Regression Test Selection for Object Oriented Systems Using OPDG and Slicing Technique

Vedpal
YMCA University of Science and Technology
Faridabad, India
Email Id: ved_ymca@yahoo.co.in

Naresh Chauhan
YMCA University of Science and Technology
Faridabad, India
Email Id: nareshchauhan19@gmail.com

Abstract – Regression testing is a selective retesting of software whenever software gets modified or some new functionality is added to it. In this paper a regression test case selection technique is proposed. This technique is based on identification of affected paths, affected functions and dynamic slicing which can be used to reduce the number of test cases for regression testing. This paper considers all three cases of modification to object oriented programs. The proposed approach is evaluated by showing the reduction in total number of test cases to be selected. For each program to be tested, this approach focuses on finding the affected paths, affected functions and on computing the dynamic slice of modified variables. In addition this approach is also open to combine a variety of available information for selection of test cases. For analysis it is applied to the software module in C++.

Keywords – Regression testing, test case reduction, object oriented testing .

I. INTRODUCTION

Regression testing is a repetitive testing, which has to be performed in case changes made to the existing software due to the various maintenance activities such as new components added to the software to add new functionalities, integration of different modules which have already been tested, bug fixing may introduce new bugs, etc. Regression testing has been used during the development and maintenance of a software product to assist software testing activities and guarantee the attainment of adequate quality through various versions of the software product [1].

The regression testing in object oriented systems proceeds at two levels: Application program testing and Class testing. .The regression testing is performed when the application program is modified or it uses modified class. The class level regression testing is done when a new class is added to the system (a class may be added through inheritance, association, composition etc), an existing class is modified, a class is deleted. In this case we want to find test cases from the original test suite that should be re-executed.

There are many factors given below which are not considered while selecting the test cases for class level regression testing in object-oriented software:

- Class level changes like modification in class.
- Major changes in the object oriented systems like addition of a function, deletion of functions in classes and adding the new class ,
- Deletion of classes.

In the light of above discussion, the objectives of the paper are:

- To utilize the limited resources (viz. cost, time, test tools, man power) in an efficient manner.
- To reduce the number of test cases by identifying the affected paths, affected functions due to modifications in the object oriented systems.
- To analyze and validate the proposed approach for test suite selection.

II. RELATED WORK

Software maintenance activities [2], on an average, account for as much as two-thirds of the overall software life cycle costs. Regression testing has been used during the development and maintenance of a software product to assist software testing activities and guarantee the attainment of adequate quality through various versions of the software product [3]. Regression testing permits to test modified software to provide confidence that there is no ripple effect of modification in previously tested code [4]. Test cases designed for original programs need to be tested again during regression testing and some new test cases need to be designed. Thus there will be a very large number of test cases, many of them are required to be re run. This includes extremely large overheads associated with regression testing.

Retesting [5] an entire system that has received very few changes is very expensive for large systems. With regression analysis and testing, we retest only the parts of software system affected by modifications. Classes [6] in an OO program can be structured as a hierarchy via inheritance relationships. In a class hierarchy, a class can either use the members inherited from its super classes without explicit

Authorized licensed use limited to: Universita degli Studi di Salerno. Downloaded on June 11,2025 at 17:30:59 UTC from IEEE Xplore. Restrictions apply.

declaration, or it can redefine them. An association shows connection betweenclasses. Objects communicate [7] each other through association.An aggregation [7] relationship between two classes means that an instance of one class is encapsulated as an attribute in the other class. Gregg Rothermel[8] et.al proposed a regression test selection technique that is based on analysis of both the source code of the object oriented program as well as the UML state machine models of the affected classes.

Yanping Chen [9] proposeda approach for identifying the affected classes when changed are made to an object oriented program was presented. To generate a suitable order for testing of the affected classes a algorithm was proposed. The basic model used in this approach is an object relation graph which shows the inheritance, aggregation and association that exist in object oriented software to be maintained.

Gregg Rothermel and Mary Jean Harrold [10] proposed a regression test selection techniquefor object–oriented software. This technique constructs graph representation for software and uses this graph to select test cases from the set of original test cases .the selected test cases execute the modified part for the new version of the software.

David Binkley [11] proposed a regression testing approach based on the program slicing. Program slicing is a useful tool for working on the incremental regression testing problem. Unlike older approaches which identify only directed affected components, such as du-pairs, slicing if two components have same execution behavior. Incremental regression testing attempts to find good approximate solutions to two unsolvable problems: determine the set of affected components and determining the set of tests that exercise these components.

Chabbi rani Panigrahi and Rajib Mall [12] presented a new technique to retest the object oriented software. An algorithm to construct dependence graphs for classes and application programs was proposed and these graphs are used to find out those test cases from the previous test suits which causes modified program or class to have different outputas compared to their original output.

Gregg Rothermel , Roland H.Untch , Chengyun Chu, Mary Jean Harrold In this[13] transformed a software architectures in to intermediate representation called architectures component dependence graph (ACDG). A slicing algorithm was presented which is based on marking and unmarking the in–service and out-service edges on an ACDG , dependencies arises and occurrence of events..

The types and relevance of faults are determined by the characteristics of object oriented software. A new technique for interclass testing is proposed. The proposed technique derives a suitable set of test case specifications for interclass testing by using data flow analysis.

OPDG [14] is a comprehensive representation that depicts object oriented software in a clear and concise manner. The representation is composed of three layers: these layers are Class Hierarchy Subgraph (CHS) , Control Dependency Subgraph (CDS) and Data Dependence Subgraph (DDS) The concept[15] of a program slice was first introduced by Weiser to add debugging of programs. A program slice is determined with the help of slicing criteria which comprises all the statements affected by slicing criteria. There are three types of slicing techniques. These technique are Execution Slicing, Dynamic Slicing, Relevant Slicing.

III. PROPOSED WORK

While performing the regression testing of object oriented systems the following changes are considered

1. Addition of a Class
2. Deletion of a Class
3. Modification of a Class
    - Addition of a function
    - Deletion of a function
    - Modification of a function

In all these cases we have to retest all the classes which are affected by relationships like inheritance, composition and association. This will produce a large test suite. Therefore, a technique is required that will reduce the test case so that only affected classes are tested. In this direction a technique has been proposed in this paper .An overview of proposed technique is shown in figure1.

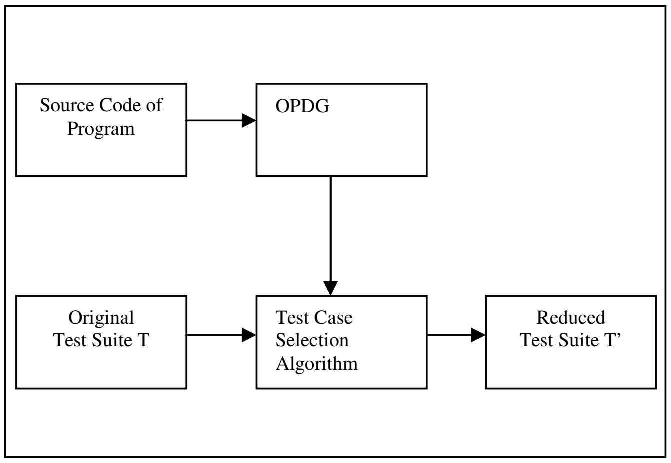

Fig. 1 An Overview of Proposed Technique

In this technique firstly an OPDG (Object Oriented Program Dependency Graphs) is constructed for the modified program. In case of addition of class to the OO system, affected paths by adding the new class are identified in the OPDG and marked. Then the test cases which execute the affected path are selected for regression testing. When the class is modified, then the affected functions and affected paths are identified. Then the dynamic slicing is applied to select those test cases whose output have been affected due to modification. In case of deletion of class different cases are there, a Class Hierarchy Subgraph (CHS) is constructed to identify the class by deleting which class the system will be invalid. Object oriented program dependency graph will be used to represent the object oriented programs whose regression testing is to be

Authorized licensed use limited to: Universita degli Studi di Salerno. Downloaded on June 11,2025 at 17:30:59 UTC from IEEE Xplore. Restrictions apply.

done. This representation is modular allowing various analysis techniques to only use the portion required for that analysis. Although this representation has three layers but in this approach only two layers CHS and CDS of OPDG will be used. CHS will be used to represent the inheritance hierarchy of classes and CDS is used to represent the control structures of various functions of classes.

A.Addition of Class

To accommodate the new requirements new classes may be added in the system. The added class may be linked to the existing classes in the system or may be an independent class in the system. If it is an independent class then there is no need of interclass testing. However if it is linked to the existing classes then interclass regression testing will play an important role.
A class may be a derived class to an existing class, or it may contain objects of other classes as its attributes i.e a composition relationship, or it may be linked by an association relationship. The idea behind test case selection is that a new class's methods might be calling the methods of old classes. Thus while performing interclass regression testing we don't need to test all the functions of old classes. Only the functions that are used by new class will be tested. The algorithm for selecting the test cases to test the affected classes is given below.

Fig.1.Algorithm1: Addition of class

```
INPUT: Source code of program;
        Original test suite T;
OUTPUT: Reduced test suite T';
Algorithm for test case selection is:
1. Make the OPDG graph of the given classes from the source code of program.
2. After the addition of new class make the interlinked OPDG graph.
3. Mark those edges in the OPDG graph where there is a function calling dependency
between two classes.
4. Select only those test cases from the test suite T for inclusion in T' for regression testing
which execute the marked edges.
```

B. Modification of Class

Sometimes modifications in classes have to be done to incorporate requirement changes. These modifications in classes can be done in a variety of ways: Addition of a function in class, Modification of a function in class, Deletion of a function in class.

C. Addition of a function in class

If a newly added function will be used by other functions in the class, then all those functions will be tested again. Also if that unction modifies the value of a variable. Then slicing technique will be used in which set of statements executed under a test case having an effect on the program output. For this dynamic slicing will be used and all those test cases will be selected for re-execution whose outputs have been affected by the use of that variable.

Fig.2Algorithm2: Addition of function in Class

```
INPUT: Source code of the program;
        Original test suite T;
OUTPUT: Reduced test suite T';
    1. From the source code of program make the OPDG of involved classes.
    2. Mark the new added function.
    3. If new added function modifies the definition of some variable, then trace all those statements in the program where a use of that variable has been made.
    4. Mark all the affected functions in the OPDG.
    5. Then the dynamic slice of that variable will be computed i.e all those functions will be marked whose output may be influenced due to modification.
    6. Else if it doesn't modify the value of any variable then trace those functions where that function is used (as a call to that function), then the function calling edges will be marked from that function.
    7. Select those test cases which executes the marked edges and marked functions.
```

D. Modification of function in class

This problem is called Fragile base class problem. Changing the super class can affect the subclass. Modification of super class can make the subclass invalid. However functions in the subclasses and other classes can be modified instead in the super class.

Fig.3Algorithm3:Modification of function in class

```
INPUT: Source code of the program;
        Original test suite T;
OUTPUT: Reduced test suite T';
Algorithm for test case selection is:
1. From the source code of program make the OPDG of involved classes.
2. Mark the statement in the OPDG which has been modified.
3. Also mark the function in the OPDG which has been modified.
4. Mark all the affected functions due to that modification in the OPDG.
5. If that function is used in other functions, then mark those function calling edges in the
   OPDG.
6. Compute the dynamic slice of the changed variable in the modified statement i.e mark
those functions whose output have been influenced due to modification.
7. Select those test cases for regression testing which executes those marked edges and marked functions.
```

Authorized licensed use limited to: Universita degli Studi di Salerno. Downloaded on June 11,2025 at 17:30:59 UTC from IEEE Xplore. Restrictions apply.

E. Deletion of a Function in Class

When a function is deleted then all function calls to that function will be invalid. All those functions which have used that function will be traced and those function calls will be invalid, then stubs have to be provided for those function calls.

F. Deletion of Class

The class is deleted when its requirement ceases to finish. There are only two cases of deletion of class. the cases are deletion of base class and deletion of derived class. the proposed algorithm for identify the effect of deletion of class is shown below.

Fig.4.Algorithm4:Deletion of Class

```
INPUT: Source code of the program;
        Original test suite T;
OUTPUT: Reduced test suite T';
Algorithm for test case selection is:
Begin
1. From the source code of program make the OPDG of involved classes.
2. If the base class deleted
    Then
3 A base class can't be deleted because then all its subclasses will become invalid.
4 If the requirement of base class finishes then base class should be changed to abstract class.
5. By changing that class into abstract class its subclasses will remain valid but application program can't instantiate that class.
6 If the derived class deleted
then
7. If the requirement of derived class finishes then a derived class can be deleted its super class will not have any effect of its deletion.
8. However if that class is used by other classes then those classes will become invalid.
9. Else if it is a independent class, then it can be deleted without any concerns.
Mark the statement in the OPDG which has been affected by deletion of class.
End
```

There are various cases of deletion of class are discussed below:

G. Deletion of Base Class

When the base class is deleted then all its subclasses will be invalid. In the figure 2 student is the base class or super class, test class is derived from the student class and, from the test class and sport class, result class is derived. When we delete the super class student, test class and result class will be invalid. However sport class continues to work, because it has no relation to the student class.

H. Deletion of Derived Class

When the derived class is deleted, then the base class will not have any effect. However if a derived class is used by another class and derived class is deleted, then that class will be invalid. In the figure 2 when we delete the class Result, links to Put_number and Put_marks will be deleted and it will have no effect on its super class student and test. However when the class test is deleted, class result will be invalid.When class sport is deleted its modify the display function of class Result.

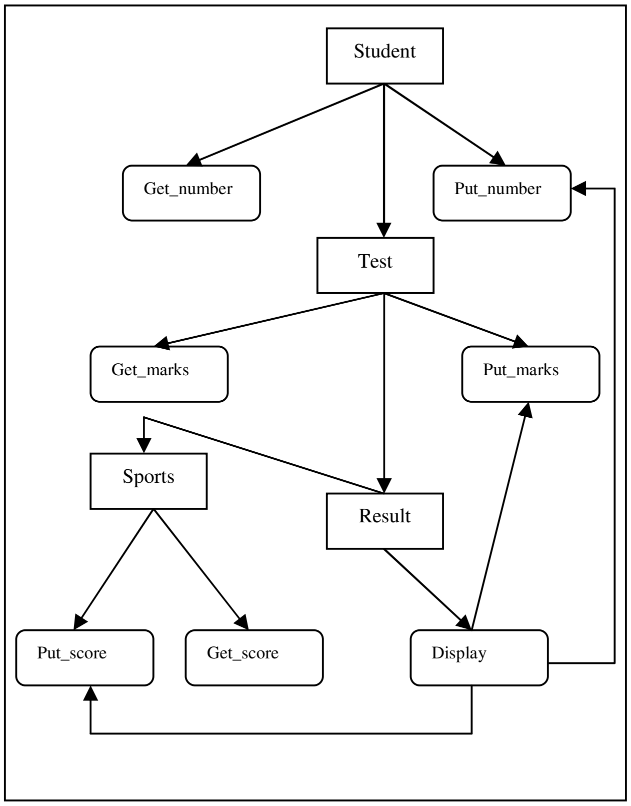

Fig.5. Deletion of a Class

IV. RESULT AND ANALYSIS

To analyze the effectiveness of proposed approach it has been applied to various software modules. Different modules of C++ are considered for each case i.e for addition of class, modification of class, deletion of class. The OPDG and test cases are designed for each modules. The proposed approach is applied and on the basis of affected paths test cases are selected.

A. Addition of Class

To demonstrate the reduction of test cases through the proposed approach, it is applied to a module1 that performs simple banking operations like depositing the amount, withdrawing the amount, computing the interest on deposited amount and displaying the balance. here for extending the functionality a new class SAccount is added which is subclass of account class

Authorized licensed use limited to: Universita degli Studi di Salerno. Downloaded on June 11,2025 at 17:30:59 UTC from IEEE Xplore. Restrictions apply.

B. Construction of OPDG

On adding the class SAccount through inheritance to the base class Account. An OPDG graph of considered program is constructed to identify the affected paths for interclass testing. The edges containing the affected paths in the OPDG graph are colored red for distinguishing those edges from other edges. The OPDG graph of this example is shown in the figure 3.

C. Designing of test cases

If interclass testing is to be performed.Total test cases to be run to test both classes on adding the class are 5 as shown in below table 1

TABLE I. TEST CASES DESIGN FOR ADDITION OF CLASS

Inputs	CustomerName	Atype	ch
TestCase 1	Ram	S	1
TestCase 2	Ram	S	2
TestCase 3	Ram	S	3
TestCase 4	Ram	S	4
TestCase 5	Ram	S	5

But if test cases are to be selected for interclass testing, while considering the affected path in the OPDG, only test case 3 and test case 4 executes the affected path. Test cases selected for execution are shown in the below table 2

TABLE II. TEST CASES SELECTED FOR ADDITION OF CLASS

Inputs	CustomerName	atype	Ch
TestCase 3	Ram	S	3
TestCase 4	Ram	S	4

So while performing the interclass testing on adding the class SAccount instead of re-executing the all test cases, only test case 3 and test case 4 need to be selected for re- execution.

D. Modification of Class

Modification of a class can be done in three ways. By adding a new function in the class, modifying the existing function anddeleting the function in class.

E. Modification of a function in the class

The software modules2 under consideration computes the simple interest and compound interest by taking the inputs of present value, rate and time. There is a function modify in the class interest, that performs the task of modifying the interest rates. This function has been modified.

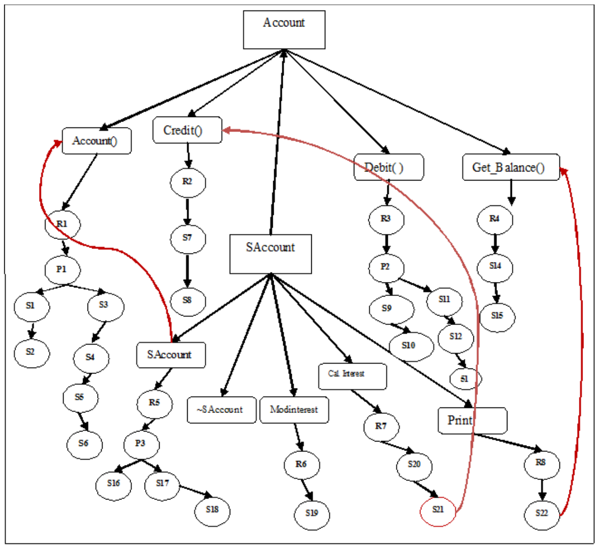

Fig. 6. OPDG for addition of class

F. Construction of OPDG

In the figure 4 given below OPDG graph of the modified program is constructed. The modified statement in the OPDGand other statements which have been affected due to modification are colored red.

G. Designing of test cases

The total number of test cases according to this application program are three. Although all the three test cases executes the modified function. But the TestCase1 and TestCase2 don't have any effect on their output.By applying the proposed approach, only TestCase3 executes the affected function i.e. Cal_interest of class CInterest . The statements executed byTestCase3 comes under the dynamic slice of the modified function. Total number of test cases is shown in the table 3

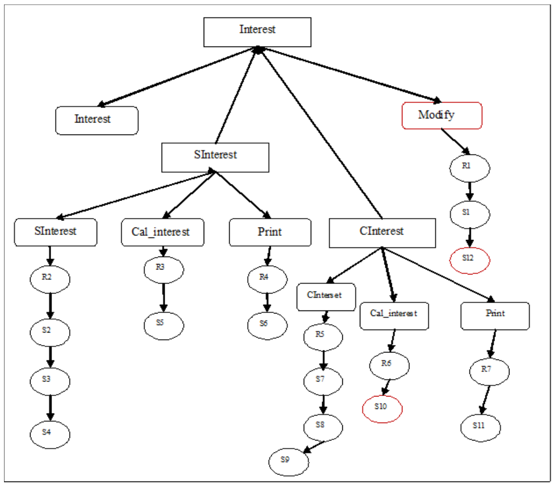

Fig.7. OPDG for modification of a function in a class

Authorized licensed use limited to: Universita degli Studi di Salerno. Downloaded on June 11,2025 at 17:30:59 UTC from IEEE Xplore. Restrictions apply.

TABLE III. OUTPUT OF TEST CASES DESIGNED BEFORE THE MODIFICATION OF A FUNCTION IN A CLASS

Inputs	P	R	Cr	t	Output
TestCase1	1500	1.5	1.5	0	Executed modify
TestCase2	1500	1.5	1.5	1	Executed modify SimpleInterest is:29.25
TestCase3	1500	1.5	1.5	2	Executed modify CompoundInterest is:2142.03

TABLE IV. OUTPUT OF TEST CASES AFTER THE MODIFICATION OF A FUNCTION IN A CLASS

Inputs	P	R	Cr	t	Output
TestCase1	1500	1.5	1.5	0	Executed modify
TestCase2	1500	1.5	1.5	1	Executed modify SimpleInterest is:29.25
TestCase3	1500	1.5	1.5	2	Executed modify CompoundInterest is:2142.03

Test case selected for re-execution is shown in the table 5

TABLE.V.TEST CASE SELECTED FOR RE EXECUTION FOR MODIFICATION OF A FUNCTION

Inputs	P	R	Cr	T	Output
TestCase3	1500	1.5	1.5	2	Executed modify comp interst CompoundInterest is:2142.03

The dynamic slice of the variable with respect to modified variable 'cr' comes under the TestCase3.

H. Addition of function in Class

To demonstrate the addition of function in class module1 has been taken. In classSAccount a new function getbonus has been added. This function getbonus() computes the bonus. This function makes a call to the function credit. And this function also modifies the value of a variable called annualInterestRate. So affected path will be identified and also slicing will be done based on variable annualInterestRate. So all those functions will be identified where a use of variable annualInterestRate has been done and output have been affected in that function due to modification.The variable annualInterestRate which is given new definition in the function getbonus() is used in the functions Cal_interest and Mod_interest. But only the function Cal_interest() need to be tested in combination with getbonus() , because in the functionCal_interest, the modified variable is used to compute the interest, thus output have been affected.Thus function Cal_interest comes under the dynamic slice of new function. But in the function Mod_interest only a new definition is given to that variable, so no need to run that function in combination with Mod_interest.

I. Construction of OPDG

The OPDG graph of the module1 is constructed in the figure 5. The new added function getbonus() in the class SAccount is colored red. Also the statement in the new function which can have effect on the other functions is also colored red.

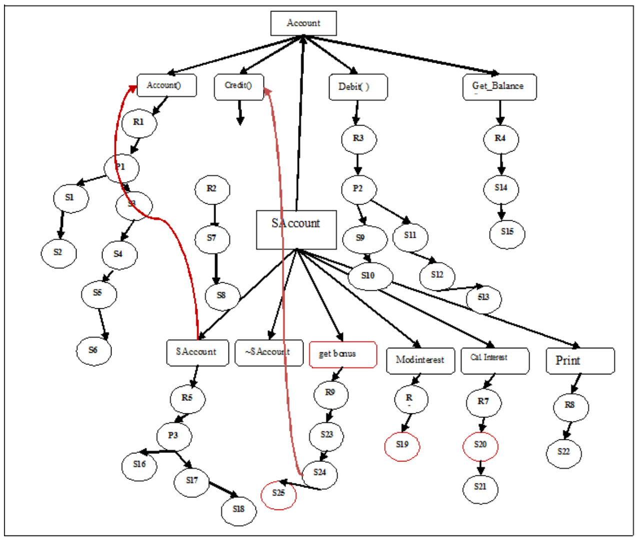

Fig.8.OPDG for Addition of function in Class

I. Designing of test cases

Total test cases run to test the program are shown in the table 6. theTestCase 4 and TestCase 5 executed the affected function or class of the software module.

TABLE VI. TEST CASES DESIGNED FOR ADDITION OF A FUNCTION IN A CLASS

Inputs	savingBalance	Acname	Acno	Atype	Ir	ch
TestCase 1	500	Ram	1261	S	1.5	1
TestCase 2	2000	Ram	1262	S	1.5	2
TestCase 3	4000	Ram	1231	S	1.6	3
TestCase 4	5000	Ram	1263	S	1.5	4
TestCase 5	2000	Ram	1261	S	3.5	5
TestCase 6	3000	Ram	1261	S	1.4	6
TestCase 7	3000	Ram	1264	S	1.2	7

Test cases that are selected that are used to test the modified software module are shown below in the table 7

Authorized licensed use limited to: Universita degli Studi di Salerno. Downloaded on June 11,2025 at 17:30:59 UTC from IEEE Xplore. Restrictions apply.

TABLE VII. TEST CASE SELECTED FOR RE-EXECUTION FOR ADDITION OF A FUNCTION IN A CLASS

Inputs	savingBalance	Acname	Acno	Atype	Ir	ch
TestCase 4	5000	Ram	1263	S	1.5	4
TestCase 5	2000	Ram	1261	S	3.5	5

J. Analysis of Proposed approach

The table 8 shows that there is significant percentage of reduction of test cases. The percentage of reduction of test cases shows the effectiveness of proposed approach

TABLE .VIII ANALYSIS OF PROPOSED APPROACH

S.No.	**Programe Name**	**No. Of test cases**	**No. Of selected test cases**	**% of reduction of test cases**
1	Addition of class	5	2	60%
2	Modification of function in a class	3	1	66.6%
3	Addition of a function in a class	7	2	71.42 %

V. CONCLUSION

In this paper an approach for regression testing has been proposed to select test cases from the original test pool. This approach considers all major modifications to the object oriented software like addition of class, modification of class by adding a function, modifying a function, deleting a function, deletion of class. To analyze the effectiveness of this approach it is applied to different software module of C++ under different cases of modification. By applying the proposed approach it is found that numbers of test cases are reduced. The result shows the efficacy of proposed approach in terms of reduced number of test cases.

REFERENCES

[1] Hyuncheol Park, HoyeonRyu, JongmoonBaik, " Historical Value-Based Approach for Costcognizant Test Case Prioritization to improve the Effectiveness of Regression Testing".

[2] Swarnendu Biswas and Rajib Mall, "Regression Test Selection Techniques: A Survey" Dept. Of Computer Science and Engineering, IIT Kharagpur, ManoranjanSatpathy and ShrihariSukumarn, GM Science Lab, Banglore, India.

[3] Mary Jean Harrold.,"Testing a roadmap". In Proceedings of the Conference on the Future of Software Engineering, ACM Press 2000.

[4] MaruanKhoury, "Cost-Effective Regression Testing", 2006.

[5] Agrawal H, Horgan J.R. Krauser E. W. And Landon S. A, " Incremental Regression Testing", Proceedings of the IEEE Conference on Software Maintenance, Montreal, Canada, 1993.

[6] Siros Supavita,**"**Object-Oriented Software and UML-Based Testing: A Survey Report", http://cc.ee.ntu.edu.tw/

[7] Jiun-Liang Chen and Feng-Jian Wang "Flow Analysis of Class Relationships for Object-Oriented Programs" Journal of information science and engineering 16, 619 – 647(2000)

[8] Gregg Rothermel, Roland H. Untch, Chengyun Chu, and Mary Jean Harrold. "Test case prioritization: An empirical study". In Proceedings of the International Conference on Software Maintenance, Oxford, September 1999.

[9] Yanping Chen "Specification based Regression Testing Measurement with Risk Analysis" .

[10] Gregg Rothermel and Mary Jean Harrold, "Selecting Regression Tests for Object Oriented Software" Department of Computer Science, Clemenson University.

[11] David Binkley "The Application of Program Slicing to Regression Testing" , Loyola College in Maryland.

[12] Chabbi rani Panigrahi and Rajib Mall, " A Hybrid Regression Test Selection Technique for Object Oriented Programs" , International journal of Software Engineering and Its Applications vol.6, No. 4, October, 2012.

[13] Gregg Rothermel (University of Nerbaska- Lincolen), Roland H.Untch (Middle Tennessee State University), Chengyun Chu (Microsoft Inc.), Mary Jean Harrold(College of Computing, Georgia Institute of Technology)**"**Prioritizing Test Cases for Regression Testing",.

[14] John D. McGregor, Brian A. Malloy, Rebecca L. Siegmud"A Comprehensive Program Representation of Object Oriented Software", Dept of Computer Science**,** Clemson University, Clemson

[15] Naresh Chauhan, "Software Testing Principles and Practices", Oxford University Press, 2010

[16] Henry Muccini, Marcio Dias, Debra J. Richardson, "Software Architecture-based Regression Testing", Journal of Systems and Software, Vol. 79, No 10, pp 1379-1396, October 2006.

[17] Rothermel and M Harrold. Analyzing regression test selection techniques. *IEEE Transactions on Software Engineering*, 22(8): 529-551, August 1996.

[18] Gregory M. Kapfhammer, "Software testing", In Allan B. Tucker, editor, The Computer Science Handbook. CRC Press, Boca Raton, FL, second edition, June 2004.

[19] David C Kung, Jerry Gao, Pei Hsia**"**Class Firewall, Test Order and Regression Testing of Object-Oriented Programs", Department of Computer Science and Engineering, The University of Texas at Arlington.

[20] Gregg Rothermel (Dept of Computer Science, Oregon State University), Mary Jean Harrold (College of Computing, Georgia Institute of Technology), JeinayDedhia (Dept. of Computer Science, Oregon State University).**"**Regression testing for C++ software"

[21] Mei – Hwa Chen and Howard M. Kao "Regression Testing on object oriented Programs" Ye Wu , staff.unak.is

Authorized licensed use limited to: Universita degli Studi di Salerno. Downloaded on June 11,2025 at 17:30:59 UTC from IEEE Xplore. Restrictions apply.