Asian Journal of Computer Science and Technology
ISSN: 2249-0701 (P) Vol.7 No.1, 2018, pp.87-92

DOI: https://doi.org/10.51983/ajcst-2018.7.1.1824

A Technique for Regression Testing of Object Oriented Software

Vedpal[1] and Naresh Chauhan[2]
[1]Department of Information Technology and Computer Applications
[2]Department of Computer Engineering
YMCA University of Science and Technology, Faridabad, Haryana, India
E-Mail: ved_ymca@yahoo.co.in, nareshchauhan19@gamil.com
(Received 13 February 2018; Revised 25 February 2018; Accepted 5 March 2018; Available online 13 March 2018)

***Abstract* - The main challenge in testing an OOS is that there are large number of test cases which are not feasible to execute all of them within constrained time and budget. This paper proposes some factors which can be used to prioritize the test cases in order to have an effective testing. The considered factors are based on the testing history and structured analysis of the software. These factors may be nature of bug, capability of a test case, execution time, business impact, coverage of code in terms of classes(old and new classes), etc. Every factor has been assigned a positive weight which shows the criticality of the factor and ability to introduce the errors in the software. The prioritized set of test cases thus obtained is helpful to discover maximum bugs as early as possible.**
***Keywords:* Regression testing, test case prioritization, history based regression testing**

I. INTRODUCTION

Software refinement and maintenance is very expensive process. During the life cycle of software it may be modified many times due to customer requirements, enhancement of the current feature, adding new feature etc. The software is also get modified if any bug occur in the software due to removal of a bug in the software. Every time new modification is introduced in the software, there is higher probability of the other components of the software being affected. So respective to any modification in software testers have to ensure that the modified part of the software does not put any critical impact on the other part of the software.

Modification and bug fixing exist in every phase of software. With every change, whether it is a minor change or critical change there is need to check the software again to validate that there has been no adverse impact on the other working part of the software. Software is put under regression testing if any one of the following reason occurs.

1. Any part of the software fail to meet the requirement
2. Adding the new functionality in the software.
3. Refining the current functionality of the software
4. After the bug fixed in the software.

To ensure that modification of the current working component and adding new component in the software do not adversely effect on the software a selective retesting of the system is performed. The process of retesting the modified and updated software is called the regression testing. But due to some constraints like time, resource, budget, business impact, it's becoming very challenging task to retest the software. To perform a regression testing the software has the large number of test suit. It is very expensive to execute all test cases to test the software so the test cases should be executed in an order such that maximum faults are detected by earlier test cases by consuming less time, and cost.

The process of ordering the test cases with the intention to find the maximum test cases is called test case prioritization. In this paper a technique for regression test case prioritization for object oriented software is presented. The presented approach orders the test cases on the basis of some factors which are related to the past testing history of the software which are going to be retested again to ensure the bug free software after incorporating modifications. Every factor has been assigned a positive weight which shows the contribution of the factor to discover the higher severity faults. The weight is assigned by the developer, tester, and project manager on the basis of their experience in relevant field. To validate, the APFD value of the proposed approach is compared with others existing and optimized approaches

II. RELATED WORK

Ahlam Ansari *et al.,* proposed [1] an approach for regression test case prioritization approach using ant colony algorithm. The approach firstly takes the test cases which have covered the maximum faults followed by the selection of test cases covering the remaining faults.

Saloni Ghai *et al* presented [2] an approach for regression test case prioritization using hill climbing. The proposed technique traversesthe DFD of the software and determines the importance of the functions. These function's importance are used by the hill climbing approach to prioritize the test cases.

Wenhao *et al.,*combined [3] the algorithm of clustering and scheduling with the aim to enhance the effectiveness of the regression testing. They used the clustering algorithm to merge the test cases in cluster having the similar properties and scheduling algorithm to assign the priority of execution to the test cases. The execution frequency is

assigned on the basis of the predictive fault detection rate, waiting time in candidate set to detect the all faults.

Samia Jafrin *et al*., used [4] the rate of severity associated with the fault to prioritize the regression test cases . They found that the latest research did not addressed the dependency among the faults. They considered the faults which are fully or partially dependent on the other faults. They proposed algorithm that distinguished the improvement between the independent and fully and partially dependent faults.

Wasiur Rhmann *et al.,* proposed [5] the fuzzy logic based test case prioritization for regression testing. The diagram of state machine is used to capture the system behavior and the information related to the risk associated with states. After calculating the value of risk exposure the state diagram is converted in to the weighted extended finite state machine(WEFSM).

The WEFSM is used to generate the different test paths by traversing in the depth first manner. For each generated path the maximum and minimum risk exposure value is calculated which are further used by fuzzy expert system to categorized the test cases.

Soumen Nayak *et al*., prioritized[6] the regression test cases using the four factors. These factors are the rate of fault detection, number of faults detected, test case ability of the risk detection and the test case effectiveness. They determine the effective test case ranking by calculating the sum of the value of the four considered factors.

Sapna P G *et al*., proposed [7]black box approach for generating the test cases for the regression testing. The UML and activity diagrams have been used to model the requirements and elaborated the functionality. They used the steiner tree algorithm with the objective to generate the minimal test set which are used to check functionality.

Bo Jiang *et al.,* proposed [8]input based randomized test case prioritization technique. They introduceda novel family of input based local beam search adaptive randomized technique.They create adaptive based randomized exploration with the randomized test strategy. They addressed the issues regarding the cost efficiency by a novel design on the size of randomized candidate set with the local beam search.

Almanda Schwartz *et al*., presented [9] the technique to investigate and determine the most cost effective technique to perform regression testing. The technique is choosing for a particular regression testing session. They also presented the comparative study adaptive test prioritization technique existed till date. The outcome of the studies indicates the proposed approach is very effective for cost saving in regression testing as compared to other existing regression testing technique.

Yuen Tak Yu *et al*., proposed [10] a fault based test suit prioritization for specification – based testing. They used the theoretical knowledge ability of detection of faults and relationship between the test cases. The test cases are generated on the basis of the faults in fault model.The experimental result of the proposed shows that all faults are detected by executing only about 72 % of the prioritized test suits.

Erik Rogstad *et al.,* presented [11] an approach for selection of the black box regression test cases for database application. They partition the input domain of testing system by using classification tree model. They select the test cases from the partition on the basis of similarity between the test cases. The experimented results show that presented approach provides the higher fault detection rate.

Alireza Khalilian *et al.,* used [12] the historical data of test cases to prioritize them. They compute the priority of the test cases by computing the test case prioritization equation. For computing the equation the historical information of test cases with constant coefficient is used.

Yu –Chi Huang *et al.,* proposed [13] a cost cognizant test case prioritization technique using historical record of test cases. They used genetic algorithm to order the test cases on the basis of the gathered historical data of latest regression testing.

Breno Miranda *et al*., proposed [14]a scope- aided technique to prioritize, selection and minimization of the test cases of white box testing. They used the reuse context to reorder and selecting the test cases.By critically reviewing the existed work it has been observed that a lot of work has been done in the regression testing but still the researcher hopesfor effective technique. The researchers used the various algorithms like ant colony, hill climbing etc. Some researchers have taken some factors related to the past history of testing to order the test cases. But they don't use the efficiency and capability of a particular factor to detect the critical and maximum bug as earlier as possible. In this paper a novel technique for object oriented software is presented.

III. PROPOSED WORK

The proposed approach prioritizes the regression test cases on the basis of some factors related to the past testing history and coverage of the code in term of classes of the software which is going to be retested after incorporating some modifications in it. All the considered factors have been shown in the table1. All the factors have been assigned a positive weight which shows the capability of the test cases to discover the maximum fault by consuming less time and cost. These factors may be considered for the prioritization factor for the regression testing of the software. The value of the considered factors is determined by using the information of past history of the test cases.

TABLE I PRIORITIZATION KEY OF TEST CASES

S. No.	Factor Name	Factor Weight
1	Severity of Bug	.25
2	Capability of Detecting the Bug	.2
3	Coverage of impacted code	.15
4	Impact on business	.3
6	Execution Time	.1

The test cases are thus prioritized on the basis of a value known as regression test case prioritization value (RTCPV) which is calculated by the following formula

$$RTCPV = \sum_{1}^{n} TFVij * FWj \quad (1)$$

Where TFV is the estimated value of the j^{th} factor and FW is factor weight of j^{th} factor for i^{th} test case.

In regression test cases if the test cases are new then it is assigned the highest priority because it is going to be executed first time and has the capability of detecting the maximum faults. It may be possible that new test cases are more than one. In such type of dilemma the newly test cases are prioritized on the basis of coverage of modified classes and coverage of new classes. The overall process of test case prioritization is shown in Figure1, which is being a described further in subsequent sections.

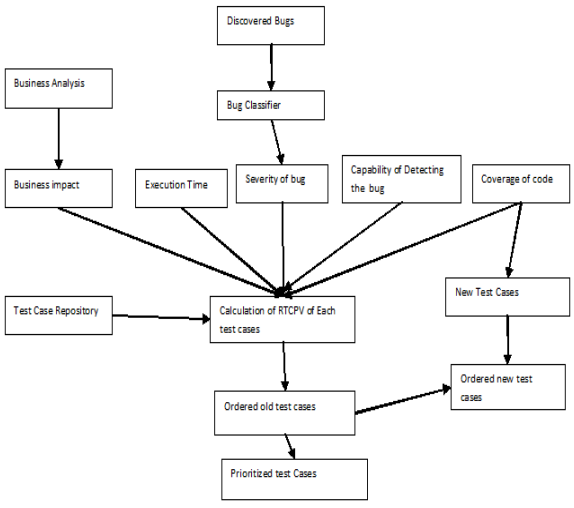

Fig. 1 Overview of Proposed Approach

IV. THE PRIORITIZATION FACTORS CONSIDERED IN THE PRESENTED APPROACH

Severity of Bug: This factor uses the classification of the bug on the basis of the impact on the software. On the bugs are classified in the four categories. These categories are [15] critical bug, major bug, medium bug and minor bug. Here on the basis of the past discovery of the bugs by test cases a scaling of bugs (1-10) may be given as below

Value = 10 is all discovered bugs are critical
Value = 8- 9 detected bugs are critical and major, medium and minor bugs
Value = 7 if all detected bugs are major and medium
Value = 5 - 6 if the all bugs are major bugs
Value = 4 if the bugs are medium and minor
Value = 2-3 if the all bugs are medium
Value = 1 of the bugs are minor bugs

Capability of Detecting the Bug (CDB): This [6] factor shows the caliber of the test case to detect the maximum bugs by executing the test cases. The value of this can be estimated by the following formula

$$CDB = (TBC/TDB) *10 \quad (2)$$

Where TDB is the total detected bug by all test cases and TBC is number of bugs detected by the current test cases.

Coverage of Code(CC): This factor shows the coverage of the code in terms of classes (modified and unmodified) and methods by the test cases. The value of this factor is based on the basis of coverage of the modified and updated classes. This value can be calculated by the following formula

$$CC = (TCC/TC)*10 \quad (3)$$

Where TC is Total classes in the software and TCC is number of covered classes by the test cases.

On the basis of this formula the value between 0 to 10 is assigned.

Business Impact: This factor shows that if the particular function being covered by the test cases is not executed successfully then how much it puts impact on the business of customer. On the basis of the business impact by test cases the value between 0 to 10 is assigned.

Execution Time (ET): This factor shows the time taken by the test case to execute the target functionality. The value of this factor is assigned on the basis of the formula 4

$$ET = (PT/TT) \quad (4)$$

Where PT is execution time ith test case , TT is the total time taken in executing all test cases and ET is the estimated value of execution time of the particular test cases

V. RESULTS AND ANALYSIS

For the experimental applicability and analysis of the proposed approach, it has been applied on a case study [16] implemented in Java. To check effectiveness of the technique to detect rate of fault detection, intentionally

some faults have been added in the considered case study and the bugs are detected manually. The outcome of the case study is given below:

Case Study: In this case study the presented approach is applied on a practical problem of Banking. In the considered example [16] the user can perform the operation of deposit, withdrawal, calculate interest and display the account information on saving account and current accounts.

The Table II shows the test case history of the program before applying the modification

From the past testing history of the case study the total 16 bugs are discovered by executing the 10 test cases. Now by using the above history the table3 shows the values of various factors which are used to prioritize the test cases for regression testing.

TABLE II TESTING HISTORY OF CONSIDER CASE STUDY

Test case	Determined value of nature of Bug	Nature of Bug	Execution time of test case (cs)
TC1	1	Minor=1	.2
TC2	2	Major =1 minor=1	.3
TC3	1	major=1	.25
TC4	1	Minor=1	.2
TC5	1	Major =1	.25
TC6	2	Minor=2	.25
TC7	2	Major=2	.3
TC8	2	Major=1 Medium=1	.35
TC9	3	Critical =1 Major =2	.35
TC10	1	Medium=1	.2

TABLE III DETERMINED VALUE OF CONSIDERED FACTORS

Test case	Determined value of Severity of Bug	Capability of Detecting Bug(CDB)	Execution time of test case (ET)	Impact on business	Coverage of code by test cases (CC)	Estimated RTCPV
TC1	1	(1/16)*10= 62.5	(.2/2.65)*10 =0.75	2	(4/5)*10 =8	(1*.25) +(0..625*.2)+(.75*.1)+(2*.3)+(8*.15) = 2.25
TC2	7	.80	1.13	8	8	5.623
TC3	5	.625	.94	8	8	5.069
TC4	1	.625	.75	9	8	4.35
TC5	5	.625	.94	5	8	4.169
TC6	1	.80	.94	2	8	2.304
TC7	5	.80	1.13	8	8	5.122
TC8	7	.80	1.32	9	8	5.942
TC9	9	1.87	1.32	9	8	6.656
TC10	3	.625	0.75	7	8	4.249

The ordered test cases are TC9, TC8, TC2, TC7,TC3, TC4, TC10,TC5,TC6,TC1. The Table IV shows the order of the test cases after applying the random, reverse, Nayak *et al.* [6] and the proposed approach

TABLE IV TEST CASE ORDER OF THE VARIOUS APPROACHES AND PROPOSED APPROACH

S. No.	No order	Random Order	Reverse Order	Nayak approach	Proposed approch
1	TC1	TC5	TC10	TC9	TC9
2	TC2	TC4	TC9	TC2	TC8
3	TC3	TC10	TC8	TC7	TC2
4	TC4	TC1	TC7	TC8	TC7
5	TC5	TC8	TC6	TC6	TC3
6	TC6	TC9	TC5	TC5	TC4
7	TC7	TC3	TC4	TC3	TC10
8	TC8	TC6	TC3	TC10	TC5
9	TC9	TC7	TC2	TC1	TC6
10	TC10	TC2	TC1	TC4	TC1

The Table V shows the faults detected by the test cases.

TABLE V FAULTS DETECTED BY TEST CASES

	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9	TC10
F1	*					*				
F2								*		
F3		*								
F4		*								
F5			*	*						
F6			*							
F7			*							
F8				*						
F9					*					
F10							*			
F11								*		
F12									*	
F13									*	
F14									*	
F15										*

The APFD of unordered, random order, reverse order, nayak approach and the proposed approach is shown in figure 2 to figure 6

Fig. 2 APFD Graph of the unordered test cases

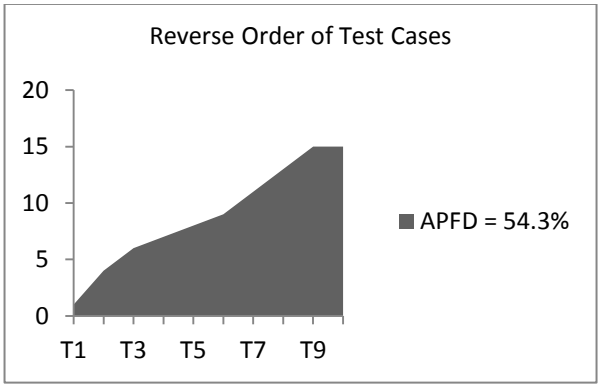

Fig. 3 APFD Graph of the test cases in reverse ordered

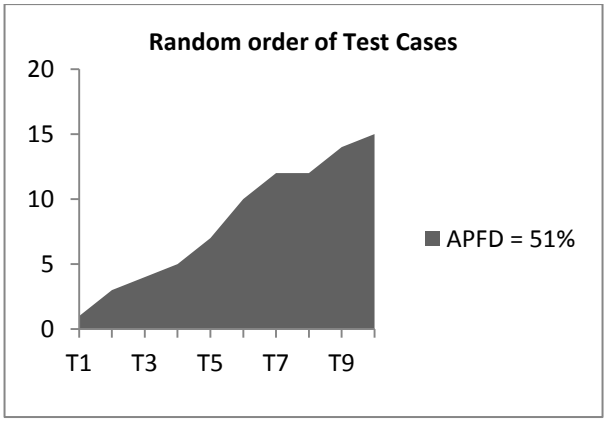

Fig. 4 APFD Graph of the test cases in Random Ordered

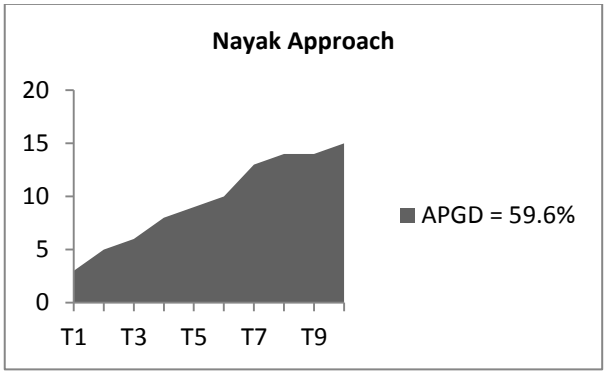

Fig. 5 APFD Graph of the test cases ordered by Nayak et. *al.*, Approach

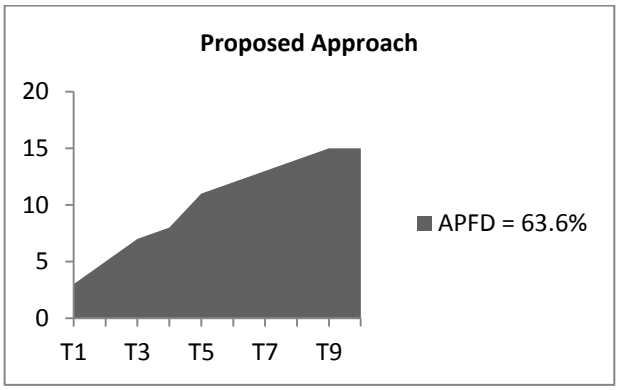

Fig. 6 APFD Graph of the test cases ordered by proposed approach

VI. COMPARATIVE STUDY OF THE PROPOSED APPROACH

The figure 7 and table 6 shows that the proposed approach is to discover the maximum faults earlier as compare to the other approaches. The result of the proposed approach is very promising and helps to reduce the testing cost of the software.

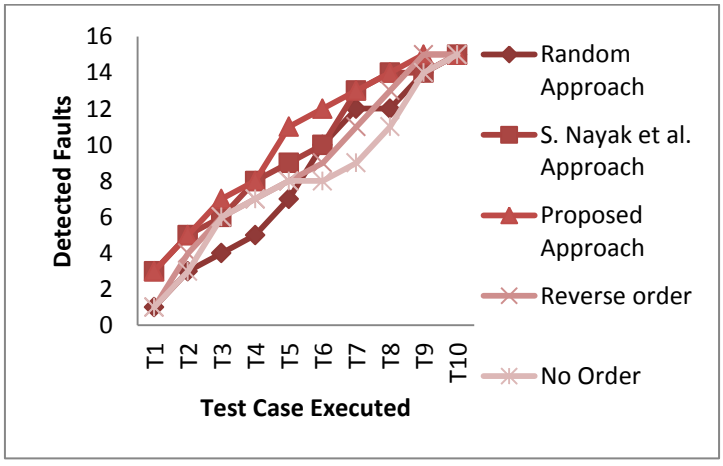

Fig. 7 APFD Graph of Various approaches

TABLE VI APFD VALUE OF THE PROPOSED APPROACH AND OTHERS APPROACHES

S. No.	Approach applied	Percentage of APFD
1	Unordered	50%
2	Reverse Ordered	54.3%
3	Random Ordered	51%
4	Nayak Approach ordered	59.6%
5	Proposed Approach	63.6%

VII. CONCLUSION

In this paper a novel technique for test cases prioritization for the regression testing of the software is presented. The test cases are prioritized on the basis of the five factors. Every considered factor has been assigned a positive weight which shows the ability of the factor to discover the faults. The weight is assigned by the developers, testers and experts on the basis of their experience and skills. The value of considered factors is calculated by analyzing the past testing history of the software and the coverage of the code by the particular test case. For experimental validation and to check the effectiveness of the proposed approach it has been applied on software and the obtained APFD result is compared with the similar existed latest approach. The comparative study shows the effectiveness of the proposed approach over the others similar approaches.

REFERENCES

[1] N. Chauhan, "Software Testing Principles and Practices", *Oxford University Press*, 2010.

[2] Y. Tak Yu, M. Fai Lau, "Fault-based test suite prioritization for specification-based testing", *Information and Software Technology*, Vol. 54, pp. 179–202, 2012.

[3] M. Abdollahi Khalilian and Y. Fazlalizadeh Azgomi, "An improved method for test case prioritization by incorporatinghistorical test case data", *Science of Computer Programming*, Vol. 78, pp. 93–116, 2012.

[4] Y.Chi Huangc, K.Li Penga, C.Yu Huang, "A history-based cost-cognizant test case prioritization technique in regression Testing", *The Journal of Systems and Software*, Vol. 85, pp. 626– 637, 2012.

[5] E. Rogstad, L. Briand and R. Torkar, "Test case selection for black-box regression testing of databaseApplications", *Information and Software Technology*, 2013.

[6] S. Malhotra and S. Chaudhary, "*Programming in Java*", Oxford University Press, 2014.

[7] P G Sapna and A. Balakrishnan, "An Approach for Generating Minimal Test Cases for RegressionTesting", *Procedia Computer Science*, Vol. 47, pp. 188 – 196, 2015.

[8] W.K. Jianga Chan, "Input-based adaptive randomized test case prioritization: A local beam search approach", *Journal of Systems and Software*, Vol. 105, pp. 91-106, July 2015.

[9] S. Jafrin, D. Nandi and S. Mahmood, "Test Case Prioritization based on Fault Dependency" *I.J. Modern Education and Computer Science*, Vol. 4, pp. 33-45, 2016. [Online] Available: http://www.mecs-press.org/DOI:10.5815/ijmecs.2016.04.05

[10] Ansari, A. Khan and K. Mukadam, "Optimized Regression Test using Test Case Prioritization" *7th International Conference on Communication, Computing and Virtualization 2016*, *Procedia Computer Science*, Vol. 79, pp. 152 – 160, 2016.

[11] A. Bertolino Miranda, " Scope-aided test prioritization, selection and minimization for software reuse", *The Journal of Systems and Software*, pp. 1–22, 2016.

[12] S. Ghai and S. Kaur, "A Hill-Climbing Approach for Test Case Prioritization", *International Journal of Software Engineering and Its Applications*, Vol. 11, No. 3, pp. 13-20, 2017. [Online] Available: http://dx.doi.org/10.14257/ijseia.2017.11.3.02

[13] W. FU, H. YU, G. FAN, and X. JI, "Coverage-Based Clustering and Scheduling Approach for Test CasePrioritization", *IEICE TRANS. INF. & SYST.*, Vol. E100–D, No.6, June 2017.

[14] W. Rhmann and V. Saxena, " Fuzzy Expert System Based Test Cases Prioritization from UML State Machine Diagram using Risk Information", *I.J. Mathematical Sciences and Computing*, Vol. 1, pp. 17-27, 2017. [Online] Available: http://www.mecs-press.net DOI: 10.5815/ijmsc.2017.01.02

[15] S. Nayak, C. Kumar and S. Tripathi, "Enhancing Efficiency of the Test Case Prioritization Techniqueby Improving the Rate of Fault Detection", *Arab J Sci Eng*, DOI 10.1007/s13369-017-2466-6

[16] H. Do Schwartz, "Cost-effective Regression Testingthrough Adaptive Test Prioritization Strategies", *The Journal of Systems & Software*, S0164-1212(16)00016-9DOI: 10.1016/j.jss.2016.01.018 Reference: JSS 9661.