Study and Analysis of Regression Test Case Selection Techniques

Sunidhi Puri
Department of Computer Science & Engineering
Amity University,Noida,U.P

Abhishek Singhal
Department of Computer Science & Engineering
Amity University,Noida,U.P

Abhay Bansal
Department of Computer Science & Engineering
Amity University,Noida,U.P

ABSTRACT
The activity of re-testing of only those parts of the program or code, in which some modifications are performed to ensure that errors have not been added and the changes do not affect the other parts of the code, which have not been modified is called as regression testing. Regression testing is essential as it reduces the size of the test suite, thus reducing the time and effort for testing. In this paper, different techniques for the regression test case selection for various programming paradigms are discussed.

General Terms
Software Engineering, Regression Testing, Test case selection

Keywords
Regression testing, re-testing, errors, test cases, test case selection, test suite

1. INTRODUCTION
The activity of re-testing of only those parts of the program or code, in which some modifications are performed to ensure that errors have not been added and the other parts are not affected, which have not been modified is called as regression testing [1]. The concept of test case selection was introduced so that the original test suite size is reduced and thus reducing the cost of testing process. When any change is introduced into software, all those parts, which are also changed due to this are discovered and the cases which validate those changes are selected and thus minimizes the regression testing time and effort.

Maintenance plays a vital role in software development life cycle. It approximately costs 60% of the total life cycle cost [2]. It is required to ensure the proper functioning of the software. When some modification is made to the program code, it is required to re-test those areas. Regression testing is used for this purpose and it is also called as program revalidation. It is usually done at the system level.

There are different techniques for the regression test case selection for various programming languages. In the later sections, the techniques for procedural and object-oriented programming languages are discussed.

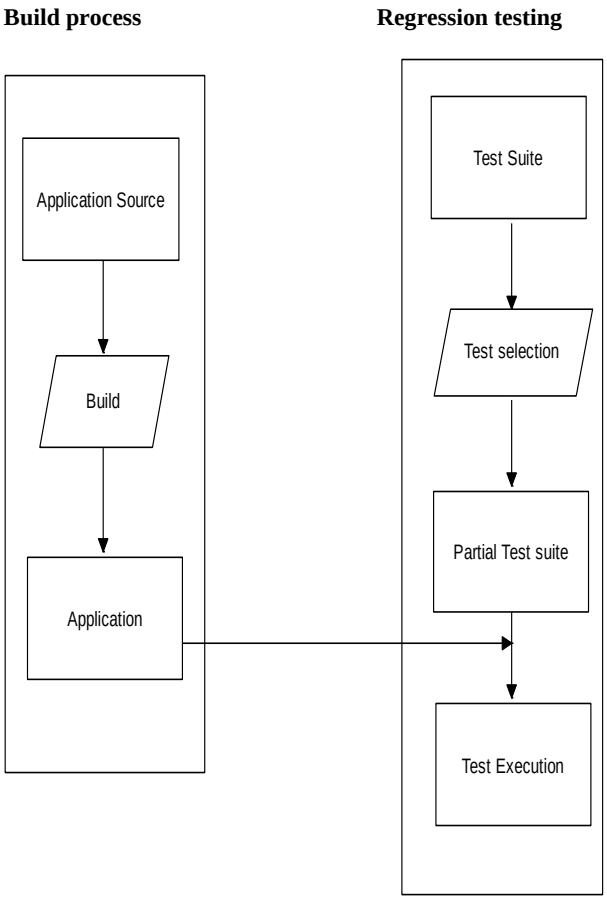

Figure 1: Regression Testing

There are a few concepts related to the regression testing. These are as follows:-

Obsolete Test Case: It refers to those test cases which are no longer justifiable for a modified code [5].

Re-testable Test Case: It refers to a test case which executes those parts of the code which are either modified or affected by the change in the original program. During regression testing, these cases are required to be rerun [5].

Redundant Test Case: It refers to a test case which executes those code areas which are not changed. These are not considered in the test suite generated for regression testing [1].

Execution Trace of Test Case: It refers to set of statements executed when a program is validated by a test case. It is

denoted by ET (P (t)), t refers to any test case and P refers to a program [1].

Fault-revealing Test Case: It refers to that test case which produces wrong outputs and causes the program to fail [4].

Modification-revealing Test Case: It refers to that test case which produces different outputs for the new modified and the original programs [4].

Modification-traversing Test Case: It refers to that test case which produces different execution traces for the new modified and the original programs [4]. It executes only modified parts of the program.

Inclusive, Precise and Safe Regression Test Case: Inclusiveness refers to the scope of selection of a modification-revealing case from the original suite of test cases [4].

It is measured as-

$$X=(m/n) * 100 \quad ,n\neq 0 \quad [4]$$

Here X represents inclusiveness, n represents the total modification revealing cases in the original suite of test cases T, m represents the number of cases which are selected from n by any technique.

A technique which is 100% inclusive is called as safe. There can be some test cases which are relevant but are ignored by a technique. These are called as false negatives. A technique is safe if it does not have any false negatives.

The extent to which a test case selection algorithm for regression testing ignores cases which is non-modification revealing is called as precision [4]. The cases which are not valid for a modification and are not ignored are false positives. A test case selection technique for regression testing is said to be precise if the selected test cases do not contain any false positives.

In the following sections, various terminologies are discussed like those related to graphs, which include flow graphs, control flow, data dependence, control dependence, program dependence and system dependence graphs; selection techniques for procedural programming language which includes techniques based on dataflow, firewall, differencing and control flow analysis, selection techniques for object-oriented programming language, which includes techniques based on firewall, design model and specification are discussed.

2. GRAPH MODELS

2.1 Flow Graph: It refers to a directed graph in which nodes represent the statements in the code and the edges joining those nodes depict the relation between them. There exist at least two nodes in a flow graph called as start and stop.

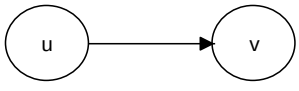

Figure 2: Flow Graph

2.2 Control Flow Graph: It captures the flow of control within a program code. Such graphs assist testers in the analysis of a program to understand its behaviour in terms of the flow of control. CFG can be constructed manually without much difficulty for relatively small programs, say containing less than about 50 statements. However, as the size of the program grows, so does the difficulty of constructing its CFG increases and hence arises the need for tools.

2.3 Data Dependence Graph: Let D be a DDG with nodes n1 and n2. Node n2 is data dependent on n1 when (a) definition of v variable is at n1 and its usage is at n2 and (b) there exists a path of nonzero length from n1 to n2 not containing any node that redefines v.

A DDG for program P contains one unique node for each statement in P. Declaration statements are omitted when they do not lead to the initialization of variables. Each node in a DDG is labeled by the text of the statement as in a CFG or numbered corresponding to the program statement.

2.4 Control Dependence Graph: Let C be a CDG with nodes n1 and n2, n1 being a predicate node. Node n2 is control dependent on n1 if there is at least one path from n1 to program exit that includes n2 and at least one path from n1 to program exit that excludes n2.

As with data dependence, control dependence can be visually represented as a control-dependence graph (CDG). Each program statement corresponds to a unique node in the CDG. A directed edge exists from n2 to n1 when n2 is control dependent on n1.

2.5 Program Dependence Graph: A PDG for program exhibits different kinds of dependencies among statements in P. For the purpose of testing, data dependence and control dependence are considered. These two dependencies are defined with respect to data and predicates in a program. Next, data and control dependences are explained, how they are derived from a program and their representation in the form of a PDG. Firstly, how to construct a PDG for programs with no procedures is explained and after that, how to handle programs with procedures is explained.

2.6 System Dependence Graph: To counter the limitations of the PDG which could model only single procedure, the enhanced form of program dependence graph was introduced which could deal with procedure calls. Firstly, a program dependence graph is created and then all the dependencies between various procedures are added to create a system dependence graph.

3. REGRESSION TEST CASE SELECTION TECHNIQUES FOR PROCEDURAL PROGRAMS

3.1 Dataflow Analysis-Based Technique:

The technique considers the definition-use pairs. The definition-use pairs which get affected due to modification in the program code are taken into account and those cases that validate these changed definition-use pairs are selected. The uses have also been divided into computation and predicate uses, i.e. c-uses and p-uses. A c-use has a direct effect on the computations whereas indirect on the control flow. A p-use has a direct effect on the control flow whereas it may have an indirect effect on the computations.

A technique was proposed by Harrold and Soffa [16] through which the changes introduced among various multiple procedures can be analyzed. First, the dataflow information is processed in an incremental fashion in which a single change is processed. Then the test cases which validate this change are selected. In the next step, the information related to dataflow and the test coverage is updated. In this approach,

CFG is used to represent a program in which each node represents a block of statements [1].

3.2 Module Level Firewall-Based Technique:

The technique was proposed by Leung and White [10]. The data and control dependencies are considered in this technique. All the modules which are either modified or affected by the change are taken into a firewall. The flow of control is represented using a call graph. If a path exists from module A to module B in the call graph, then A is an ancestor module of module B and B is the descendant. In a firewall, the direct ancestors and descendants are also included. Selection of only those test cases is done which validate modules within the firewall.

3.3 Differencing-Based Technique:

The techniques which consider the differences between the code before modifications and the one after modifications are known as differencing-based techniques. It involves (i) Modified Code Entity-Based Technique (ii) Technique Based on Textual Differencing

3.3.1 Modified Code Entity-Based Technique:

Chen et al [13] proposed this technique. There can be a directly executable or non-executable code entity. A directly executable code entity includes a function whereas any global variable is a non-executable code entity. First, the original code is tested by using all the cases. When a program is modified, it is checked for the code entities if any change is introduced. Then all those cases which validate the modified code entities are selected.

3.3.2 Technique Based on Textual Differencing:

A technique based on textual differencing in the code was proposed by Vokolos and Frankl [17].It includes trivial differences between a code and its modified form. This could include blank lines, comments etc. Therefore, the program is converted into its canonical form which ensures both the programs follow similar guidelines. After executing the canonical version of the original program, test case coverage is identified. After that the syntax of both the programs is analysed to check for the changes. Then the cases are selected which validate these changes introduced in the program code.

3.4 Control Flow Analysis-Based Technique:

Rothermel and Harrold [18] proposed this technique which considers control flow graphs. The purpose behind this technique is to traverse the control flow graphs of the original code as well as of the changed code, and identifying the changes. Firstly, for all the test cases, the execution traces are generated. Then, the graphs for both the programs are traversed in a depth-first manner in accordance with the execution traces generated. The execution trace of each test case generated for both the software versions are compared. For example, if the statements corresponding to nodes a and a', where a represents the node in the original code and a' represents the node in the modified code, are different, then the edges linking these nodes are termed as dangerous edges and all those cases which exercise these edges are selected.

4. REGRESSION TEST CASE SELECTION TECHNIQUES FOR OBJECT-ORIENTED PROGRAMS

4.1 Firewall-Based Techniques:

It follows the concept provided by Leung and White [10] for the procedural programming language. When a program is modified, all those classes which are affected by the modification are identified and included in a firewall. So, the cases which validate at least one class in the defined firewall are selected for regression testing [1].

4.1.1 Kung's Class Firewall Technique:

It was proposed by Kung et al. [9] for the software using C++ programming language. ORD, BBD and OSD, i.e. Object Relation Diagram, Block Branch Diagram and Object State Diagram respectively, can be used to show the dependencies between different program elements. The inheritance, association and aggregation relationships are represented through an ORD. It also represents the static dependencies between different classes. The type of relation between two nodes is depicted through the edge between them. For a method of a class, the interface and control structure can be represented through a BBD. The relation with other classes can also be depicted. Through an OSD, the dynamic behaviour of a class can be depicted.

The first step in this technique is to gather information about which test case validates which class. If changes are introduced in a class, say C, then all those classes which are directly or indirectly affected by this, along with the class C are put into a firewall. Then all those test cases which validate any of the classes present in the firewall are selected.

Below is an example of object relation diagram in which a class D is modified. A firewall is created which is shown through the dashed line in which D,A,B and C classes are included which indicates that whenever changes are made to D, classes A,B,C are also affected through that change and so, need to be tested again. The relationship between two classes is shown through a solid arrow. In the figure, which class is validated through which test case is represented through a solid line. Therefore, whenever D is modified, TC1 and TC2 are needed to be exercised only [8].

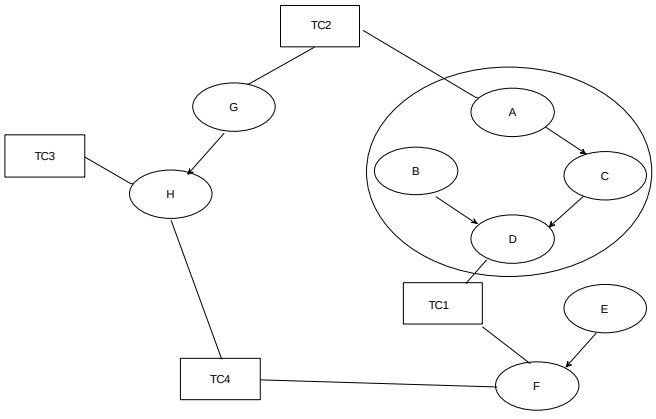

Figure 3: Example showing firewall for D class

4.1.2 Method-level Firewall Technique:

It was proposed by Jang et al. [19] for C++ programming language. In this technique, instead of classes, methods are considered, i.e. whenever some change is introduced to a

method, all the methods which are also affected by this change are introduced into a firewall. Then, all those cases which validate at least one of the methods from the firewall are selected.

4.2 Design Model-Based Technique:

This has gained much popularity. Its importance has been increased due to the advancement in the paradigm of MDD, i.e. Model Driven Development. In MDD, a code can be easily obtained from a design model. Thus, these techniques can be used for regression test case selection.

For object-oriented programming, the design models can be represented through UML, i.e. Unified Modeling Language. The advantages of model-based techniques [6] include the following:-

i. Traceability can be easily maintained between the design models and the cases as compared to that between a code and a test case.

ii. In case some modifications are done in the software, then it is easier to identify those changes through a design model rather than identifying changes in the program code.

iii. In case of large program codes, the cost of regression testing can be very high if a code-based technique is applied. So, this technique is more efficient.

iv. It provides language independent solutions.

4.2.1 Technique based on Class and Sequence diagrams:

A technique was proposed by Ali et al. [20] which make use of class and sequence diagrams. The technique analyses the sequence diagrams in detail and CCFG, i.e. Concurrent Control Flow Graph is designed. A CFG is not efficient in this case since the Concurrent Control Flow Graphs model the concurrency, if it exists, in the sequence diagram and this is done by using parallel instructions and asynchronous messages which are not possible in case of a control flow graph [1].

Then, the class diagrams are considered so that information can be extracted and included into CCFG. In this way, a ECCFG, i.e. Extended Concurrent Control Flow Graph is created by analyzing the two diagrams. When modifications are made to software, the ECCFGs of both the versions, the original one and the modified one are considered and analysis is done, and then based on this analysis, the test cases which validate the changes introduced into the software are selected.

4.2.2 Class and State Diagram based technique:

This technique was proposed by Farooq et al. [21] which make use of class and state diagrams for the test case selection used for regression testing [1]. If any modification is introduced in the code, the class and the state diagram also tend to change. So, with the help of these diagrams, it can be easily find out which elements are affected due to the change introduced. Then, the cases which exercise the changed transitions are selected [2].

4.2.3 UML Architectural and Design model-based technique:

This technique was proposed by Briand et al. [22]. In this technique, the traceability between program code, design model and test cases is obtained.

When any modification is introduced in the software, the changes can be easily identified through the design model and hence, test cases exercising those affected areas can be selected. Sequence, class and use case diagrams are used in this technique. The technique also distinguishes the test cases into three types- re-testable, reusable and obsolete.

4.3 Specification-Based Technique:

This technique was introduced since there was a drawback with the design models. The model or the code-based analysis cannot be used as the testers may not be provided with the source code or the design models. So, in such cases, specification-based technique for regression testing is more suitable. This technique was developed by the researchers which are based on specifications, generally available to testers.

The technique is based on activity diagrams and was proposed by Chen et al. [23]. It models the requirements which are affected due to introduction of modifications and also the system behaviour. The test cases are distinguished into two types- target cases and the safety cases. The cases which validate the affected requirements attributes are called as target cases. The cases that are selected to reach the predefined coverage target are called as safety targets. These are incorporated on the basis of risk analysis [3].

There are a number of steps required to select target test cases. The first step involves creation of a traceability matrix. Traceability specifies which requirement attribute is exercised by which test case. If a program code is modified, the specifications may change. In the next step, the activity diagram is traversed and all the nodes and edges affected due to modification in the program are recognised. Then in the next step, all those test cases which validate those edges are selected by using the traceability matrix created. These test cases are known as target test cases.

Then, safety test cases are selected which also involves a few steps. The first step is calculation of the cost of each case. The next step involves the calculation of severity probability which is achieved by multiplying total defects and the average severity of defects. The next step involves the calculation of risk exposure which is done by the multiplication of cost and severity probability. The result of this is considered to be the risk. The last and the final step involve the selection of those test cases which have a higher value of risk [2].

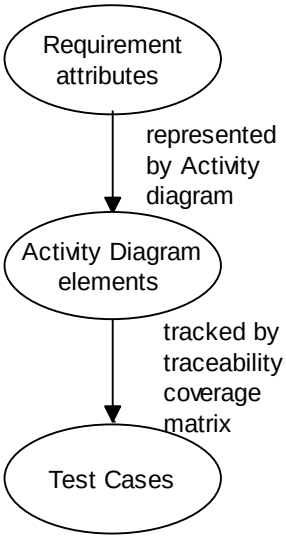

Figure 4: Traceability between requirement attributes and test cases

5. COMPARATIVE STUDY OF THE TECHNIQUES FOR PROCEDURAL PROGRAMS

Table 1: Comparison between techniques

Techniques	Key Features	Merits	Demerits
Dataflow analysis-based [1][16]	Based on dataflow in a program	Analyze inter-procedural modifications also but these should alter the definition-use pairs	If dataflow information is unaffected, do not analyze effect of modification
Module level firewall-based [10]	Based on the analysis of dependencies between modules	Comparatively more efficient as analysis is limited to modified modules only	Test cases that execute affected modules from outside the firewall are not selected
Modified code entity-based [13]	Based on analyzing affected code entities	Analyze all affected code entities	Test case may execute function without executing modified code
Textual differencing-based [17]	Based on textual differencing	Easy to implement	Inefficient for large programs
Graph walk-based [18]	Control flow graph analysis	Most precise	High computation effort required

6. COMPARATIVE STUDY OF THE TECHNIQUES FOR OBJECT-ORIENTED PROGRAMS

Table 2: Comparison between techniques

Techniques	Key Features	Merits	Demerits
Firewall-based [1] [8][9] [10] [19]	Based on the analysis of dependencies between modules	Efficient as analysis is limited to modified modules only	Test cases that execute affected modules from outside the firewall are not selected
Design model-based [1] [2] [6] [20] [21] [22]	Analyzes various UML design models	Suitable for large programs	Less precise than detailed code analysis
Specification-based [2] [3] [23]	Based on analysis of requirements, traceability between specifications and test cases	More efficient, platform independent	Precision depends on accuracy of requirement coverage

7. CONCLUSION

Regression testing is essential as it reduces the test suite size, thus reducing the time and effort for testing. Various techniques for procedural and object-oriented programming languages regarding the selection of test cases for regression testing have been discussed. For procedural programming language, it is observed that the technique which is the most precise is graph walk-based technique which is based on the analysis of control flow graphs. In case of object-oriented programming language, it is observed that various techniques were proposed but none of the techniques were precise as they do not work on fine granularity level. So, the techniques which analyze the modifications at the program statement level are more accurate.

8. REFERENCES

[1] Biswas, Swarnendu, Rajib Mall, Manoranjan Satpathy, and Srihari Sukumaran. "Regression Test Selection Techniques: A Survey." Informatica 35.3 (2011).

[2] Bharati, Chandana, and Shradha Verma. "Analysis of Different Regression Testing Approaches." Analysis 2.5 (2013).

[3] Chen, Yanping, Robert L. Probert, and D. Paul Sims. "Specification-based regression test selection with risk analysis." Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research. IBM Press, 2002.

[4] Panigrahi, Chhabi Rani, and Rajib Mall. "A Hybrid Regression Test Selection Technique for Object-Oriented Programs." International Journal of Software Engineering & Its Applications 6.4 (2012).

[5] Iqbal, Muhammad Zohaib Z., Zafar I. Malik, and Aamer Nadeem. "An approach for selective state machine based regression testing." Proceedings of the 3rd international workshop on Advances in model-based testing. ACM, 2007.

[6] Farooq, Qua, et al. "A model-based regression testing approach for evolving software systems with flexible tool support." Engineering of Computer Based Systems (ECBS), 2010 17th IEEE International Conference and Workshops on. IEEE, 2010. 22

[7] Vincent, Pierre-Luc, Linda Badri, and Mourad Badri. "Regression Testing of Object-Oriented Software: Towards a Hybrid Technique." International Journal of Software Engineering & Its Applications 7.4 (2013).

[8] Skoglund, Mats, and Per Runeson. "A case study of the class firewall regression test selection technique on a large scale distributed software system." Empirical Software Engineering, 2005. 2005 International Symposium on. IEEE, 2005.

[9] D. Kung, J. Gao, P. Hsia, F. Wen, Y. Toyoshima, and C. Chen. On regression testing of object-oriented programs. Journal of Systems and Software 32.1 (1996).

[10] H. Leung and L. White. A firewall concept for both control-flow and data-flow in regression integration testing. In Proceedings of the Conference on Software Maintenance,1992.

[11] H. Leung and L. White. Insights into regression testing. In Proceedings of the Conference on Software Maintenance,1989.

[12] G. Rothermel and M. Harrold. Analyzing regression test selection techniques. IEEE Transactions on Software Engineering 22.8 (1996).

[13] Y. Chen, D. Rosenblum, and K. Vo. TestTube: A system for selective regression testing. In Proceedings of the 16th International Conference on Software Engineering,1994.

[14] J. Ferrante, K. Ottenstein, and J. Warren."The program dependence graph and its use in optimization." ACM Transactions on Programming Languages and Systems 9.3 (1987). 23

[15] S. Horwitz, T. Reps, and D. Binkley. Interprocedural slicing using dependence graphs. ACM Transactions on Programming Languages and Systems 12.1 (1990).

[16] M. Harrold and M. Soffa. An incremental approach to unit testing during maintenance. In Proceedings of the International Conference on Software Maintenance,1988.

[17] P. Frankl, G. Rothermel, K. Sayre, and F. Vokolos. An empirical comparison of two safe regression test selection techniques. In ISESE '03 Proceedings of the 2003 International Symposium on Empirical Software Engineering. IEEE Computer Society, 2003.

[18] G. Rothermel and M. Harrold. A safe, efficient regression test selection technique. ACM Transactions on Software Engineering and Methodology 6.2 (1997).

[19] Y. Jang, M. Munro, and Y. Kwon. An improved method of selecting regression tests for C++ programs. Journal of Software Maintenance: Research and Practice 13.5 (2001).

[20] A. Ali, A. Nadeem, Z. Iqbal, and M. Usman. Regression testing based on UML design models. In Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing, 2007.

[21] Q. Farooq, M. Iqbal, Z. Malik, and M. Riebisch. A model-based regression testing approach for evolving software systems with flexible tool support. In 17th IEEE International Conference on Engineering of Computer-Based Systems (ECBS). IEEE Computer Society, 2010.

[22] L. Briand, Y. Labiche, and S. He. Automating regression test selection based on UML designs. Information and Software Technology 51.1 (2009). 24.

[23] Y. Chen, R. Probert, and D. Sims. Specification-based regression test selection with risk analysis. In CASCON '02: Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research, 2002.