Test Cases Prioritizer for Regression Testing

[1]Gunjan Agarwal, [2]Dr. Jayant Shekhar
[1]Student, [2] Professor (Director)
[1]Department of Computer Science & Engineering,
[1]Subharti Institute of Technology & Engineering, Meerut, India

***Abstract* - Regression testing is an essential part of the software development method. It is a testing that is performed to increase confidence in the knowledge that newly introduced software features do not obstruct the existing features. Test case prioritization plays a vital role in it. In this paper we have proposed a new technique for test case prioritization which is the result of some modifications made in additional fault- exposing- potential (FEP) prioritization.**

***Index Terms* - Regression testing, test case prioritization, Fault-exposing-potential (FEP).**

I. INTRODUCTION

Test Case Prioritization means to organize test cases which aim at increasing the efficiency and effectiveness of goals. Some possible goals of prioritization are: increasing the rate of fault detection of a test suite; increasing the coverage of code in the system which is under test; increasing the rate of high - risk faults detection. In section 2 of this paper we have defined regression testing in detail and discussed its techniques to reduce the regression testing cost. In section 3 we have described the test case prioritization which attempts to increase the fault detection rate of a test suite. In section 4 we have proposed a new technique of test case prioritization. Section 5 and 6 present results and overall conclusion.

II. REGRESSION TESTING

Software inevitably changes, however well written and designed it may be initially. The word *regress* [4] [13] refers to return to a previous, usually worse, state. Regression testing is an expensive process in which test suites are executed ensuring that no new errors have been introduced into previously tested code. In this method, design documents are replaced by extensive, repeatable, and automated testing of the entire software package at every stage in the software development cycle. Regression testing[1] [13]refers to that portion of the test cycle in which a program S' is tested to ensure that the newly added or modified code behaves correctly and also that the code carried over unchanged from the previous version S continues to behave correctly. Thus regression testing is useful, and needed, whenever a new version of a program is obtained by modifying an existing version. Regression testing (RT) is used to revalidate the modifications of the software [2] .
The term [9] "corrective regression testing" refers to regression testing of a program acquired by making corrections to the previous versions. Another term "progressive regression testing" refers to regression testing of a program procured by adding new features. A typical regression testing scenario often includes both corrective and progressive regression testing techniques.
The various techniques [4] of regression testing are: (1) Retest all; (2) Regression Test Selection; (3) Test Case Prioritization; (4) Hybrid Approach. Figure 1 depicts various regression testing techniques [3]

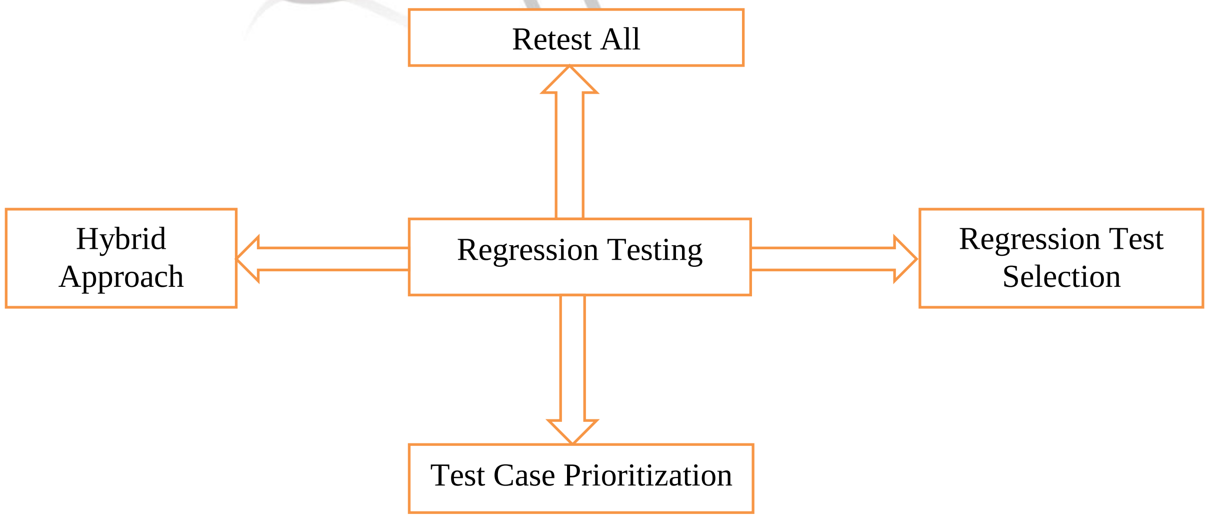

Figure 1. Regression Testing Techniques

A. Retest all

Retest all [5] method is one of the traditional methods for regression testing in which all the tests in the existing test suite are executed again. So the retest technique [6] is very expensive as compared to techniques which will be discussed further as regression test suites are costly to execute in full as it require more time and budget.

B. Regression Test Selection

Regression test [7] selection technique is performed to lower the regression testing cost as it selects a subset of the test suite to rerun them in spite of rerunning the whole set of test suite. Selection of test cases, from existing test suite, is done on the basis of information about the program, modified version of program, and already existing test suite. RTS [5] divides the existing test suite into (1) Reusable test cases; (2) Retestable test cases; (3) Obsolete test cases. In addition to this classification RTS may create new test cases that test the program for areas which are not covered by the existing test cases.

C. Test Case Prioritization

The main purpose of test case prioritization [8] is to rank test cases execution order to detect fault as early as possible.

D. Hybrid Approach

The fourth regression technique is the hybrid Approach of both Regression Test Selection and Test Case Prioritization. A whole lot of research and algorithms are being explored in this direction [4] .

III. TEST CASE PRIORITIZATION

This technique of regression testing prioritize the test cases so as to increase a test suite‘s rate of fault detection that is how quickly a test suite detects faults in the modified program to increase reliability. This is of two types: (1) General prioritization [11] which attempts to select an order of the test case that will be effective on average subsequent versions of software (2) Version Specific prioritization which is concerned with particular version of the software.
Test Case Prioritization (TCP)[12] [14]Techniques let testers order their test cases so that test cases with the highest priority, according to some criterion, are executed beforehand in the regression testing process than lower priority test cases. We define the test case prioritization problem [3] as follows:

Given: *T*, a test suite, *P_T*, the set of permutations of *T*, and *f*, a function from *P_T* to the real numbers.
Problem: Find *T'* belongs to *P_T* such that (*T''*) (*T''* belongs to *P_T*) (*T''* $\neq$ *T'*) [*f* (*T'*)$\geq$ *f* (*T''*)].

In this definition, P_T represents the set of all possible prioritizations (orderings) of T, and f is a function that, applied to any such ordering, yields an award value for that ordering. (For simplicity the definition assumes that higher award values are preferable to lower ones.)

Techniques of prioritization

When an application is developed and it is tested for the first time a set of test cases means test suite is designed to verify and validate its functionality. To decide the priority of the test cases the various factors depending upon the need are decided then the priority is assigned to the test cases. There are nine techniques which could be used for this purpose [10].
We next describe the nine techniques listed in Table I.

table I. Test Case Prioritization techniques

S.no	Techniques
1.	No prioritization
2.	Random prioritization
3.	Optimal prioritization
4.	Total statement coverage prioritization
5.	Additional statement coverage prioritization
6.	Total branch coverage prioritization
7.	Additional branch coverage prioritization
8.	Total Fault- Exposing- Potential (FEP) prioritization
9.	Additional Fault- Exposing- Potential (FEP) prioritization

1. *No Prioritization*

In this case no techniques are implemented and is been used as an untreated test suit and it serves as a control.

2. *Random Prioritization*

This is applied to have an additional control in studies where the test cases are ordered randomly in the test suite.

3. *Optimal Prioritization*

In this technique known faults are been used so that its results can be used to measure the effects of other prioritization techniques which are to be used.

4. *Total statement coverage prioritization*

This is a coverage based technique of prioritization. It covers the program with any test cases and finds out that which statements were covered by the test cases; then these test cases can be prioritized on the bases of number of statements they covered. If more than one tests case covers equal number of statements then we have use some additional rules or order them randomly.

5. *Additional statement coverage prioritization*

This method covers the shortcomings of total statement coverage technique and iteratively selects a test case that gives the highest statement coverage and then adjusts the coverage information on rest of the test cases to find out their statements not yet covered. This whole process is repeated till at least one test case covers all the statements.

6. *Total branch coverage prioritization*

This technique is same as statement coverage prioritization but it just uses test coverage measured in form of program branches as there was the case of statements in other technique. We define *branch coverage* as coverage of each possible overall outcome of a condition in a predicate.

7. *Additional branch coverage prioritization*

This technique is same as additional statement coverage prioritization but the only difference is that it uses test coverage measured in term of program branches and not in statements.

8. *Total Fault- Exposing- Potential (FEP) prioritization*

Other method like statement and branch coverage takes in context that whether a statement or branch is been reached by some test cases or not and do not takes in context the case that some faults are more easily seen as compared to others. As some test cases can expose faults more easily as compared to others, so it is called as fault-exposing potential of a test case. In this work, to obtain an approximation of the fault-exposing-potential of a test case, we adopt an approach that uses mutation analysis to produce a combined estimate of propagation-and-infection that does not incorporate independent execution probabilities. The approach works as follows: Given program *P* and test suite *T*, we first create a set of mutants for *P*, noting which statement s_j in P contains each mutant. Next, for each test case t_i belongs to *T*, we execute each mutant version of P on $t_{i,,}$ noting whether t_i kills that mutant. Having collected this information for every test case and mutant, we consider each test case t_i and each statement s_j in *P*, and calculate the fault-exposing potential FEP(s, t) of t_i on s_j as the ratio of mutants of s_j killed by t_i to the total number of mutants of s_j. Note that if t_i does not execute s_j, this ratio is zero.

9. *Additional Fault- Exposing- Potential (FEP) prioritization*

As additions were made in total coverage and branch coverage prioritization, extensions were made in total fault-exposing potential prioritization and as a result this technique is created where we extend total FEP to create additional FEP prioritizations.

IV. PROPOSED WORK

Test case prioritization techniques arrange the test cases in some order, based on some criteria of prioritization, and therefore attempt to increase their effectiveness at meeting some performance goal. Prioritization does not remove any test case from test suite that is why prioritization is not a risky technique of regression testing.

We proposed a new technique of test case prioritization which results in earlier fault detection and named as *Extended additional fault exposing potential (FEP) prioritization*. For this, we first implemented a series of pre- existing 9 techniques.

Extended additional fault exposing potential (FEP) prioritization technique is the result of some modifications in additional fault exposing potential (FEP) prioritization. As in additional fault exposing potential (FEP), a term called *confidence* was used, in a very same way we also used this term. In this proposed technique, we use *C(s)*as a randomly generated value in our implementation. Let *s* denotes statement, *t* denotes test case and FEP (s, t) denotes faults in *s* covered by test case *t*, C(s) denotes confidence before execution of t and C'(s) denotes new confidence after execution of *t*. Therefore, after this change in the value of *C(s)*, equation here for the additional confidence in statement *s* becomes:

$$C_{addi}(s) = (1 - rand\,(C(s))).\ FEP\,(s, t) \qquad (1)$$

We can say that C_{addi} (t) can be defined as the additional confidence gained from executing *t* on *P* program. It can be calculated by summing up the value of C_{addi} (s) of all the statements s covered by *t*.

Proposed Algorithm

We here present a step by step algorithm for our proposed technique:

```
Input: Test cases $t_1, t_2, . . . , t_n$, Test suite T and generated mutants.
Output: Prioritized test suite T'.
Process:
Begin
  Set T' empty
     For each test case t € T do
  Calculate $C_{addi}$
     End for
     Sort T in descending order of $C_{addi}$ value
End
```

V. RESULTS AND DISCUSSIONS

This chapter discusses an evaluation result of the above prioritization technique. In this section, we present a graph that compares the above proposed technique of prioritization with a pre- existing test case prioritization technique. This comparison is based on the cumulative mutants killed by one of the each test case. In a graph, there are two dimensions: (a) on horizontal axis and (b) on vertical axis. The horizontal represents test cases taken in this experiment whereas the vertical represents the cumulative mutants killed by executing the test cases.

Figure 2 depicts the performance graph comparing the proposed prioritization technique to additional fault- exposing- potential (FEP) prioritization technique. The above graph shows that this new technique detected greater number of mutants than already existing technique of prioritization. Therefore the fault detection is improved and it takes less time to detect more mutants in comparison to other techniques.

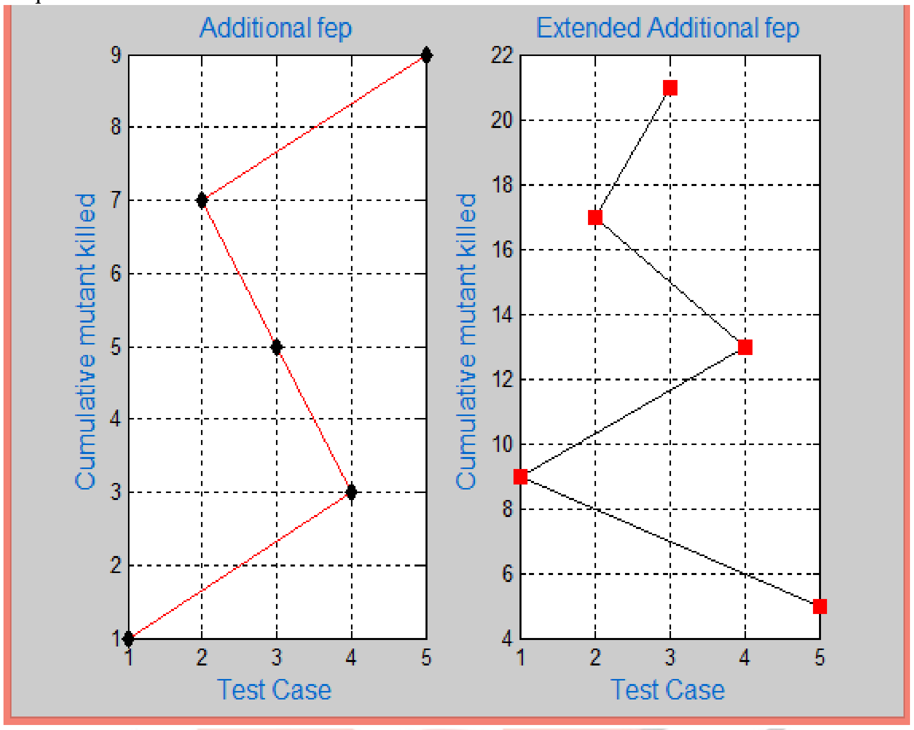

Figure 2. Performance graph for the comparison of two techniques

VI. CONCLUSION

The purpose of this paper is to introduce the techniques of regression testing and the criteria of prioritization technique. Regression Testing is retesting the unchanged parts of a program in order to ensure that: (a) despite the changes, the existing unchanged part of a program continues to function as desired; (b) its revision does not produce faults. Regression Testing improves correctness and locate errors. It is a costly process in software development. So, there is a need to minimize cost. Prioritization is one of a most important method of regression testing. It arranges the test cases present in a test suite based on different techniques of prioritization. So that tester can run only test cases which have higher priority. It results in consumption of cost and time. Therefore it is one of a most beneficial technique of regression testing. In this paper, a new technique for test case prioritization is proposed. This new technique is the result of some modifications made in additional fault- exposing- potential (FEP) prioritization. It is designed aiming at minimizing the time, effort and cost in software testing process.

REFERENCES

[1] Xuan Lin, "Regression Testing in Research And Practice", Computer Science and Engineering Department University of Nebraska, Lincoln, 2007.

[2] Ashima Singh , Kamal Parkash, "Fault Based Analysis to Perform Test Case Prioritization in Regression Testing", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 9, September 2012.

[3] 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.

[4] Last M., Eyal S., and Kandel A.,"Effective Black-Box Testing with Genetic Algorithms",2005.

[5] Neelam , Sukhvir Singh, "A Novel Fault Analysis Based Dynamic Programming Approach for Regression Testing", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 8, August 2013.

[6] Gaurav Duggal, Mrs. Bharti Suri, "Understanding Regression Testing Techniques", Proceedings of 2nd National Conference on Challenges and Opportunities in Information Technology, 2008, India.

[7] Rothermel G., Roland H.," Test Case Prioritization: An Empirical Study", International Conference on Software Maintenance, Oxford, UK, September, 1999, IEEE Copyright.

[8] Thillaikarasi Muthusamy, Dr. Seetharaman.K, "Comparisons of Test t Case Prioritization Algorithm with Random Prioritization", International Journal of Computer Science and Information Technologies, Vol. 5 (5) , 2014.

[9] https://www.cs.purdue.edu/homes/apm/foundationsBook/samples/regression-chapter-sample.pdf

[10] Ekta Khandelwal, Madhulika Bhadauria, "Various Techniques Used For Prioritization of Test Cases", *International Journal of Scientific and Research Publications*, Volume 3, Issue 6, June 2013.

[11] Sebastian E., Alexey G. Malishevsky, G. Rothermel, "Prioritizing Test Cases for Regression Testing", University of Nebraska-Lincoln, Computer Science and Engineering,pp. 102-112, August 2000.

[12] G. Rothermel, R.H. Untch, C. Chu, and M.J. Harrold, ªTest Case Prioritization: An Empirical Study", Proceedings of the International Conference on Software Maintenance, Oxford, UK, September, 1999, IEEE Copyright.

[13] Mathur Aditya P, "Foundations of Software testing", Pearson Education India,2008, pp. 336.

[14] W.Wong, J.Horgan, S.London, and H.Agrawal, "A study of effective regression testing in practice", In Proc. of the Eighth Intl. Symp. on Softw. Rel. Engr., pp. 230-238, Nov. 1997.

[15] http://en.wikipedia.org/wiki/Regression_testing.

[16] http://en.wikipedia.org/wiki/Test_suite.

[17] Roger S. Pressman , "Software Engineering: A Practitioner‟s Approach", 7/edition .