Test Case Reduction For Regression Testing

Amit Srivastava, Er. Rajesh Tripathi

Abstract : Software testing is one of the most important stages of software development. In any software development, the development teams always depend on testing to know errors in the program. In the maintenance stage test suite size grow because of integration of new module in the main program. Addition of new module force to create new test case which increase the size of test suite. Regression testing is a type of testing in which we test the program after any modification in the program. In regression testing new test case may be added to the test suite during the whole testing process. The new additions of test cases create possibility of presence of same type of test cases. Due to limitation of time and resource, reduction techniques should be used to recognize and remove them.

Keywords: Regression testing, Test case, test case minimization.

1. Introduction

Regression testing is a testing method that is applied when a program is changed.It involves checking the changed program with some test cases so as to re-establishour confidence that the program can perform in line with the specification. In thedevelopment section, regression testing could begin after the detection and correctionof errors in a tested program. A tested program may be a program thathas been tested with a top quality test arrange. Regression testing may be a majorelement within the maintenance section wherever the software may be corrected,adapted to new setting, or increased to enhance its performance. Modifying a programinvolves making new logic to correct a mistake or to implement a modificationand incorporating that logic into an existing program.

1.1 Regression Testing

Regression testing [1,2] that is performed on changed programs to produce confidencethat modifications are correct and haven't adversely affected alternative parts ofthe program. A crucial distinction between regression testing and developmenttesting is that in regression testing we tend to sometimes have a longtime suiteof tests accessible for use. One regression testing strategy reruns all such tests;however this retest all approach might consume immoderate time and resources.Another, selective retest, chooses the checks from the recent test suite that aredeemed necessary to check the changed program. We tend to run these tests, soif necessary produce new ones, probably to satisfy some coverage criterion. Thisselective approach is useful given that the price of choosing the check set is asmaller amount than the price of running the tests we tend to are able to omit.

1.2 Types of Regression Testing

Two varieties of regression testing is known supported the modification of the specification.[2]

- *Mr. Amit Srivastava(CSED) amit.prince1983@gmail.com*
- *Er. Rajesh Tripathi(Associate Prof. - CSED) rajeshtcsed@mnnit.ac.in,rajesht63@rediffmail.com*
- *Motilal Nehru National Institute of Technology, Allahabad.*

Progressive Regression Testing:
It involves a changed specification. Whenevernew enhancements, or new knowledge necessities square measure incorporatedin an exceedingly system, the specification are changed to replicate theseadditions. In most cases, new modules are other to the software with the consequencethat the regression testing method involves testing a changed programagainst a changed specification.

Corrective Regression Testing: In corrective regression testing the specificationdoesn't modify, just some directions of the program and probablysome style choices square measure changed. This has necessary implications asa result of most test cases within the previous test arrange square measureseemingly to be valid within the sense that they properly specify the input-output relation. However, due to the modifications to the management and knowledgeflow structures of the software system, some existing test cases are notany longer testing the antecedent targeted program constructs. The corrective regressiontesting is usually done when some corrective action is performed on thesoftware system
.

1.3 Differences between Testing and Regression Testing[2]

Regression testing is not as normal testing we done in the testing phases. However,it is not always the case. There are so many difference between them.

Availability of test plan: Testing begins with a specification, an implementationof the specification and a test plan with test cases added during the specification,design and coding phases. All these test cases are new in the sense that they havenot been used to exercise the program previously. Regression testing starts with apossibly modified specification, a modified program and an old test plan which requiresupdating. All test cases in the test plan were previously run and were usefulin testing the program.

Objective of test: The main objective of the testing is to test the correctness ofthe program, the interconnection of the modules in the program and it gives the desireresult as the user wants. Regression testing is concerned with the modificationpart of the program, we totally focuses on that part of program which are modify;there is no need to test all the test cases.

Time allocation: Testing time is normally budgeted before the development of aproduct. This time can be as high as half the total product completion time. However,regression testing time, especially time for corrective regression testing, isnot normally included in the total product cost and schedule. Consequently, whenregression testing is done, it is nearly always performed in a crisis situation. Thetester is normally urged to complete retesting as soon as possible and most often isgiven limited time to retest.

Development information: In testing, the testing group and development groupmay be same, they know all about software. If the testing groups are different thenthe testers can usually query the developers about any problem in the software.But in regression testing, the testers most likely will not be the developers of theproduct. Since regression testing may be done at a different time and place, theoriginal developers may no longer be available. This situation suggests that anyrelevant development information should be retained if regression testing is to besuccessful.

Time duration: The time duration for testing should normally be more than thatfor regression testing since testing covers all parts of a program and regressiontesting covers only part of a program so testing needs more time.

1.4 Similarities between Testing and Regression Testing[2]

Several aspects of regression testing are almost like that of testingespecially,the needs and testing techniques used are virtually a similar.

Purposes: The purposes of testing and regression testing are quite similar. Theyboth aim to: 1) Increase ones confidence in the correctness of a program, and 2)Locate errors in a program.

Testing techniques:

Since test cases in a test set up rely on the chosen testing technique, the testingtechnique utilized by each testing and regression testing ought to be constant if theregression testing method involves the recycle of test cases. If regression testingwere to involve a unique testing technique, then it might be tough to recycle theprevailing check set up one more reason for victimization constant testing techniqueis that it's easier to gauge the standard of two computer code merchandiseif they're tested by constant technique. At the present state of the art, it's tough tomatch the relative check effectiveness of two totally different testing techniques.

2 Related Work

2.1 Basics Approach of Proposed Idea

The test case reduction can be achieve by coverage based method. Using coveragebased method we have a test case T' which is a subset of given test case T of aoriginal program P. T' can be used in the modified program P'. It covers all the theaspects of the modified program. Using T', one big advantage is that it save timeand resource. The following are the proposed algorithm for test case reduction:

minimizetestcase(testcase[][])

Step 1: Convert the given test case into binary 2D matrix if it is not given in binary form.

Step 2: Scan 2D matrix from first column to the last column.

Step 3: During first scan if the value of a particular cell is one then its relativerow value are added in the another 2D matrix m.

Step 4: During second to last scan if the cell value is one then we add that rowvalue to every row of 2D matrix m.

Step 5: Check the 2D matrix m row by row with the values (the values are testcase no.) using OR operation if the value is zero then result not found if value isone then we found the result.

Example: Consider a test suite with number of test cases and covering total10 faults. The test suites are T1, T2, T3, T4, T5, T6, T7,T8 and the faults areF1, F2, F3, F4, F5, F6, F7, F8,F9, F10. The T1 test case finds four faults i.e. F1,F3, F6,F9. So the binary form of T1 is 1010010010 we implement all the binarymatrix with the related test cases.

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

Sample data

TestCase	Binary Form	Fault Cover
T1	0100000100	2
T2	1010010010	4
T3	0001010110	4
T4	0100101000	3
T5	0000100100	2
T6	1000010001	3
T7	0010000001	2
T8	1000001100	3

Table 1: *Binary Matrix with fault covers*

Step 1:

In the first step we select those test cases which have first cell valueone, so we get T2, T6, and T8.

Test Case	Binary Form	Fault Cover
T2	1010010010	4
T6	1000010001	3
T8	1000001100	3

Table 2: *Reduced Matrix m*

Step 2:
In the next step we select those test cases which have second cell valueone,so we get T1 and T4. These test cases are added one by one in the ReducedMatrix m and its binary values are ORed with existing values in the reduced matrixm.

Test Case	Binary Form	Fault Cover
T2, T1	1110010110	6
T6, T1	1100010101	5
T8, T1	1100001100	4
T2, T4	1110110010	6
T6, T4	1100111001	6
T8, T4	1100101100	5

Table 3: *Reduced Matrix m*

Step 3:
In the next step we select those test cases which have third cell valueone, so we get T2, and T7. These test cases are added one by one in the ReducedMatrix m and its binary values are ORed with existing values in the reduced matrix m and also do not add test case on a row which have same teat case.

Test Case	Binary Form	Fault Cover
T2, T1	1110010110	6
T2, T4,	1110110010	6
T6, T1, T2	1110010111	7
T8, T1, T2	1110011110	7
T6, T4, T2	1110111011	8
T8, T4, T2	1110111110	8
T2, T1, T7	1110010111	7
T6, T1, T7	1110010101	6
T8, T1, T7	1110001101	6
T2, T4, T7	1110110011	7
T6, T4, T7	1110111001	7
T8, T4, T7	1110101101	7

Table 4: *Reduced Matrix m*

Step 4:
In the next step we select those test cases which have fourth cell valueone, so we get T3. These test cases are added one by one in the ReducedMatrix m and its binary values are ORed with existing values in theReduced matrixm and also do not add test case on a row which have same teat case. Here inthe following table T6, T4, T2, T3 has fault cover value 10 and also T6, T4, T7, T3has fault cover value 10 so we select only one from them which we get first that isT6, T4, T2, T3 the reduced test cases which covers all the faults.

Test Case	Binary Form	Fault Cover
T2, T1, T3	1111010110	7
T2, T4, T3	1111110110	8
T6, T1, T2, T3	1111010111	8
T8, T1, T2, T3	1111011110	8
T6, T4, T2, T3	1111111111	10
T8, T4, T2, T3	1111111110	9
T2, T1, T7, T3	1111010111	8
T6, T1, T7, T3	1111010111	8
T8, T1, T7, T3	1111001111	8
T2, T4, T7, T3	1111110111	9
T6, T4, T7, T3	1111111111	10
T8, T4, T7, T3	1111101111	9

Table 5: *Reduced Matrix m*

2.2 Minimization Coverage Based with Execution Time

When we include execution time in the table as a priority the minimized resultshould be different from the above result. There is no need to select result if morethan one result found.

Example: Consider a test suite with number of test cases and covering total10 faults. The test suites are T1, T2, T3, T4, T5, T6, T7, T8 and the faults are F1,F2, F3, F4, F5, F6, F7, F8 ,F9, F10 and Execution time. The T1 test case finds fourfaults i.e. F1, F3, F6, F9. So the binary form of T1 is 1010010010 we implementall the binary matrix with the related test cases.

Test Case	Binary Form	Fault Cover	Execution Time
T1	0100000100	2	7
T2	1010010010	4	3
T3	0001010110	4	5
T4	0100101000	3	5
T5	0000100100	2	3
T6	1000010001	3	6
T7	0010000001	2	3
T8	1000001100	3	2

Table 6: *Binary Matrix with fault covers & Execution Time*

Step 1:
In the first step we select those test cases which have first cell value one,so we get T2, T6, and T8.

Test Case	Binary Form	Fault Cover	Execution Time
T2	1010010010	4	3
T6	1000010001	3	6
T8	1000001100	3	2

Table 7: *Reduced Matrix m*

Step 2:
In the next step we select those test cases which have second cell valueone,so we get T1 and T4. These test cases are added one by one in the ReducedMatrix m and its binary values are ORed with existing values in the reduced matrixm. We also add the execution time with respect to the test cases.

Test Case	Binary Form	Fault Cover	Execution Time
T2, T1	1110010110	6	10
T6, T1	1100010101	5	13
T8, T1	1100001100	4	09
T2, T4	1110110010	6	8
T6, T4	1100111001	6	11

T8, T4	1100101100	5	07

Table 8: *Reduced Matrix m*

Step 3:
In the next step we select those test cases which have third cell valueone, so we get T2, and T7. These test cases are added one by one in the ReducedMatrix m and its binary values are ORed with existing values in the Reduced matrixm and also do not add test case on a row which have same teat case.

Test Case	Binary Form	Fault Cover	Execution Time
T2, T1	1110010110	6	10
T2, T4,	1110110010	6	08
T6, T1, T2	1110010111	7	15
T8, T1, T2	1110011110	7	12
T6, T4, T2	1110111011	8	14
T8, T4, T2	1110111110	8	10
T2, T1, T7	1110010111	7	08
T6, T1, T7	1110010101	6	15
T8, T1, T7	1110001101	6	12
T2, T4, T7	1110110011	7	14
T6, T4, T7	1110111001	7	10
T8, T4, T7	1110101101	7	08

Table 9: *Reduced Matrix m*

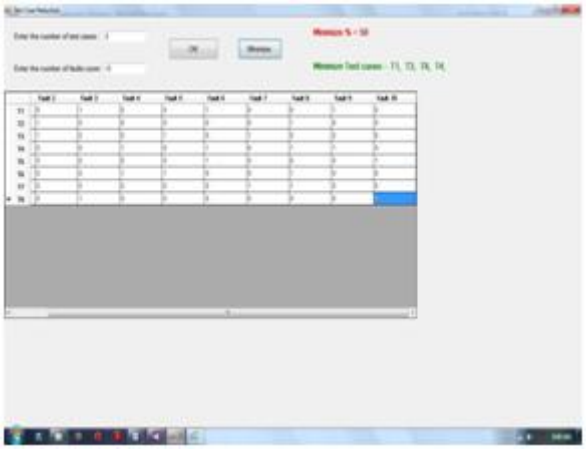
Step 4:
In the next step we select those test cases which have fourth cell valueone, so we get T4. These test cases are added one by one in the ReducedMatrix m and its binary values are ORed with existing values in the Reduced matrixm and also do not add test case on a row which have same test case. Here inthe following table T6, T4, T2, T3 has fault cover value 10 and total execution timeis 19 and also T6, T4, T7, T3 has fault cover value 10 and execution time is 15 sowe select T6, T4, T7, T3 as its total execution time is minimum.

Test Case	Binary Form	Fault Cover	Execution Time
T2, T1, T3	1111010110	7	15
T2, T4, T3	1111110110	8	13
T6, T1, T2, T3	1111010111	8	20
T8, T1, T2, T3	1111011110	8	17
T6, T4, T2, T3	1111111111	10	19
T8, T4, T2, T3	1111111110	9	15
T2, T1, T7, T3	1111010111	8	13
T6, T1, T7, T3	1111010111	8	20
T8, T1, T7, T3	1111001111	8	17
T2, T4, T7, T3	1111110111	9	19
T6, T4, T7, T3	1111111111	10	15
T8, T4, T7, T3	1111101111	9	13

Table 10: *Reduced Matrix m*

3 Results
The minimization result of the above test cases is T6, T4, T7, T3 which coversall the faults and total execution time is 15. This minimization technique is simple and get result faster as compareto the ”Test Suite Reduction using an Hybrid Technique Based on BCO AndGenetic Algorithm”.

4 Conclusion
The main objective of test case minimization is to save time and cost. For this we want to create minimization technique which gives minimized test case. The test case minimization by coverage based is very simple technique it only searches combination of test cases which covers all the faults. There is no need of analyze of modified code once minimization result found it will used many times without compare modified program with original program. The two step test case reduction, there is no need to choose result when more than one result found, it gives result on the basis of execution time. There is no need of details of the code in our approach but it takes argument as test case suite and gives the reduced code as an output.



6 References
[1] Rothermel, G.; Harrold, M.J., ”A safe, efficient algorithm for regression testselection,” Software Maintenance ,1993. CSM-93, Proceedings., Conferenceon , vol., no., pp.358,367, 27-30 Sep 1993

[2] Leung, H.K.N.; White, L., ”Insights into regression testing [software testing],”Software Maintenance, 1989., Proceedings., Conference on , vol., no.,pp.60,69, 16-19 Oct 1989.

[3] A test-suite reduction approach to improving fault-localization effectivenessGong DandanWangTiantian Su Xiaohong Ma Peijun School of Computer Scienceand Technology, Harbin Institute of Technology, Harbin 150001, ChinaComputer Languages Systems Structures (Impact Factor: 0.3). 01/2013;39(3):95108. DOI: 10.1016/j.cl.2013.04.001

[4] Mohapatra, S.K.; Prasad, S., ”Minimizing test cases to reduce the cost of regressiontesting,” Computing for Sustainable Global Development (INDIACom),2014 International Conference on , vol., no., pp.505,509, 5-7 March2014.

[5] Pan Liu, ”An efficient reduction approach to test suite,” Software Engineering,Artificial Intelligence,

Networking and Parallel/Distributed Computing(SNPD), 2014 15th IEEE/ACIS International Conference on , vol., no., pp.1,5,June 30 2014-July 2 2014.

[6] P. B. Sharma, Ruchika Malhotra and Mohit Garg Empirical Validation of anEfficient Test Data Generation Algorithm Based on Adequacy based TestingCriteria Software Engineering : An International Journal (SEIJ), Vol. 2, No. 1,March 2012.

[7] Test Case Prioritization based on Varying Testing Requirement Priorities andTest Case Costs by Xiaofang Zhang ChanghaiNieBaowen Xu Bo Qu.SeventhInternational Conference on Quality Software (QSIC 2007) 0-7695-3035-4/072007.

[8] DebasisMohapatra,GA based Test Case Generation Approach for Formationof Efficient Set of Dynamic Slices, International Journal on Computer Scienceand Engineering (IJCSE), Vol. 3 No. 8 August 2011.

[9] R. S. Pressman. Software Engineering: A Practitioners Approach , 3rd Edition,McGraw Hill, New York (1992), p. 559.

[10] Regression Test Suite Reduction using an Hybrid Technique Based on BCOAnd Genetic Algorithm, Bharti Suri, IshaMangal Varun Srivastava, International

[11] Journal of Computer Science Informatics (IJCSI), ISSN (PRINT) :22315292, Vol.- II, Issue-1, 2.