www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 3 March,2014 Page No. 5078-5082

EXPLORATORY STUDY FOR REGRESSION TEST SELECTION TECHNIQUES IN PROCEDURAL AND OBJECT ORIENTED PROGRAMMING PARADIGM

Priyanka Rana[1], Dr. Anita Ganpati[2]

[1]Computer Science Department,
St Bede's College, Shimla, India.
Email-priyankarana.id@gmail.com

[2]Computer Science Department,
Himachal Pradesh University, Shimla, India
Email-anitaganpati@gmail.com

Abstract: The most crucial phase in the software development life cycle is maintenance phase. Whether it is corrective, adaptive or perfective maintenance it must be ensured that the modification does not affect other portions of the program. Therefore testing is required. Regression testing is carried out after a software developer has attempted to fix a problem or has introduced source code to software. An important research problem in regression testing is the selection of a suitable subset of test cases from existing test suite that would reduce the regression testing time and effort without giving up the completeness of regression testing. Researchers have suggested a number of regression test selection (RTS) techniques for different programming paradigms. In this paper an exploratory study of regression test selection techniques was carried out for procedural and object-oriented programming paradigm. From the study it was concluded that Specification Based RTS Technique is most effective as it does not depend on model-based or code based analysis.

Keywords: maintenance, regression testing, selection, procedural, object oriented

1. Introduction

Software maintenance activity consumes 40-70% of the cost of the entire software life cycle. Software Maintenance is required for error correction, enhancement of existing functionalities, and deletion of obsolete capabilities. Whenever software is changed for enforcement of maintenance activity, regression testing is performed to retest the modified part of the software and ensuring that no new errors have been introduced into the previously tested code.

A successful test results in the discovery of errors and the errors must be corrected. Whenever software is corrected, some part of software configuration is changed. Regression testing is the activity that helps to ensure that changes do not introduce unintended behavior or additional errors [13].

A number of different approaches have been studied to aid the regression testing process. The three major branches include test suite minimization, test case selection and test case prioritization. Test suite minimization is a process that seeks to identify and then eliminate the obsolete or redundant test cases from the test suite. Test case selection deals with the problem of selecting a subset of test cases that will be used to test the changed parts of the software. Finally, test case prioritization concerns the identification of the 'ideal' ordering of test cases that maximizes early fault detection. This paper presents the various types of regression test selection techniques proposed by various researchers, their classifications in procedural and object-oriented applications.

2. Regression test selection (RTS)

Various RTS techniques have been proposed by researchers that select subset of test cases from an initial test suite to test the affected but unchanged parts of a program. Effective techniques for regression test selection can be helpful in reducing the testing costs in environments where software undergoes repeated changes. Regression test selection consists of two main steps. The first steps identify unchanged parts of the software that are affected by the modifications. The second step identifies subset of test cases from the initial test suite which can test the unchanged parts of the software.

Various RTS techniques have been described for procedural and object-oriented programs. Each technique tries to reduce the cardinality of a test suite. The problem of RTS has been greatly investigated in the resent past; various techniques have emerged with the newer programming paradigms.

2.1. RTS Techniques for Procedural Programs

RTS for procedural programs is vastly researched topic therefore various techniques are proposed in the past. The techniques proposed for procedural programming paradigm includes, Data flow analysis techniques which gathers information about the possible set of values calculated at various points in a program. A program's control flow graph (CFG) is used to determine those parts of a program to which a particular value assigned to a variable might propagate. Program slicing technique decomposes programs by analyzing their data and control flow. Roughly speaking, a program slice

consists of those program statements which are related to the values computed at some program point and/or variable, referred to as a slicing criterion. Firewall based technique; a firewall is an imaginary boundary that limits the amount of retesting of the modules which are affected by a change. Differencing Based Techniques is analysis the differences for the original and modified software. Control flow analysis technique analyzes the input programs for their control flow models while selecting regression test cases.

2.1.1. Dataflow Analysis-Based Techniques

Harrold and Soffa [11] suggested a dataflow coverage-based RTS technique which can analyze changes occurring in multiple procedures. In this approach the dataflow information is processed, each change is processed by selecting test cases for that change and accordingly the dataflow information and test coverage information are updated. The process is iterated for all the modifications one by one. In their approach CFG is used, in which the nodes represent basic functional blocks. Reduced size of the flow graph is achieved; this makes graph analysis more efficient when compared with substituting individual program statements as nodes.

2.1.2. Slicing-Based Techniques

Bates and Horwitz [14] suggested a PDG-based slicing based technique. Binkley [2] proposal for inter-procedural RTS technique was based on slicing SDG models of unmodified and modified program. If the components executed same number of times for any given input, the two components are said to carry same execution patterns [2]. The technique of common execution patterns [2] has been introduced as an extension in the patterns suggested in [14]. Code elements are said to have a same execution pattern when they have same execution pattern during a call made to a procedure. Same execution patterns identify the semantic differences between code elements [2]. The semantic differences among unmodified and modified program are determined by comparing the modified version of the two programs. The modified versions of the programs are tested to determine affected program part which requires being regression tested.

2.1.3. Firewall-Based Techniques

Leung and White [6] were the first to suggest firewall based technique, which analysis data and control dependencies between modules for procedural program. A firewall is a set of modified modules of program and modules which interact with changed modules. This technique uses a call graph to identify control flow structure for a program [6].

2.1.4. Differencing Based Techniques

Chen et al. [17] suggest code entity-based RTS technique. The program elements are decomposed into functional and non-functional code entities. Code entities are identified as function/ statement or non-executable parts such as global variable or a macro. The unmodified program is executed with each test case of original test suite. When the original program is modified, all the code entities changed to create the modified program are identified. Test cases that exercise any of the modified entities are selected for regression testing.

2.1.5. Control Flow Analysis-Based Techniques

Laski and Szermer [8] suggested a cluster identification technique; the program modification is localized into code areas known as clusters. The Cluster identification based approach uses control dependence information of the unmodified and the modified procedures to compute the clusters in the two graphs. When clusters have been defined as CFGs, each cluster is displayed as single node to form a reduced CFG. Reduced flow graphs are analyzed assuming that complex program changes can be achieved by following one of the three operations: inserting a cluster into the code, deleting a cluster, or changing the functionality of a cluster. Test cases are defined as local to the clusters and global to the entire program. The former includes test cases which execute modified clusters, and the latter includes test cases which execute other areas of the program affected due to the changed clusters based on control dependencies. The information of test coverage is used for regression test selection.

Rothermel and Harrold[4] have suggested RTS technique which traversal the CFGs for the unmodified and the modified programs . This technique involves constructing separate CFGs for unmodified and modified programs. The execution information for each test case is recorded. Modification revealing test cases are assumed execute the set of identified dangerous edges. Therefore, a test case is selected for retesting modified program.

Ball [16] has suggested RTS technique by modeling CFG for unmodified program as a deterministic finite state automaton (DFA). The technique introduces intersection graph model for a pair of unmodified and modified CFGs. Edge coverage criterion is used as the basis for RTS analysis.

Figure: 1 summarizes the RTS techniques for procedural programming paradigm.

Regression Test Selection Technique for Procedural Programming Paradigm

RTS Techniques	Characteristics	Advantages
Dataflow analysis based technique	Based on dataflow and structural coverage criteria	Can analyze intra and inter procedural modification
Slicing based technique	Based on slicing of programs or dependency graph model	Can analyze both intra and inter procedural modifications
Firewall based techniques	Based on analyzing dependencies among modules	Only modified modules of the source code are efficiently analyzed
Differencing based technique	Based on textual differencing of C program	Safe and easy to implement
Control flow analysis based technique	Based on analysis of control flow models	Safe and precise procedural RTS technique

Figure: 1

2.2. RTS Techniques for Object-Oriented Programs

Object-oriented programming is an approach to designing modular, reusable software systems. These modules regularly undergo modifications to remove errors and enhance the functionalities; therefore effective regression test cases selection is required. Various RTS techniques for object-

oriented programs are categories. Firewall based RTS techniques, determines the affected classes for the changed version of the software. A firewall is defined to be a set of all the affected classes that need to be retested. These techniques select all test cases which exercise at least one class from within the firewall. Program Model Based Technique analyzes program models for selecting regression test cases. Design model-based techniques have become very popular with the advent of the model-driven development paradigm. In the model-driven development (MDD) paradigm, a design model is usually refined to obtain the code. Specification-Based RTS Techniques eliminated practical difficulty in RTS i.e. the testers may not have access to the source code of software under test. In such case, model-based or code based testing is not possible. Therefore researchers developed RTS techniques which are based on specifications that are made available to the testers.

2.2.1. Firewall-Based Techniques

Kung et al. [3] suggested a firewall-based RTS technique. Three models were proposed to show dependencies between various elements of a C++ program:

Object Relation Diagram (ORD): shows inheritance, aggregation and association relations, and captures the static dependencies among classes. An edge in an ORD is that it defines the type of relationship (inheritance, association, aggregation) that exists between the ends nodes associated with that edge.

Block Branch Diagram (BBD): shows the interface and the control structure for a method of the class, and the relationship for the class with the other classes in the program.

Object State Diagram (OSD): is designed to determine the dynamic behavior of the class.

Revision to data items, methods and class definitions of the original program are identified by analyzing the three models related to unmodified and modified program. When a class C is modified, the technique selects all the test cases that exercise one or more classes within the firewall for Class.

Jang et al. [19] represented change impact analysis approach for selecting regression test cases in C++ programs. The technique recognizes procedure for retesting and analyzes all affected methods. The researchers have determined certain common types of changes that are possible for a C++ program and a method level firewall is designed for each change to analyze the result of the changes.

2.2.2. Program Model-Based Techniques

Rothermel et al. [5] suggested RTS technique for C++ programs based on the study of control flow representations of unmodified and the modified programs by continuing the technique proposed in [4]. Inter-procedural Control Flow Graph (ICFG) and Class Control Flow Graph (CCFG) have been introduced to represent control flow of multi-function programs and object-oriented programs respectively. An ICFG for unmodified program is made up of CFGs for each method. An ICFG is used to model single entry point programs, whereas a class can have multiple entry points [5]. A CCFG is used to model classes, and has individual CFGs for all methods of a class. For the graph models of the original and the modified programs, the RTS algorithm [5] extends the graph walk-based approach [4] to pass through the models and select relevant regression test cases.

Harrold et al. [10] suggested RTS technique for Java programs based on control flow analysis. The technique is an adaptation of the graph walk techniques suggested in [4, 5]and can handle various object oriented features such as inheritance, polymorphism, dynamic binding and exception handling. This technique selects test cases on the basis of the dangerous edges identified during graph traversal.

Orso et al. [1] suggested a two-phase partitioning approach for RTS for large Java programs.

2.2.3. Design Model-Based Techniques

Ali et al. [15] suggested an RTS technique that analyzes UML classes and sequence diagrams. This technique analyzes class and sequence diagrams at the level of class attributes and operations. The sequence and the corresponding class diagrams are analyzed and an extended concurrent control flow graph (ECCFG) is constructed to model the program. The information about which attributes of a class receive messages in a sequence diagram is derived from the corresponding class diagrams, and is represented in the ECCFG. The ECCFG models for the unmodified and the modified version of the application are then analyzed to find out the changes between program versions, this knowledge is further used to optimize regression test cases.

Briand et al. [9] suggested an RTS technique based to analyze UML design models. The design and test cases are tracked. This helps to club the modifications of design models to the test cases which need to be executed for affected parts of the design. Their approach involves analysis of use case, class and sequence diagrams. The paper defines test cases as obsolete, retest able and reusable during the analysis.

2.2.4. Specification-Based RTS Techniques

Chen et al. [18] suggested a specification based RTS technique that uses UML activity diagrams in modeling potentially affected requirements and system behavior. They classified regression test cases as target and safety test cases for selection. Target test cases identify the affected requirements and safety test cases tries to achieve a pre-defined coverage target. A traceability matrix is defined to capture the relation among requirements and the test cases. The changes to the original program can result in change of specification. Chen et al. [5] extended the RTS technique to handle changes which lead to the changes in specifications. Safety test cases are selected with a target to mitigate risks. The idea is to more thoroughly test those parts of the code for which the probability of a fault being present and its cost is high [2].

Chittimalli and Harrold [12] suggested specification based RTS approach. The technique is based on finding out specifications being tested by which test case from original test suite. This information results in requirement coverage matrix among the set of requirements and the test cases. The technique proposed by [1] is utilized in identifying affected parts of the code, and later the set of requirements affected due to modifications are also discovered. These are defined as affected requirements. The knowledge gathered from requirement coverage matrix is used in selection of the test cases which will executes on affected requirements.

Figure: 2 summarizes RTS techniques for object oriented paradigm.

Regression Test Selection Technique for Object Oriented Programming Paradigm		
RTS Techniques	Characteristics	Advantages
Firewall based technique	Dependencies among modules is analyzed	It is computationally efficient
Program model based technique	Dependencies among class elements is analyzed	It can be applied for modified classes and classes derived from the modified classes
	Control flow model analysis	Efficient and safe technique for C++ program
	Based on analysis of control flow models, it is a two phased technique	Two phased technique is computationally more efficient, safe and precise for Java programs.
Design model based technique	Different UML design models are analyzed	It is suited for RTS of large programs.
Specification based techniques	Requirement models are analyzed, provides completeness in the traceability of test cases.	These techniques are platform independent and they can be extended to a wide class of programs

Figure: 2

3. Research Methodology

The paper is an exploratory study of regression test selection techniques in procedural and object oriented programming paradigm. The research methodology used in the paper is a comprehensive study of the work by various scholars through their research papers published in different e-journals, acclaimed publications and reference books. The research study in also conducted through different websites, online portals and forums.

4. Analysis and Conclusion

The paper is an exploratory study of regression test selection techniques suggested by different researchers in the procedural and object oriented paradigm. The paper further categories regression test selection techniques respectively as explained by the researchers.

From the exploratory study analyzed for procedural programming paradigm the Dataflow Analysis based Technique is based on dataflow and structural coverage criteria which can analyze both intra and inter-procedural modifications. Slicing Based Techniques is based on slicing of programs or dependence graph models which can analyze both inter and intra procedural modifications. Firewall Based Technique is based on analyzing dependencies among modules of source code of only modified modules. Differencing Based Technique is based on textual differencing of C programs, it is safe and easy to implement. Control Flow Analysis Based technique is based on analysis of control flow model, it is safe and precise procedural technique.

For object oriented programming paradigm the Firewall Based Techniques analyzes dependencies among modules which provide computational efficiency. Program Model Based Technique analyses dependencies among class elements, which is safe and precise, it is applicable for both modified classes and classes derived from the modified classes. Program Model Based Technique which is based on control flow models analysis is efficient and safe RTS technique for C++ programs. Program Model based Technique which is based on analysis of control flow models is a two phase technique which is safe and precise for Java programs. Design Model Based Technique is based on analysis of different UML design model, assumes that traceability exists between the design model, the source code and he test cases, suited to model driven development environments. It is suited for RTS for large programs, analysis is at a higher level of abstraction, and is independent of the implementation. Specification Based Technique; analysis the requirement models, it assumes complete traceability from the specifications to test cases. This technique is platform independent and can be easily extended to a wide class of programs. The paper analyzed that specification based technique is one of the most efficient technique as tester do not have source code of the software under test with them. Therefore specification based technique facilitates the tester to create test cases on the basis of specification provided. Regression Test Selection has many facets some of them have been discussed in this paper the study can be extended further to uncover various techniques in different paradigms.

References

[1] A. Orso, N. Shi, and M. Harrold. Scaling regression testing to large software systems. In Proceedings of the 12th ACM SIGSOFT Twelfth International Symposium on Foundations of Software Engineering, November 2004.

[2] D. Binkley. Semantics guided regression test cost reduction. IEEE Transactions on Software Engineering, August 1997.

[3] 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, January 1996.

[4] G. Rothermel and M. Harrold. A safe, efficient regression test selection technique. ACM Transactions on Software Engineering and Methodology, April 1997.

[5] G. Rothermel, M. Harrold, and J. Dedhia. Regression test selection for C++ software. Software Testing, Verification and Reliability June 2000.

[6] H. Leung and L. White. A study of integration testing and software regression at the integration level.In Proceedings of the Conference on Software Maintenance.

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

[8] J. Laski and W. Szermer. Identification of program modifications and its applications in software maintenance. In Proceedings of the Conference on Software Maintenance.

[9] L. Briand, Y. Labiche, and S. He. Automating regression test selection based on UML designs. Information and Software Technology, January2009.

[10] M. Harrold, J. Jones, T. Li, D. Liang, A. Orso, M. Pennings, S. Sinha, S. A. Spoon, and A. Gujarathi. Regression test selection for Java software. In Proceedings of the 16th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications.

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

[12] P. Chittimalli and M. Harrold. Regression test selection on system requirements. In ISEC '08: Proceedings of the 1st conference on India software engineering conference.

[13] R. Pressman. Software Engineering: A Practitioner's Approach. McGraw-Hill, New York, 2002.

[14] S. Bates and S. Horwitz. Incremental program testing using program dependence graphs. In Conference Record of 20th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages.

[15] Sebastian Elbaum, Praveen Kallakuri, Alexey G. Malishevsky, Gregg Rothermel, SatyaKanduri, "Understanding the Effects of Changes on the Cost-Effectiveness of Regression Testing Techniques," Journal of Software Testing, Verification, and Reliability.
[16] T. Ball On the limit of control flow analysis for regression test selection. In ISSTA '98: Proceedings of the 1998 ACM SIGSOFT international symposium on Software testing and analysis.
[17] Y. Chen, D. Rosenblum, and K. Vo. Test Tube: A system for selective regression testing. In Proceedings of the 16th International Conference on Software Engineering.
[18] Y. Chen, R. Probert, and D. Sims. Specification based regression test selection with risk analysis. Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research.
[19] Y. Jang, M. Munro, and Y. Kwon. An improved method of selecting regression tests for C++ programs.