2012 International Conference on Computer and Software Modeling (ICCSM 2012)
IPCSIT vol. 54 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V54.01

A Personalized Approach for Code Coverage based Test Suite Selection

R.Beena [1], Dr.S.Sarala [2 +]

[1]Research Scholar, Dept. of Information Technology, Bharathiar University, Coimbatore.
[2]Assistant Professor, Dept. of Information Technology, Bharathiar University, Coimbatore.

Abstract. Regression testing is a very expensive technique which is performed to ensure that the modifications made to the existing software works correctly. The test suite maintained is usually very large and requires more cost and time to run all the test cases. So in order to decrease the cost of regression testing, the test cases in the test suite must be minimized. In this paper, a new customized approach is proposed that performs test case selection and prioritizes the selected test cases in order to achieve maximum code coverage.

Keywords: Test Suite Selection, Code Coverage

1. Introduction

Regression testing means retesting the program to ensure that the modified program works correctly. Regression Testing plays an important role in any Scenario where a change has been made to a previously tested software code. Regression Testing is hence an important aspect in various Software Methodologies where software changes enhancements occur frequently. Test cases are re-executed in order to check whether previous functionality of application is working fine and new changes have not introduced any new bugs.

Test suite selection means finding a smaller subset of test suite from the existing very large test suite [1]. According to [2], test suite selection problem is stated as below.

Given: The program, P, the modified version of P, P' and a test suite, T.

Aim: To find T' $\in$T, for the modified version P'

2. Related Work

Fischer et al. introduced a test case selection problem using Integer Programming [3]. In this approach the control flow changes were not dealt. Agrawal et al. proposed a test case selection techniques based on different program slicing techniques [4]. Rothermel and Harrold presented regression test case selection techniques based on graph walking of Control, Program Dependence Graphs [5], System Dependence Graphs [6] and Control Flow Graphs [7]. Benedusi et al. applied path analysis for test case selection [8]. Chen et al. introduced a testing framework called TestTube, which utilizes a modification-based technique to select test cases [9]. Leung and White introduced a firewall technique for regression testing of system integration [10]. Laski and Szemer presented a test case selection technique based on clusters [11].

3. Test Suite Selection

The existing test cases are clustered into three groups, namely out-dated, required, and surplus. The test cases that are no more required for the current version and the modified version are clustered into the out-

+ *E-mail address*: [1]beenamridula@yahoo.co.in;[2]sriohmau@yahoo.co.in

dated group. The test cases that are needed for the modified version P' is clustered into the required group. The test cases that are not required for the modified version P' but may be used later for new versions of P is clustered into surplus group.

Input:

- Matrix TCCij representing the test cases and their statements covered
- Vector SDELi representing the statements deleted in P'
- Vector SMODi representing the statements modified in P'

Output:

Modified Matrix TCCij, Cluster of Test Cases out_dated_i, $surplus_i$, $required_i$

Begin

1. for each statement that belongs to SDELi, remove the corresponding statements from TCCij.
2. find the sum of each row.
3. if (sum of the row == 0), then add the corresponding test case in the vector out_dated_i and remove it from TCCij.
4. find the test cases that do not cover the statements in the vector SMODi, add the corresponding test case in the vector $surplus_i$ and remove it from TCCij.
5. add the remaining test cases in the vector $required_i$.

End

4. Experiments and Results

The test cases and the statements covered by each test case (TCCij) are shown in table 1.

Table. 1

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
T1	1	0	1	1	1	0	1	1	0	1	1	0	0	0	0
T2	1	0	0	1	0	1	0	1	1	1	0	0	1	0	0
T3	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0
T4	0	1	0	1	0	0	1	1	1	0	0	0	1	1	0
T5	1	0	1	0	1	1	0	0	0	1	0	0	0	1	1
T6	0	1	0	1	0	0	1	0	1	1	1	1	0	0	0
T7	1	0	0	0	1	0	0	0	1	0	0	1	0	1	0
T8	0	1	1	0	0	1	0	0	1	0	1	1	0	0	0
T9	0	0	0	1	1	1	1	0	1	1	0	0	1	0	0
T10	1	1	0	0	0	0	1	1	1	0	0	0	1	1	1
T11	0	0	0	1	0	0	0	1	0	1	0	0	1	0	0
T12	1	1	0	0	1	0	1	0	0	0	0	0	0	0	1
T13	0	1	0	1	0	1	1	1	0	1	0	1	1	1	0
T14	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
T15	1	0	0	0	1	0	0	0	1	0	1	1	0	0	0

SDELi = { S3, S4, S6, S8, S10, S13 }

SMODi = { S2, S7, S15 }

After the execution of step 1, the matrix TCCij is modified by deleting the statements that are present in the vector SDELi. The modified TCCij is given in Table 2.

Table. 2

	S1	S2	S5	S7	S9	S11	S12	S14	S15
T1	1	0	1	1	0	1	0	0	0
T2	1	0	0	0	1	0	0	0	0
T3	0	0	0	0	0	0	0	0	0
T4	0	1	0	1	1	0	0	1	0
T5	1	0	1	0	0	0	0	1	1
T6	0	1	0	1	1	1	1	0	0
T7	1	0	1	0	1	0	1	1	0
T8	0	1	0	0	1	1	1	0	0
T9	0	0	1	1	1	0	0	0	0
T10	1	1	0	1	1	0	0	1	1
T11	0	0	0	0	0	0	0	0	0
T12	1	1	1	1	0	0	0	0	1
T13	0	1	0	1	0	0	1	1	0
T14	0	1	0	0	0	0	0	0	0
T15	1	0	1	0	1	1	1	0	0

In step 2, the sum of each row is calculated. It is given in table 3

Table. 3

Test Cases	Sum
T1	4
T2	2
T3	0
T4	4
T5	4
T6	5
T7	5
T8	4
T9	3
T10	6
T11	0
T12	5
T13	4
T14	1
T15	5

As in step 3, the test cases with the sum as zero are removed from the matrix TCCij. Now the new matrix TCCij is given in table 4. A new vector out_dated$_i$ is created.

Table. 4

	S1	S2	S5	S7	S9	S11	S12	S14	S15
T1	1	0	1	1	0	1	0	0	0
T2	1	0	0	0	1	0	0	0	0
T4	0	1	0	1	1	0	0	1	0
T5	1	0	1	0	0	0	0	1	1
T6	0	1	0	1	1	1	1	0	0
T7	1	0	1	0	1	0	1	1	0
T8	0	1	0	0	1	1	1	0	0
T9	0	0	1	1	1	0	0	0	0
T10	1	1	0	1	1	0	0	1	1
T12	1	1	1	1	0	0	0	0	1
T13	0	1	0	1	0	0	1	1	0
T14	0	1	0	0	0	0	0	0	0
T15	1	0	1	0	1	1	1	0	0

Out-datedi = { T3, T11 }

In step 4, the test cases from TCCij that do not cover the statements given in the vector SMODi are removed and those test cases are moved to the vector surplus$_i$. The new TCCij is given in table 5.

Table. 5

	S1	S2	S5	S7	S9	S11	S12	S14	S15
T1	1	0	1	1	0	1	0	0	0
T4	0	1	0	1	1	0	0	1	0
T5	1	0	1	0	0	0	0	1	1
T6	0	1	0	1	1	1	1	0	0
T8	0	1	0	0	1	1	1	0	0
T9	0	0	1	1	1	0	0	0	0
T10	1	1	0	1	1	0	0	1	1
T12	1	1	1	1	0	0	0	0	1
T13	0	1	0	1	0	0	1	1	0
T14	0	1	0	0	0	0	0	0	0

Surplusi = { T2, T7, T15 }

As in step 5, the remaining test cases are added to a new vector required$_i$.

required$_i$ = { T1, T4, T5, T6, T8, T9, T10, T12, T13, T14 }

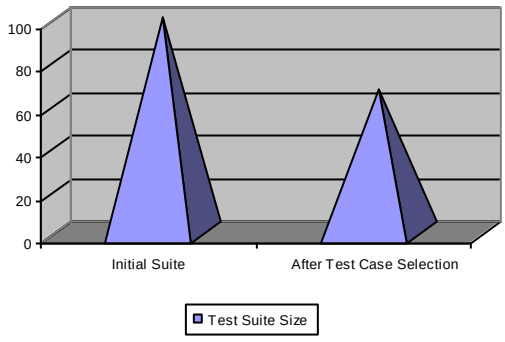

Fig. 1 Test Suite Size

After test suite selection process, the size of the original test suite is reduced. The comparison between the two test suites is represented in fig 1.

5. Conclusion and Future Work

Regression testing is carried out in the maintenance phase of the software development life cycle to retest the software for the modifications it has undergone. In this paper regressing testing based on test suite selection is proposed. Experimental results prove that the technique is effective for cost and time based prioritization. The future direction will be the combination of optimization algorithms with test suite selection technique.

6. 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] 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.

[4] 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.

[5] 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.

[6] Rothermel G, Harrold MJ. A safe, efficient regression test selection technique. ACM Transactions on Software Engineering and Methodology April 1997; 6(2):173–210.

[7] Rothermel G, Harrold MJ. Experience with regression test selection. Empirical Software Engineering: An International Journal 1997; 2(2):178–188.

[8] Benedusi P, Cmitile A, De Carlini U. Post-maintenance testing based on path change analysis. Proceedings of theInternational Conference on Software Maintenance (ICSM 1988), IEEE Computer Society Press, 1988; 352–361.

[9] 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.

[10] Leung HKN, White L. Insights into testing and regression testing global variables. Journal of Software Maintenance 1990; 2(4):209–222.

[11] Laski J, SzermerW. Identification of program modifications and its applications in software maintenance. Proceedings of the International Conference on Software Maintenance (ICSM 1992), IEEE Computer Society Press, 1992; 282–290.