IJITKM Volume 6 • Number 2 • July-December 2013 pp. 113-117

REGRESSION TESTING TECHNIQUES: A SURVEY

Tanvi Agrawal[1] Vandana Sharma[2] and Arun Prakash Agrawal[3]

ABSTRACT: Regression testing is a part of software maintenance which consumes around two-third of the overall software life cycle cost hence; it is an expensive activity that may be conducted either manually by re-executing a subset of all test cases or using automated capture/playback tools. Capture/playback tools enable the software testers to capture test cases and results for subsequent playback and comparison. Regression testing tests both the modified code and other parts of the program that may be adversely affected by changes introduced in the program or a part of it. Test case selection selects the test cases to test the modified part of the program from the original test suite. Test case prioritization reorders test cases in a way that premature fault detection is maximised. In this paper we have done a survey of selection and prioritization techniques in regression testing.

Keywords: Software Testing, Regression Testing, Test Case Selection, Test Case Prioritization.

1. INTRODUCTION

Software Testing is the process of executing a program or system with the intent of finding errors. This is done after the development phase is complete. Once the product is delivered it enters into the maintenance phase. Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of existing features. These modifications may cause the system to work incorrectly. Therefore, Regression Testing becomes necessary. Regression testing is a type of software testing executed in the maintenance phase which uncovers new errors in existing functionality of the software after changes have been made, such as functional enhancements, patches or configuration changes. An important difference between development testing and regression testing is that while performing regression testing an established suite of tests is available for reuse. A regression test means that whenever a new function is added or modifications are done in the software, all previous validated test cases are run, and the results are compared with the standard results previously stored on file. Major objectives of regression testing are firstly retest changed components and secondly check the affected parts. This testing is done to make sure that new code changes should not have any side effects on the existing functionalities. It ensures that old code still works once the new code changes are done. Figure 1 below shows the various tasks performed in regression testing. The test cases are classified as: reusable, re-testable, obsolete, new-structural and new-specification test cases [1].

[1] ASET, Amity University, Sector-125, Noida, India
E-mail: tanviagrawal1987@gmail.com
[2] ASET, Amity University, Sector-125, Noida, India
E-mail: vandysharma12@gmail.com
[3] Assistant Professor, Amity University, Sector-125, Noida, India
E-mail: apagrawal@amity.edu

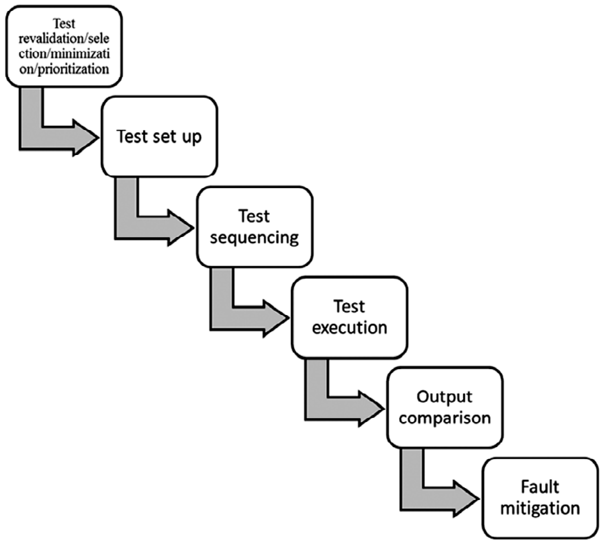

Figure 1: Various Tasks Performed in Regression Testing [11]

1.1. Regression Testing Techniques

Retest all: This is the method for regression testing in which all the tests in the existing test suite are re-executed. This is very expensive as it requires huge time and resources. It is 100% fault detecting technique and also there is no size reduction in the test suite.

Regression Test Selection (RTS): RTS selects specific test cases from the existing test suite instead of running the entire test suite again. The strategy of RTS is to minimize the test suite and maximize fault detection ability. Selected test cases can be classified as:

(1) Reusable Test Cases that can be used in succeeding regression cycles.

(2) Obsolete Test Cases that can't be used in succeeding cycles.

Prioritization of Test Cases: It is ordering of test cases for testing depending on business impact, critical & frequently used functionalities. This type of ideal ordering of test cases will greatly reduce the test suite of regression.

2. REGRESSION TEST CASE SELECTION (RTS)

Regression test selection (RTS) techniques selects a subset of valid test cases from original test suite {T} to test that the modified part of the program does not affect the unmodified parts of a program and hence the program must continue to work correctly. It affects the cost-effectiveness of regression testing and involves two major activities:

(1) Identification of the modified parts of the program

(2) Selecting the subset of test cases that has high probability of detecting errors.

2.1. Outline for Regression Test Selection (RTS) Techniques [2]

Some of the matrices and characteristics on the basis of which RTS techniques can be categorized as: inclusiveness, precision, efficiency, and generality.

Inclusiveness: It measures the extent to which the RTS technique chooses modification revealing tests from previous test suite {T} for inclusion in modified test suite {T′}.

Precision: It measures the extent to which the RTS technique omits test cases that does not reveal modifications.

Efficiency: The efficiency of RTS technique measures the space and time efficiency which depends on the computational cost and the size of the test suite selected by the technique.

Generality: The generality of RTS technique is its ability to perform in different environment and situations. The technique needs to be practical, able to handle realistic modifications and does not depend on some specific tools.

2.2. Various Techniques of RTS are

1. ***Random Technique:*** In this random technique the test cases are selected randomly without following any criteria from the original test suite [3].

2. ***Integer Programming Approach:*** One of the earliest approaches to test case selection was used in Integer Programming (IP) to represent the selection problem for testing FORTRAN programs presented by Fischer [8, 49]. It presented a selective test case technique to select test suites and uses system of linear equations to yield segment coverage of modified code. Lee and He [2, 25] proposed a similar technique. Fischer, Raji, and Chruscicki [2, 11] extended Fischer's earlier work incorporating information on variable definitions and its uses. Hartmann and Robson [2, 18], [2, 19], [2, 20] extended and implemented Fischer, Raji, and Chruscicki's techniques. IP technique use structures of linear equations for expressing relationships between tests and program segments. The track program segments reached by retest, segments accessible from other segments and information about the segments form matrices which are used to obtain system of linear equations. It does not necessarily perform minimization rather it can select between numbers of test cases traversing the corresponding coverage entity.

3. ***Path Analysis:*** Benedusi [8,9] introduced a selection technique based on path analysis. They constructed exemplar paths from the original program P and the modified program P′ expressed in an algebraic expression. The technique is then required to compare between the exemplar path of P and exemplar path of P′ and classify them into new, modified, cancelled or unmodified paths. In P the paths executed by the test cases are known and hence all the test cases that will traverse modified paths in P' are selected.

4. ***Data-flow Analysis Technique:*** This technique identifies the definition-use pairs that are new, modified or deleted in P', and select those pairs of test cases that exercise these pairs. Two approaches were given for data flow analysis i.e. incremental technique and non-incremental technique. In the incremental technique a single change is processed, tests are selected for that change, information about the test trace and data flow are incrementally updated and then the process is repeated for the next change. In the non-incremental technique a multiply-changed program is processed considering all modifications simultaneously. The dataflow regression testing techniques described by Gupta, Harrold, and Soffa [2, 13], Harrold and Soffa [2, 15] Taha, Thebaut, and Liu [2, 39], and Ostrand and Weyuker [2, 31] are sufficiently alike to justify treating them together.

5. ***Safe Technique:*** This technique was implemented by Rothermel and Harrold. They implemented a safe algorithm which is implemented as a tool called DejaVu [3]. This tool performs the depth first search on two control flow graphs (CFG) - one is the CFG of the original program and the other one is the CFG of the modified program.

6. ***Symbolic Execution Technique:*** Yau and Kishmoto [2, 45] introduced symbolic execution approach for test case selection techniques. It uses symbols for variable values rather than using them as actual values and also used input partitions and symbolic execution to select the test cases and execute them. In this approach the input partitions are derived from analysing code and specifications for the modified program. Then new test cases are generated so that each input partition value is executed at least once by the test case. Once the information is given that where the code has been modified, the edges in the control flow graph for the new program are recognized that leads to the modified program. Next the symbolic execution is performed on all the test cases and determines the test cases that traverse edges that do not reach any modification. The symbolic test cases that reach the modifications are not required to be executed more and one which matches with the symbolic test cases are retested again. The shortcoming of this approach is algorithmic complexity of the symbolic execution.

7. ***Dynamic Slicing Based Technique:*** Slicing technique is a test case selection technique introduced by Agrawal et al. [2, 1]. There are four types of slicing techniques i.e. dynamic slice, execution slice, relevant slice and approximate relevant slice. An execution slice of a program contains the same set of statements executed by the given test case and also the output of the program is not affected. Dynamic slice is a subset of execution slice. A dynamic slice of a program contains the set of all the statements in the execution slice which have an influence on the output statement. A relevant slice for a test case is similar to the dynamic slice for the same test case with all the predicate statements that, it changes would be done, it would return different output. Finally, the approximate relevant slice in a program is like a dynamic slice which contains all the predicate statements in the execution slice.

8. ***Textual Difference Technique:*** A selection technique was proposed by Volkolos and Frankl [8, 165] [8, 166]. This technique is based on the textual difference between the source codes of two versions of system. UNIX tool was applied to the source code of different versions of the system to identify the modified parts. This technique is similar to CFG based graph walk approach.

9. ***Graph-Walk Technique:*** Rothermel and Harrold presented regression test case selection techniques based on graph walking of Control Dependence Graphs (CDGs), Program Dependence Graphs (PDGs), System Dependence Graphs (SDGs) and Control Flow Graphs (CFGs) [8, 135] [8, 137] [8, 139] [8, 140]. The difference between CDG and PDG is that CDG lacks data dependency relations. To identify the points in a program through which execution traces reaches the modifications, a depth-first is performed to traverse both the programs P and P′. All the test cases are selected that execute the control-dependence predecessors of the mismatching node if a node in the CDG of P is not same as that to the corresponding node in the CDG of P'. PDG approach is used for intra-procedural selection and SDG approach is used for inter-procedural selection. PDGs contain data dependence for a single procedure; SDGs extend this to a complete program with multiple procedures. By using these graphs, the algorithm is able to check whether a modified definition of a variable is actually used later. The CFG technique follows the approach which was introduced for the CFG technique. CFG is more efficient technique with a simpler representation of the structure of the program.

10. ***Minimization Technique:*** In this technique a simulator tool is implemented which works as a minimization technique. The tool is implemented using the regression test selection safe algorithm (DejaVu) implemented by Rothermel and Harrold [3].

11. ***Modification-Based Technique:*** This technique is based on test tube framework which partitions the program into entities and monitors the execution to set up the relationship between the program and entities and then the selected test cases are re-executed. It is contemplation of an extended adaptation of graph walk technique. Pointer handling is the only weakness of this technique and work on assumption basis.

12. ***Firewall Approach:*** Leung and White introduced the firewall approach and later implemented it for regression testing of system integration [8, 102] [8, 103] [8, 171] [8, 172]. The main concept revolves around the modules of the system that need to be retested. They categorised the modules into the following categories: No Change, Only Code Change, and Spec Change. This approach has been functional to Object-Oriented programs [8, 9] [8, 169] [8, 170] and GUIs [8, 168] and later the information of this approach has been extracted by many researchers.

13. ***Design-Based Approach:*** This approach is presented by Briand et al. for UML-Based designs [8, 17] [8, 18]. There is an assumption in this approach that there is traceability among the design and the regression test cases. This approach is likely to execute RTS of code-level test cases from the contact analysis of UML design models. It classifies the applicable test cases into these categories: obsolete, re-testable and reusable [8, 101].

14. ***Cluster Identification:*** This technique is a test case selection technique based on analysis of the Control Flow Graph (CFG). It identifies the clusters prepared by the single entry and single exit of the CFG of the program [8, 5]. The idea behind this technique is to create two CFG's one for the original program and one for the modified program. After receiving both the

CFG's they compared both the CFG's on the basis of node of the graph. The best part of this technique is that it guarantees to select all the modified test cases in spite of the kind of modification.

15. ***SDG Slicing Approach:*** This technique is based on the program slicing which further depends on the basis of program dependency graphs [8, 8]. It introduced the meaning of an equivalent implementation outline and identifies all the test cases that is crucial to be reused. Bates and Horowitz proposed test case selection techniques based on program slices from Program Dependency Graphs. In this technique, the equivalent behaviour is recognized when the slices of two statements is isomeric. The reusable test cases are chosen on the basis of sequence retrieved from the recognition stage. This technique is further extended by Binkley who introduced the idea of frequent execution patterns, which corresponds to the implementation patterns of the original technique proposed, in order to detain the numerous incantation of the course of action.

3. REGRESSION TEST CASE PRIORITIZATION

These techniques help in the execution of the test cases based on the assigned priority to the test cases. The priorities are set by certain criteria. It is beneficial as meeting the testing goals earlier can lead to desired results. This technique specifies which test case will be addressed first from the original test cases. It does not discard any test case and the efficiency of the regression testing depends on the criteria of the prioritization. [15] General test case prioritization and version specific test case prioritization are the two varieties of test case prioritization. In General test case prioritization the test cases are prioritized on the basis of their usefulness and without any prior knowledge of modifications. In specific test case prioritization the test cases are prioritized on the basis of with any prior knowledge of modifications. All the test cases are prioritized on the basis of risk analysis. The risk analysis is done on the basis of complexity, criticality, and impact of failure. The impact of failure can range from 'no failure' to 'human life'. The risky factor should be tested on higher priority.

3.1. Various Techniques of Regression Test Prioritization are:

1. ***Coverage-Based Prioritization:*** This technique can be used to examine the internal structure of the program [15]. It is based on: code-coverage which is done for the complete code of the program, branch-coverage done to test every 'true' and 'false' condition of the program , path-coverage done to test each and every path of the program, statement-coverage is done in order to achieve 100% execution of every statement of the program, condition-coverage is done to achieve the confidence about the correctness of the program, variable-coverage and requirement-coverage is done till time and resources are available or confidence is achieved.

2. ***Random Prioritization (Random):*** This technique selects the order of the test cases randomly from the original test suite [13].

3. ***Interaction Testing:*** Interaction testing is required when different combinations of environments (operating systems) are required like, configuration testing. It involves multiple combinations of components and not simply putting the components together [8]. This type of testing is required to ensure that the system under test works correctly. It is a systematic technique for constructing the program structure and at the same time conducting tests to discover errors linked with interfacing. The purpose of this technique is to take unit tested components and construct a program structure that has been described by the design. The integration testing is of two types:

   (a) ***Decomposition Based Integration:*** Decomposition Integration is based on the functional decomposition of the System. These are of four types:

   - Top-down Approach
   - Bottom-up Approach
   - Mixed Approach
   - Big bang Approach

   (b) ***Call Graph Based Integration***

   - *Pair-wise Integration:* this testing is done basically to eliminate the stub and driver.
   - *Neighborhood Integration*: It involves reduction in the number of integration test sessions. In this testing there is a graph consisting of set of nodes that separates from each other one edge away.

4. ***Optimal Prioritization (Optimal):*** This technique intends to find out the optimal order of the test cases on the basis of genetic algorithm which tries to find out the best and effective order of test cases [13].

5. ***Model-Based Prioritization:*** This technique prioritizes or reorders the test cases for object-oriented programs and represents all the features of object-oriented i.e. polymorphism, inheritance, aggregation and association [14]. Whenever there are some changes in the program it compares the code of both the modified and the original program in-order to find the modified statement.

6. ***Historical Fault Detection Effectiveness Prioritization:*** This technique reorders the test cases on the basis of the historical values depending on the fault detection effectiveness of the test cases [13].
7. ***Cost-Aware Test Case Prioritization:*** This prioritization technique not only depends on highest priority and the lowest priority of the test cases in order to achieve the results but also considers the cost effectiveness of the techniques so that the selected technique should not be costly as compared to the original test suite.
8. ***Time-Aware Test Suite Prioritization:*** This technique reorders the test cases on the basis of genetic algorithm and the fitness function. This approach considers both the execution history and coverage of the program [13].
9. ***Prioritization Approaches Based on Other Criteria:*** The majority of active prioritization concerns with the structural coverage in several forms. There are various prioritization techniques based on other criteria [8, 100] [8, 158][8, 163] [8, 180].

   These criteria's are:

   - Distribution-based Approach
   - Human-based Approach
   - History-based Approach
   - Requirement-based Approach
   - Model-based Approach
   - Other Approaches

4. CONCLUSION

This paper provides a detailed analysis of trends in regression test case selection and prioritization. The study of trends reported in this paper reveals some intriguing properties of regression test selection and prioritization techniques. This paper discussed some work that has been done in the techniques of test case selection as well as in prioritization which helps future researchers to study, learn and understanding the work done till now.

REFERENCES

[1] H. Leung and L. White, "Insights into Regression Testing", *In Proceedings of the Conference on Software Maintenance.*

[2] G. Rothermel, M.J. Harrold, "Analyzing Regression Test Selection Techniques", *IEEE Transactions on Software Engineering*, **22,** No. 8, August 1996.

[3] T. L. Graves, M.J Harrold, JM Kim, A Porter, G Rothermel, "An Empirical Study of Regression Test Selection Techniques", *ACM Transactions on Software Engineering and Methodology.*

[4] G. Rothermel and M.J. Harrold, "A Safe, Efficient Algorithm for Regression Test Selection", Technical Report 115, Clemson University, Clemson, SC, April, 1993.

[5] G. Duggal, Mrs. B. Suri, "Understanding Regression Testing Techniques", Guru Gobind Singh Indraprastha University, Delhi, India.

[6] S. Biswas and R. Mall, Regression Test Selection Techniques: A Survey, Informatica 35 (2011).

[7] X. Lin, "Regression Testing in Research and Practice", *Computer Science and Engineering Department University of Nebraska*, Lincoln.

[8] S. Yoo, M. Harman, "Regression Testing Minimisation, Selection and Prioritization: A Survey, King's College London", *Centre for Research on Evolution, Search & Testing*, Strand, London, WC2R 2LS, UK.

[9] Y. Chen, R. L. Probert, D.P. Sims, "Specification-Based Regression Test Selection with Risk Analysis", *CASCON '02 Proceedings of the 2002 Conference of the Centre for Advanced Studies on Collaborative Research.*

[10] Sujata, G.N. Purohit, "Tool Support for Test Case Selection in Regression Testing", *International Journal of Software Engineering & Applications* (*IJSEA*), **2,** No. 4, October 2011.

[11] A. P.Mathur, Foundations of Software Testing: *Fundamental Algorithms and Techniques*, 2008.

[12] G.Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, "Prioritizing Test Cases for Regression Testing", *IEEE Transactions on Software Engineering*, **27,** No. 10, OCTOBER 2001.

[13] Y.Huang, K.L Peng, C. Huang, "A History-Based Cost-Cognizant Test Case Prioritization Technique in Regression Testing", *The Journal of Systems and Software.*

[14] C. R. Panigrahi, R. Mall, "Model-Based Regression Test Case Prioritization", *ACM SIGSOFT Software Engineering* Notes Page 1 November 2010 **35** No. 6.

[15] K.K. Aggrawal, Y. Singh, A. Kaur, "Code Coverage Based Technique for Prioritizing Test Cases for Regression Testing", *ACM SIGSOFT Software Engineering Notes September* 2004 **29** No. 5.