

IMPROVING SOFTWARE QUALITY THROUGH EFFECTIVE SOFTWARE TESTING

***B.Saravanan, **Dr. C.S.Ravichandran**

**Research Scholar, Dept. of CSE, Coimbatore Institute of Engineering and Technology, Coimbatore*

***Dean & Professor - PG, Dept. of EEE, Sri Ramakrishna Engineering College, Coimbatore*

ABSTRACT

In a typical Software Development Life Cycle (SDLC), Software testing phase is one of the important and critical phase. This is why because; the quality of the software is decided based on the amount of errors we find in the software testing phase. The more amount of error we find in the testing phase will bring good quality product. The fewer amounts of errors we find will bring quality degradation in the end product. Even though, many categories of testing available, regression testing are one of the most critical activities of software development and maintenance. Regression testing is basically re-executing or re-testing the modified software by using test cases. However, most of the time it is impractical for the software tester to re-execute all the test cases available in the test suite due to time constraints and resource constraints. This problem can be solved by using the prioritization of the test cases, so that the test case with the highest priority will be executed first than the lower priority test cases to meet some performance goals. In this paper, we have discussed various techniques used for prioritization of test cases so that we can save time and cost involved in the regression testing and thereby improving the software quality and increasing the customer satisfaction.

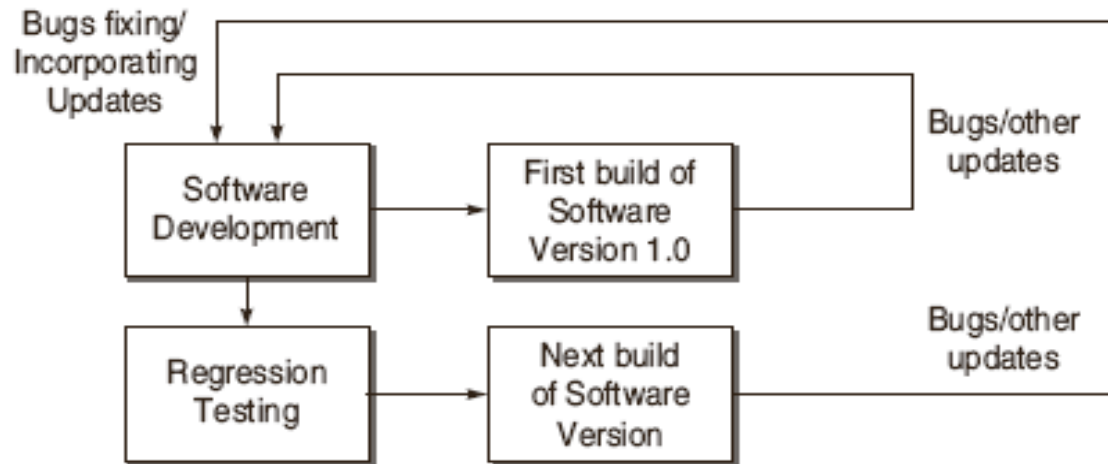
KEYWORDS: *Test cases, Test Cases Prioritization, Regression Testing, Test Case Selection, Test case Minimization, Software Quality.*

1. INTRODUCTION

The purpose of regression testing is to ensure that bug-fixes and new functionalities introduced in a new version of the software do not adversely affect the correct functionality inherited from the previous version. Regression testing is the selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirement as shown in figure 1.

The regression testing validates the parts of the software where changes occur, validates the parts of the software which may be affected by some changes, ensures proper functioning of the software enhances the quality of software.

Figure 1: Regression testing operation



The various regression testing techniques are test case minimization or test suite reduction technique, test case selection technique and test case prioritization technique. In this paper, we discuss various operation performed on the above techniques in order to achieve effective software testing and thereby increasing the software quality.

2. REGRESSION TESTING SELECTION TECHNIQUES:

Regression test selection attempt to reduce the cost of regression testing by selecting a subset of the test suite that was used during development and using that subset to test the modified program. With this approach only a subset of test cases are selected and rerun. Regression test selection divides the existing test suite into reusable test cases, re-testable test cases and obsolete test cases.

Retest-All Technique: This conventional method reruns all test cases in T that were previously run during testing phase and not a single test case is left. So the technique is very expensive as regression test suites are costly to execute in full as it require much time and budget and therefore it may be used when test effectiveness is the utmost priority with little regard for cost.

Random / Ad-Hoc Technique: In this technique, the testers rely on their previous experiences and knowledge to select which test cases need to be rerun. This can include selecting a percentage of test cases randomly.

Dataflow Techniques: This technique is coverage-based regression test selection technique that selects test cases that exercise data interactions that have been affected by modifications.

Safe Technique: This technique by definition eliminates only those test cases that are probably not

able to reveal faults. A safe technique, routinely selected almost all test cases when more than a few changes were made to the subject programs. This technique does not focus on criteria of coverage but select all those test cases that produce different output with a modified program as compared to its original version.

Hybrid Technique: Hybrid Approaches includes techniques of both Regression Test Selection and Test Case Prioritization or Regression Test Minimization and Test Case Prioritization.

3. MINIMIZATION TECHNIQUES:

In this technique a minimum number of test cases are selected from T , which helps the testers to uncover the modified elements of P' . This technique can include selecting those test cases that are related with those modified elements of the program.

Minimization can be defined as: A test suite, T , a set of test requirements $\{r_1, r_2, \dots, r_n\}$, that must be satisfied to provide the desired „adequate“ testing of the program, and subsets of T , T_1, T_2, \dots, T_n , one associated with each of the r_i s such that any one of the test cases t_j belonging to T_i can be used to achieve requirement r_i , find a representative set, T'' , comprised of test cases from T , that satisfies all r_i . Under this definition, T'' is equivalent to a hitting set for all T_i and a minimal T'' is equivalent to the minimal hitting set.

4. TEST CASE PRIORITIZATION:

Test case prioritization techniques schedule test cases in an execution order according to some criterion. The purpose of this prioritization is to increase the likelihood that if the test cases are used for regression testing in the given order, they will more closely meet some objective than they would if they were executed in some other order.

Test case prioritization can address a wide variety of objectives, including the following:

1. Testers may wish to increase the rate of fault detection – that is, the likelihood of revealing faults earlier in a run of regression tests.
2. Testers may wish to increase the rate of detection of high-risk faults, locating those faults earlier in the testing process.
3. Testers may wish to increase the likelihood of revealing regression errors related to specific code changes earlier in the regression testing process.
4. Testers may wish to increase their coverage of coverable code in the system under test at a faster rate.
5. Testers may wish to increase their confidence in the reliability of the system under test at a

faster rate.

The test case prioritization is classified into four categories:

- Statement coverage prioritization
- Additional statement coverage prioritization
- Branch coverage prioritization
- Additional branch coverage prioritization

4.1. STATEMENT COVERAGE PRIORITIZATION:

In this technique, we can prioritize test cases in terms of the total number of statements they cover by counting the number of statements covered by each test case and then sorting the test cases in descending order of that number. (When multiple test cases cover the same number of statements, an additional rule is necessary to order these test cases; we order them randomly or use First Come First Serve basis (FCFS)).

Table 1: Statement Coverage by test cases

Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
*		*		*
*		*		
*	*	*		
*	*	*		
	*	*	*	
	*	*	*	
	*	*	*	
*		*		*
*		*		*
		*		
			*	

When we apply the statement coverage technique for the table 1, the order of execution of the test cases will be 3, 1, 2, 4 and 5.

Table 2: Statement Coverage by test cases (Random Selection)

Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
*		*		*
*		*		
*	*	*		
*	*	*		
	*	*	*	
	*	*	*	
	*	*	*	
*		*		*
*		*		*
		*	*	
			*	

When we apply the statement coverage technique for table 2, the order of execution of the test cases will be 3, 1, 4, 2 and 5 (for random selection) and the order of execution will be 3, 1, 2, 4 and 5 (for FCFS basis).

Number of statements exercised

Percentage of statement coverage = ----- * 100

Total number of statement available

4.2. ADDITIONAL STATEMENT COVERAGE PRIORITIZATION:

In the additional statement coverage prioritization, first we select the test case which covers the maximum number of statements and then select the test case which covers additional statements which is not covered by the already selected test cases.

Table 3: Additional Statement Coverage by test cases

Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
*		*		
*		*		
*	*	*		
*	*	*		
	*	*	*	
	*	*	*	
	*	*	*	

*		*		
*		*		
		*	*	
			*	*

When we apply the additional statement coverage to table 3, the order of selection of execution of the test cases will be 3, 5, 1, 2 and 4.

4.3. BRANCH COVERAGE PRIORITIZATION:

Branch coverage prioritization is the same as statement coverage prioritization, except that it uses test coverage measured in terms of program branches rather than statements. In this context, we define branch coverage as coverage of each possible overall outcome of a condition in a predicate. Thus, for example, each if or while statement must be exercised such that it evaluates at least once to true and at least once to false. To accommodate functions that contain no branches, we treat each function entry as a branch, and regard that branch as covered by each test case that causes the function to be invoked.

Number of branches exercised

Percentage of branch coverage = ----- * 100

Total number of branches available

4.4. ADDITIONAL BRANCH COVERAGE PRIORITIZATION:

Additional branch coverage prioritization is the same as additional statement coverage prioritization, except that it uses test coverage measured in terms of program branches rather than statements.

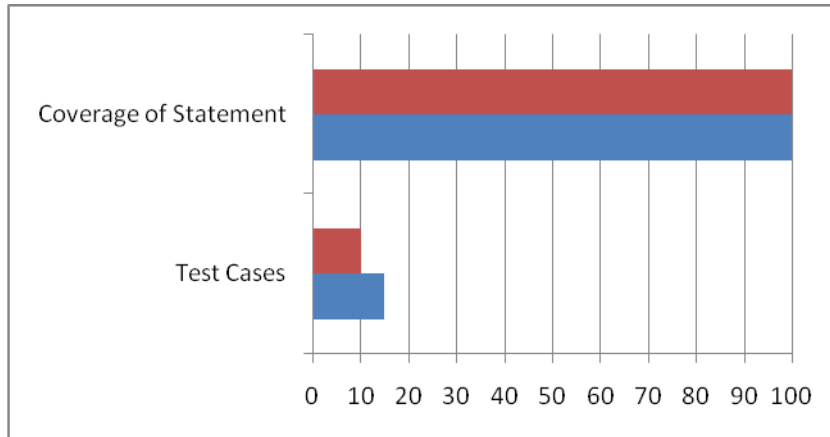
5. REDUCTION IN TEST SUITE:

Sometimes, if the time allocated for the testing process is limited, then it is enough to execute the test cases which perform 100% statement coverage and branch coverage or maximum coverage.

For example, when we consider the table 3, it is enough to execute the test case 3 and 5 to have the 100% statement coverage.

The comparison between the original size of the test suite and the reduced size of the test suite is specified in Figure 1. The result shows that there is a notable reduction in the size between the two test suites.

Figure 2: Test Suite Size and statement coverage



6. SOFTWARE TESTING VS SOFTWARE QUALITY:

The amount of time and effort spent on the software testing process will have a great impact in software quality. The software quality is defined as “Customer should come back and not the product”, “Satisfying the customer requirements” or “zero defects”. The more amount of errors found in software testing will drastically improve software quality and thereby increase the customer satisfaction. So, the prime responsibility of the software testing team is to write effective test cases, so that maximum amount of errors is captured before delivery of the product to the customer.

7. CONCLUSION:

Test case prioritization is a method to prioritize and schedule test cases. The technique is developed in order to run test cases of higher priority in order to minimize time, cost and effort during software testing phase and thereby increasing the quality of the software. This paper concentrates on the prioritization of test cases in regression testing based on the statement coverage and branch coverage.

Table 4: Results of a regression test cycle

Current result from regression	Previous results	Conclusion	Remarks
Fail	Pass	Fail	Need to improve the regression process and code reviews
Pass	Fail	Pass	This is the expected result of a good regression to say defect fixes work properly
Fail	Fail	Fail	Need to analyze why defect fixes are not working. “Is it a wrong fix?” Also should analyze why this test is rerun for

			regression
Pass	Pass	Pass	This pattern of results gives a comfort feeling that there are no side-effects due to defect fixes

8. FUTURE ENHANCEMENT:

In future, this paper can be extended to prioritize the test cases in the test suite with the help of APFD (Average Percentage Fault Detection). Test case prioritization can also be applied over requirement analysis using APFD and risk metrics. We can also extended focus to the following areas: practical prioritization weight values for commercial systems, improve the ability to automatically find duplicate test cases with the same values, improve the ability to automatically prioritize multiple large test suites with real commercial data.

9. REFERENCES:

- [1] Rothermel G, Harrold MJ. A safe, efficient algorithm for regression test selection. Proceedings of International Conference on Software Maintenance (ICSM 2003), IEEE Computer Society Press, 1993; 358–367.
- [2] Rothermel G, Harrold MJ. Analyzing regression test selection techniques. IEEE Transactions on Software Engineering August 1996; 22(8):529–551.
- [3] G. Rothermel, R. Untch, C. Chu, and M.J.Harrold, “Prioritizing Test Cases for Regression Testing,” IEEE Trans. Software Eng., vol. 27, no. 10, pp. 929-948, Oct. 2001.
- [4] Fischer K. A test case selection method for the validation of software maintenance modifications. Proceedings of International Computer Software and Applications Conference, IEEE Computer Society Press, 1977; 421–426.
- [5] Agrawal H, Horgan JR, Krauser EW, London SA. Incremental regression testing. Proceedings of the International Conference on Software Maintenance (ICSM 1993), IEEE Computer Society, 1993; 348–357.
- [6] Rothermel G, Harrold MJ. Selecting tests and identifying test coverage requirements for modified software. Proceedings of International Symposium on Software Testing and Analysis (ISSTA 1994), ACM Press, 1994; 169–184.
- [7] Rothermel G, Harrold MJ. A safe, efficient regression test selection technique. ACM Transactions on Software Engineering and Methodology April 1997; 6(2):173–210.
- [8] Rothermel G, Harrold MJ. Experience with regression test selection. Empirical Software Engineering: An International Journal 1997; 2(2):178–188.
- [9] Benedusi P, Cmitile A, De Carlini U. Post-maintenance testing based on path change analysis. Proceedings of the International Conference on Software Maintenance (ICSM 1988), IEEE Computer Society Press, 1988; 352–361.

- [10] Chen YF, Rosenblum D, Vo KP. Testtube: A system for selective regression testing. Proceedings of the 16th International Conference on Software Engineering (ICSE 1994), ACM Press,1994; 211–220.
- [11] S. Elbaum, A. Malishevsky, and G.Rothermel Test case prioritization: A family of empirical studies. IEEE Transactions on Software Engineering, February 2002.
- [12] Z. Li, M. Harman, and R. M. Hierons. Search Algorithms for Regression Test Case Prioritization, IEEE Transaction on Software Engineering, Vol.33, no. 4, PP. 225-237, 2007.

IJEST