert M. Hierons, 2007. "*Search Algorithms for Regression Test Case Prioritization*",IEEE Transactions On Software Engineering, Vol. 33, No. 4, pp 225 – 237

[14] Hyunsook Do, Gregg Rothermel, and Alex Kinneer, "*Empirical Studies of Test Case Prioritization in a Unit Testing Environment***",** 2004. IEEE Proceedings of the 15th International Symposium on Software Reliability Engineering (ISSRE'04).

[15] Md. ImrulKayes, "Test Case Prioritization for Regression Testing based on Fault Dependency", IEEE, pp- 48-52.

[16] Srinivasan Desikan,Gopalaswamy Ramesh, 2006. "software testing principles and practices",Pearson Education, 1st Edition