International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 5, May - 2014

Review of Regression Test Case Selection Techniques

Manisha Rani
CSE Department, DeenBandhuChhotu Ram University of Science and Technology,
Murthal, Haryana, India

Ajmer Singh
CSE Department, DeenBandhuChhotu Ram University of Science and Technology,
Murthal, Haryana, India

Abstract **- Regression Testing is the most important activity that is done to ensure that the modifications do not introduce new errors in the previous existing code. An important research problem is the selection of relevant test cases from the initial suite that would minimize the test time and efforts without affecting test quality. In this paper, we review various test case selection techniques and do their comparisons.**

1 INTRODUCTION

1.1 Background

Regression testing is the testing strategy which guarantees that newly introduced changes in software do not affect the unchanged parts of the software. We can apply "Retest All" Strategy to regression testing in which all of the existing test cases are executed to ensure that new changes do not introduce any error. But it is a very expensive process , it consumes a lot of time and resources. Therefore, we consider test case selection and prioritization techniques. Test Case Selection Techniques reduce the number of test cases and satisfy the testing requirements. There are various test criteria on which we can perform testing e.g. fault detection ability, code coverage, time taken to detect fault, past fault detection history. This is beneficial for the tester to consider multiple test criteria. We are not only concerned with quality of test data but with cost also. The purpose of test case selection is to achieve effective test case selection in terms of minimum cost.

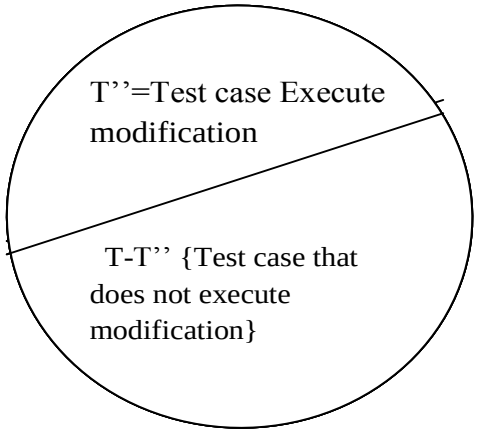
Regression Test Selection consists of mainly two activities:

a) Firstly, we identify the affected part of the code. This involves identifying the unmodified part of program that gets affected due to modifications.

b) Secondly, we select the test case that involves the selection of subset of test cases from the initial test suit T.

Regression Testing

Let P be our original program. Let P' be a modified version of P and T be a test suite for P. A typical regression test proceeds as follows:
(1) Select T' subset of T, a set of test cases to execute on P.
(2) Test P′ with T' to check whether P' is correct with respect to T'.
(3) Create T", a set of new functional or structural test cases for P'.
(4) Test P' with T" to check whether P' is corrected with respect to T".
(5) Create T"' that includes test cases from T' and T".



T is the original regression test set. Fig: 1 Classification of test cases

1.2 Classification of Test Cases

Leung and White [1] classify test cases into five classes. The first three classes consist of test cases which already exist in T.

Reusable: Reusable test cases execute the parts of the program that remain unchanged
and common to program P and P'. It is not needed to execute these test cases on P'. These are called reusable because we can use them for the future version of P.

Re-testable: Re-testable test cases execute the parts of P that get changed in P'. Thus

Re-testable test cases should be re-executed in order to test P'.

Obsolete: Test cases can be considered obsolete because
1) Their input/output relation get changed due to changes in specifications
2) They no longer test what they were designed to test due to modifications in the program
3) They are 'structurally' test cases that no longer contribute to structural coverage of the program.

The remaining two classes consist of test cases that are to be generated for the regression testing of P'.

New-structural: New-structural test cases test the modified program constructs that provide structural coverage of the modified parts in P'.

New-specification: New-specification test cases test the modified program specifications. It will test the new code

generated from the modified parts of the specifications of P'.

1.3 Concepts Related to Regression Testing

Execution Trace of a Test Case: The execution trace of a test case t on a program P is defined as the sequence of statements in P that are executed when P is executed with t. The execution trace information for P can be generated automatically with the help of the source code.

Fault-revealing Test Cases: A test case $t \in T$ is said to be fault-revealing for a program P, if and only if by executing this on program P, there is a failure due to incorrect output.

Modification-revealing Test Cases: A test case $t \in T$ is considered to be modification-revealing for P and P′ if it produces different outputs for P and P′.

Modification-traversing Test Cases: We said a test case $t \in T$ is modification-traversing for P and P′ if and only if the execution traces of t on P and P′ are different. A test case t is said to be modification-traversing if it executes the modified regions of code in P′. The set of modification-traversing test cases is a super-set of the set of the modification-revealing test cases.

Inclusive, Precise and Safe Regression Test Cases:
Inclusive: Inclusiveness measure the extent to which we select modification revealing test cases from the original test suit T. Suppose the original test suite contains n modification revealing test cases. If regression test selection technique selects m modification revealing test cases, the inclusiveness is expressed as (m/n)*100.

Safe: A safe regression test selection technique selects all the modification revealing test cases. In other ways, if it is 100% inclusive then we said it a safe technique. Regression test that are relevant to change but are not selected are corresponds to the false negative. A safe technique excludes all the false negatives.

Precision: Precision measure the extent to which regression test selection ignores the non modification revealing test cases. The selected test cases that are not relevant to change are known as false positives. Thus Precision measures the degree to which an RTS ignores the false positives.

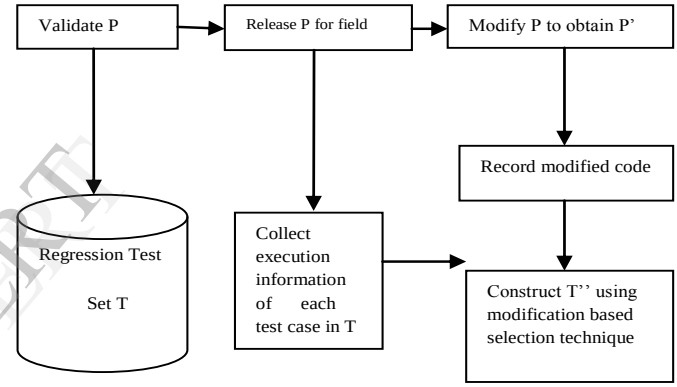


Fig 2: Regression Test Case Selection

2 RTS Techniques for Procedural Programs
There are five major classes of testing techniques:
1. Dataflow analysis-based techniques
2. Slicing-based techniques
3. Firewall-based techniques
4. Differencing-based approaches
5. Control flow analysis-based techniques

2.1 Dataflow Analysis-Based Techniques
Several techniques have been proposed based on dataflow analysis [2-5].In this technique, we execute test cases for definition use pairs of variables that are affected by program modification. The test cases execute the path from definition of modified variables to their use. We can use variable in two different ways-computation use(c-uses) and predicate uses (p-uses). A c-use occurs when it is used in computation and a p-use occurs when it is used in conditional statement. A c-use indirectly affects the control flow of the program while a p- use may affect directly or indirectly.
Harrold and Soffa [3] have proposed a dataflow coverage-based RTS technique that can be applied to analyze changes

across multiple procedures. It involves the incremental approach for processing the dataflow information i.e select test cases to process a single change and update test coverage information and dataflow information. This process is then repeated for all the changes one by one. In this approach, we represent P by CFG in which basic blocks are represented with nodes. This will reduce the size of flow graph and we can do analysis of graph efficiently. We introduce additional nodes to model global variables, function parameters and return values of function. When we define variable in node n, we store the node number of all c-auses of the variable in node n. We also store block number for all p-uses in n. This information is used to select the test cases.

Critical Evaluation
1) These techniques are unsafe because these do not consider control dependency among program elements.
2) These techniques are also imprecise because the presence of an affected code in a block does not guarantee that every test case will execute this block.

2.2 Slicing-Based Techniques
Aggrawal et al.[6] have proposed the slicing based technique. We select those test cases which execute to produce different outputs with the modified program P'. A slice is defined with respect to T as the set of statement which are executed when P is executed with test case T. There are four slicing techniques[6]: Execution slice, dynamic slice, relevant slice, approximate relevant slice. Two components have equivalent execution patterns if they are executed the same number of times on any given input. Code elements possess common execution pattern if they have the same equivalent execution pattern during some call to procedures. These patterns are used to find out the semantic difference.

Critical Evaluation
1) These are not safe technique when there is deletion of statement involve in a block.
2) These are precise because these do not involve test cases which do not produce different output.

2.3 Module Level Firewall-Based Techniques
Leung and White [7-9] have proposed the firewall based techniques. It is based upon data and control dependency among various modules of a procedural program .A firewall is defined as the set of modified modules and the modules which interact with these modified modules. The firewall limits the amount of retesting and selects the test cases which execute the modified modules. To represent the control flow structure of program, it uses call graph approach. If there is a path from module A to B then module A is known as ancestor and module B is known as descendant.
It includes direct ancestor and direct descendant with modified module while constructing a firewall for all possible interactions. Then it use coverage based information to select the test cases.

Critical Evaluation
1) The firewall techniques are not safe because these do not select test cases from outside the firewall which can execute affected statements.
2) The firewall techniques are imprecise because test cases executed are not necessarily to execute the affected modules within the firewall.

2.4 Differencing Based Techniques
These techniques are based on differences between original and modified program [10,11]. These techniques can be divided into various categories:

2.4.1 Modified Code Entity-Based Technique
Chen et al.[10] have proposed the modified code entity based technique. In this technique, program code is decomposed into functional and non-functional code. We can define code entity as either executable code such as function or a statement or a non executable code as global variable or a macro. The test coverage information is analyzed to determine the set of executable code entities that are exercised by each test case.

2.4.2 Technique Based on Textual Differencing
Vokolos and Frankl [11,12,13] have proposed technique based on textual differencing of original and modified program. It does not use intermediate representation of program. In this technique program is converted into canonical form before comparison. The modified program should follow the same syntactic and formatting guidelines. The canonical version of P is instrumented to generate test coverage information. It compare the canonical version of P and P' to find out modifications to the code.

Critical Evaluation
1) It is a safe technique because test cases are selected on the basis of affected code entities.
2) It is imprecise if code changes are arbitrary because it selects test cases only on the basis of differentiation of syntax.

2.5 Control Flow Analysis-Based Techniques

These techniques[14,15,16] analyze control flow models of the input programs for selecting test cases. There are various types of control flow analysis based techniques.

2.5.1 Cluster Identification Technique

Laski and Szermer[14] have proposed the cluster identification technique. Cluster is defined as the localization of program modification in one or more areas. Cluster is a single- entry and single- exit block that changes from one version to another. In this technique program P and P' are represented as CFGs and then nodes G and G' which corresponds to the modification are identified that forms the cluster. It uses control dependence information of original and modified procedure to find out clusters. We can modify program in three ways: Inserting a cluster into the program, deleting a cluster or by modifying a cluster. There are two types of test cases: local to cluster and global to the cluster. The former are executed inside the modified cluster and latter is to find out the affected module due to the modified module.

2.5.2 Graph Walk Based Technique

Rothermel and Harrold [15] have proposed graph walk based technique which is based on the traversal of Control Flow Graphs(CFGs) of original and modified program. It is better than RTS technique based on dependence graph. In this, CFGs G and G' for program P and P' are constructed. Then by instrumenting P, the execution trace of each test case t, ET(P(t)) is recorded by performing Depth first traversal. This technique examines the program statements along identically labeled edges of G and G' are equivalent or not. The edges corresponds to the non identical nodes are identified as dangerous edges. Modification revealing test cases are those which execute the set of identified dangerous edges.

2.5.3 DFA Model Based Approach

Ball [16] has proposed DFA Model Based Approach. It is based on modeling of CFG in deterministic finite automata. It constructs DFA M for CFG G based on following conditions:

1) Each node v in G is corresponds to two states v1 and v2 of M where v1 and v2 is connected by transition of basic block associated with v in G.
2) The set of edges in G constitutes the state transition in M.

These two conditions ensure that DFA M will work for graph G. It uses intersection graph to represent CFG. It is based on the reach-ability of edges in the intersection graph. It considers the edge coverage criteria for test case selection.

Critical Evaluation

1) The control flow analysis techniques are safe.
2) Cluster Identification techniques are imprecise because in this test case selection is done on the basis of whether it execute cluster or not but not on the basis of program modification.
3) DFA based approach is more expensive in terms of computation.

3.1 Comparison of Test Case Selection Techniques

Class of RTS Techniques	Key Features	Merits	Demerits
Dataflow Analysis Based Techniques	It is based upon dataflow and structure of program	It can analyze both intra and inter-procedural modification if there is alteration of def-use pair	Low safety and imprecise
Slicing based Techniques	It is based on slicing of program and use dependence graph	Used for both intra and inter-procedural modification	Low safety, imprecise and computationally complex than dataflow techniques
Module level Firewall based Techniques	It is based on dependencies among modules	It is more efficient as only modules are analyzed for modification	Low safety and highly imprecise
Modified code entity based Techniques	Based on level of granularity	It is safe and most efficient procedural RTS Technique	Highly imprecise

Textual differencing based technique	Based on syntactical difference	It is safe and easy to implement as prototype	Imprecise and inefficient for large programs
Graph walk based Technique	It is based on control flow models	It is safe and most precise RTS Technique	Less efficient for large programs

3.2 Techniques for Regression Test Selection

Technique	Origin	Description
T1	Harrold and Soffa (1988)	Dataflow-coverage-based
T2	Leung and White (1990)	Procedural-design firewall
T3	Gupta et al. (1992)	Coverage-focused, slicing
T4	Vokolos and Frankl (1997)	Textual Differing – Pythia
T5	Wong et al. (1997)	Hybrid: modification, minimization and prioritization-based selection
T6	Willmor and Embury (2005)	Test selection for DB-driven applications
T7	White et al. (2005)	Extended firewall additional data-paths
T8	Leung and White (2005)	Retest-all

4. CONCLUSION

In this paper, we have represented the systematic review of test case selection techniques. We have identified the following points :

1) There are five different techniques of test case selection which can be classified on the basis of what input is required and on which level of granularity change is considered.

2) The differences between the techniques is not very strong and sometimes contradictory

3) There is no base for selecting superior technique. Instead the techniques are tailored to some specific situation.

5. REFERENCES

[1] Leung HKN, White L. "Insight into regression testing",Proceedings of International Conference on Software Maintenance(ICSM 1989), IEEE Computer Society Press, 1989.

[2] Gupta R, Harrold MJ, Soffa ML " An approach to regression testing using slicing", Proceedings of the International Conference on Software Maintenance (ICSM 1992), IEEE Computer Society Press, 1992.

[3] Harrold MJ, Soffa ML " An incremental approach to unit testing during maintenance" , Proceedings of the International Conference on Software Maintenance (ICSM 1998), IEEE Computer Society Press, 1988.

[4] Harrold MJ, Soffa ML "Interprocedual data flow testing" , Proceedings of the 3rd ACM SIGSOFT Symposium on Software Testing, Analysis, and Verification (TAV3), ACM Press, 1989.

[5] Taha AB, Thebaut SM, Liu SS " An approach to software fault localization and revalidation based on incremental data flow analysis" ,Proceedings of the International Computer Software and Applications Conference (COMPSAC 1989), IEEE Computer Society Press, 1989.

[6] H. Agrawal, J. Horgan, E. Krauser, and S. London "Incremental regression testing", In IEEE International Conference on Software Maintenance, 1993.

[7] H. Leung and L. White. A study of integration testing and software regression at the integration level. In Proceedings of the Conference on Software Maintenanc, November 1990.

[8] H. Leung and L. White. "A cost model to compare regressiontest strategies", In Proceedings of the Conference on Software Maintenance,1991.

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

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

[11] F. Vokolos and P. Frankl. Pythia: "A regression test selection tool based on textual differencing", In Proceedings of the 3rd International Conference on Reliability, Quality & Safety of Software-Intensive Systems (ENCRESS' 97), May 1997.

[12] F. Vokolos and P. Frankl " Empirical evaluation of the textual differencing regression testing technique", In ICSM '98: Proceedings of the International Conference on Software Maintenance, 1998.

[13] 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.

[14] J. Laski and W. Szermer "Identification of program modifications and its applications in software maintenance", In Proceedings of the Conference on Software Maintenance, November 1992.

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

[16] 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.