https://doi.org/10.15623/ijret.2018.0712009 Received: 22-10-2018, Accepted: 01-12-2018, Published: 24-12-2018

A SURVEY ON TEST CASE SELECTION AND PRIORITIZATION TECHNIQUES

Kiran Jammalamadaka[1]

[1]*Research Scholar, Computer science K L University, Andhra Pradesh, India*

Abstract

One of the expectations from Agile software development is to reduce the cost of the project as the Agile development methodology focuses on delivering the right product and eliminating the waste. However, with shorter project cycles, smaller and agile teams the cost is not going down [3]. This emphasis the focus to re look at the phases in the software product development and one such area is regression testing. Regression testing is one of the most expensive yet important phase in software development, hence cannot be completely ignored, however can be reduced or minimized without compromising on the quality of the product. Researchers used several techniques to reduce and prioritize the test cases. In this paper we have presented various techniques presented to test case selection and prioritization.

Keywords: *Regression testing, Test case reduction, Test case Selection, Test Case Prioritization, Test Suite*

----------***----------

1. INTRODUCTION

Regression testing is testing of already tested component or system, following a few code changes to ensure that new implementation or defect fixes do not break the existing functionality[5].The purpose of the regression testing is to give the confidence about the product, as the existing functionality works as expected and not hampered due to the changes made to the existing software, in order to implementing the new features. Ideally untouched areas need not be tested and only modified areas to be tested, with this approach. It has become more complex due to the trends like component-based development as a small change in the component leads to an entire testing of the module [4]. Often teams end up in executing all the test cases for all the changes.

This phenomenon is more likely to happen in the agile development and has become more complex as for every iteration a new set of test cases will be added, and test effort increases cumulatively. As for every iteration, an increment will be developed, tested and delivered, here the testing includes validation of existing functionality is not broken and new features are working as per the expectation.

The amount of regression suite is directly proportional to the amount of time which in turn cost, more number of test cases to be executed needs more time. Therefore, several methods and techniques have been studied with an objective to optimize or reduce the size of the test suite.

In this process of study, test suite optimization has been classified into three categories test case reduction, test case selection and test case prioritization [4].

1.1 Test Case Reduction

In agile, testing gets involved in the early stages of development and evolves along with the development, as and when the increment gets implemented and its immediately tested. However, in this process a few test cases may get repeated. Test case is a called redundant when the objectives of the testcase are same, this can be compared to code refactoring, during this activity a few test cases can be merged into one testcase or delete the sub tests from the super test. This activity is a permanent activity not for the temporary session or iteration or release, this refactoring will be done at the overall test suite level.

1.2 Test Case Selection

Unlike the above, test case selection is not a permanent activity, its dynamic in nature and a few test cases will be selected from the test suite after understanding the change went into the code, static code analysis is required to understand the change and its implications. By definition
Let's assume a program P and its changed to P1, the problem is the identify the subset of the test cases T1 from the overall test suite T to test thoroughly the P1 wit out compromising the quality.

1.3 Test Case Prioritization

Test case prioritization determines the order of the test cases to be executed, like after selecting the subset of test cases T1 from the suite T, test prioritization determines in which order the test cases need to be executed to uncover the defects early so that developers can fix the defects early. If a potential test case which is capable of uncovering a major defect executed at the end of the regression cycle and it uncovers a major defect, at the that time developers may not get time to fix and which may lead to slip the release date.

https://doi.org/10.15623/ijret.2018.0712009 Received: 22-10-2018, Accepted: 01-12-2018, Published: 24-12-2018

2. REGRESSION TEST SUITE SELECTION TECHNIQUES

Test case selection techniques are intended to reduce the number of test cases without compromising the quality. The objective of the test suite selection is to identify the most fault uncovering test cases from the given test suite in the modified program as well.

Let's assume a program P and a modified program P1 and a test suite T and subset of test suite is T1.T1 should be able to detect the errors on the program P1,however, if a new test cases are needed to test the output of the program then T1 should include the newly written program, the existing test cases should test the un modified part of the program.

Owing to cost, running all tests approach can not be taken as a preferred approach, Same with the Random select approach, as no guarantee that this approach can uncover the defects from the program as the selection of test cases are random.

With respect to the above, having a selection technique would be a better approach.

Hence, having a test case selection technique will help in reducing the cost.

Rothaermel and Harrold [6] have formally defined the regression test selection problem as follows:
"Let P be an application program and P′ be a modified version of P.
Let T be the test suite developed initially for testing P. An RTS technique aims to select a subset of test cases T1 and T to be executed on P1, such that every error detected when P1 is executed with T is also detected when P1 is executed with T1"

2.1 Metrics to Measure the Efficiency of Regression Test Selections

Regression Test selection has become very popular and attracted many authors from the last decade, a large number of RTS techniques have been introduced which we will discuss in detail in the later part of the paper.

To evaluate the effectiveness of different RTS techniques Rothaermel and Harrold have proposed a set of metrics [7].

2.1.1 Execution Trace of a Test Case

When a program gets executed a set of statements get executed and the set of lines are called code traces, similarly when a test case get executed on a program a set of code statements get executed and those can be captured using instrumenting the code, So this metric insists that modified code should be in the set of test case trace, otherwise the RTS selection is not accurate.

2.1.2 Fault-Revealing Test Cases

The selected test case should be fault revealing, of the modified program. When the selected test case executed on a modified program, the test case should uncover the error and cause the program to fail.

2.1.3 Modification-Revealing and Traversing Test Cases

The selected test case should be able to reveal the modification done to program by providing a different output, original program and modified programs should give different results for the same test case selected. Similarly, the selected test case should traverse through the modified code of the program.

2.1.4 Inclusive, Precise and Safe Regression Test Cases

Another metrics are Inclusive, precise and safe, inclusive is about how extent RTS technique selected the testcase related to the change and how safe each of the selected test case as any missed test case may lead to a failure of the program. And how optimized the RTS is it should not pick which are irrelevant to the modified program.

2.2 Classification of RTS Techniques

Swarnendu Biswas and Rajib Mall [8] have classified the several proposed RTS techniques in the following

1. Dataflow analysis-based techniques
2. Slicing-based techniques
3. Module level Firewall-based techniques
4. Differencing-based approaches
5. Control flow analysis-based techniques

2.2.1 Dataflow Analysis-Based Techniques

Many researchers proposed techniques fall under this category, basically it uses the 'definition –use' variables. These variables can be used in two different ways one to compute operation like multiplication etc., and other is for direction or the execution path. Harrold[7] and sofa have done some work this and they proposed a RTS technique that can be applied to multiple programs, for they individually trace the program change to select test case and they repeat till all the changes have been covered.

Critical Evaluation

The major challenge here is these techniques cannot control the dependency among the elements of the program, hence its these techniques are unsafe to use.

2.2.2 Slicing-Based Techniques

Slicing based RTS techniques have been proposed by Agrawal [9], the objective of these techniques is to select a

https://doi.org/10.15623/ijret.2018.0712009 Received: 22-10-2018, Accepted: 01-12-2018, Published: 24-12-2018

set of test cases executed on a modified program gives a different output.

It's basically about comparing the expanded programs in case of any internal calls to the procedure version of the two programs, the schematic differences between these two programs will be analyzed to find the regression techniques

There are 4 slicing techniques,
1. Execution slice
2. Dynamic slice
3. relevant slice
4. approximate relevance slice

These techniques have been explained in detail in the Agrawal article [9].

Critical Evaluation

These RTS techniques are precise as they exclude the test cases which produces a different output. However, Rothermel and Harlold [6] slicing techniques are not safe when the changes are due to deletion of code statements.

2.2.3 Module Level Firewall-Based Techniques

Fire wall is a virtual boundary that helps testing to limit the effected or modified modules, Leung and white [10] defined firewall as the set of all modified modules in a program along with those modules which interact with the modified modules.

The firewall technique is based on the data and control dependency among various modules of a program.

These techniques use a call graph to represent the control flow of a program, Module M1 is called ancestor of M2, if there exists a path in the call graph from Module M1 to M2, and M2 is called as descendant of the modified modules. Both direct ancestors and the direct descendants of the modified modules are also included in the defining the firewall to cover all the modified modules. The effected modules can be identified by using test coverage information.

Critical Evaluation

Firewall techniques are efficient because the technique considers only modified modules in the firewall, and narrows down the source code analysis to greater extent, However these techniques are not safe as it does not select testcases from outside the firewall and that may execute the affected modules with in the firewall[10]

2.2.4 Differencing-Based Approaches

Differencing-based approaches as name suggested, these are dependent on the differences between the programs, original program and the modified program [11],These techniques can be classified into two major classes

2.2.4.1 Modified Code Entity-Based Technique

In this technique, A program code is decomposed into functional and nonfunctional code.

A functional code entity is executable like function or a statement and nonfunctional code is not executable like global variables or a macro, the test coverage information is analyzed to determine the set of executables that are touched by each test case.

2.2.4.2 Textual Differences-Based Technique

This technique does not use intermediate representation of the programming, in this technique program is converted to a differential i.e. canonical form before comparison. After modification of the program should also follow the same syntactic and formatting guidelines, this technique compares the canonical form of the original and modified programs

Critical Evaluation

It is safe technique as the affected code drove the selection of the test cases, however it is imprecise, if the code changes are arbitrary. As it compares the differences between the syntax, so there could be chances of getting repeated test cases or unwanted test cases

2.2.5 Control Flow Analysis-Based Techniques

A few techniques have been proposed which analyze the control flow of the input programs for selecting the regression test cases they are of the below types

2.2.5.1 Cluster Identification Technique

Laski and Szermer [12] have proposed the cluster identification technique. Cluster is a block which has a single entry and exit that changes from one version to another version.
Cluster uses control dependence information of original and modified procedure to find a cluster.

2.2.5.2 Graph Walk-Based Technique

Rothermel and harrold [6] have proposed graph walk-based technique which is based on the traversal of control flow graphs of original and modified program. In this CFGs G and G1 for program P and P1are constructed.

Then by instrumenting P, the execution trace of each test case t, ET(p(T)) is recorded by depth first traversal.

This technique observers the program statements along with identically labelled edges of G and G1 are equivalent or not. The non-identical nodes are identified as dangerous edges. Form the test suite all the test cases which can execute the dangerous edges will be selected.

https://doi.org/10.15623/ijret.2018.0712009 Received: 22-10-2018, Accepted: 01-12-2018, Published: 24-12-2018

2.2.5.3 DFA Model-Based Approach

Ball [14] has proposed DFA model, it constructs DFA M from CFG G based on the below conditions:
1. Each node v in G 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 the DFA accepts all possible paths of 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

These techniques are safe. However, these techniques are not driven by the modification of the program rather execution of the cluster and more over it is more expensive in terms of computation

3. REGRESSION TEST SUITE PRIORITIZATION TECHNIQUES

Test case prioritization does not minimize or filter the test cases, the test engineer executes all the given test cases in the order given by the prioritization technique or approach.

Once the selection of the test case have been done, then need to identify the order of the test cases to be executed, the order is important as defect needs to be uncovered as soon as possible in order give more time to analyze the defect and fix from the developers, uncovering defects at the end of the testing may lead to unnecessary pressure on the developer adding to that for various reason testing may need to stop at any point of time by that time defects should get uncovered as much as possible.

The first formal definition of the prioritization problem and metrics were provided by Rothermel et al. and Elbaum et al. And Kamna and Yudhvir singh have mentioned a novel classification “3CMDHO” of test case prioritization techniques[14].

3.1 Cost Based Techniques

These techniques are cost based techniques, this includes the cost of analysis and prioritization. Many researchers have proposed many techniques [15,16]

When the basic metric APFD applied to the model, it has two limitations [17],the proposed model considers all the faults to be equally severe, and it assumes the every costs the same resource. Elbaum extended the basic metric APFD to APFDc so that the metric can consider not only the rate of fault detection but also the severity of detected faults and their expenses [16].

Yoo and Harman [15] studied test suite minimization, their multi objective approach is relevant to cost based approach.

Although time constraints impact will be different on these techniques, it is always advisable to use some prioritization techniques

3.2 Distribution Based Approach

These techniques reduce and prioritize the test cases on the basis of profiles of the test cases in the multi-dimensional profile space. [14].

Test cases can be classified into classes having the similar profiles as per the properties. The grouping can help whether there are any redundant test cases, and may suggest some extreme or rare conditions that may cause failure, and uncommon usage behaviors can also be indicated. In the given class anyone of the test case is sufficient to execute, however later points are about unusual behaviors and conditions, so the test cases can be given high precedence so that fault can be uncovered early.

3.3 History Based Approach

Sherriff et al. proposed a matrix analysis called singular value decompositions [18], this prioritization depends on three elements: Association clusters, relationship between test cases and files and a modification vector. In order to fix a defect, often a few files get modified together, this approach cluster those files into same association cluster, and each file is also associated with the test cases that effect. And a system modification is represented as a vector in which the value indicates a particular file has been modified or not. This helps in assigning a priority to testcases which are associated with the clustered files.

3.4 Requirement Based approach

Srikanth et al. proposed [19] the requirement based test case prioritization approach, Test cases are mapped to the software requirements that are tested by the testers and then prioritize the requirements considering various factors like complexity, customer assigned priority etc. And weights will be calculated for each test case and re order them to initiate the order of execution, highest weight will be executed first. And another simple approach for test case prioritization which was proposed earlier by establishing the traceability matrix between the test cases and their requirements.

3.5 Coverage Based Approach

The goal of the test case prioritization is to achieve a higher fault detection rate with in less time to maximize the impact. This will be observed red by the metric structural coverage. This technique is to increase the chances of uncovering faults early by maximizing the coverage for each test case. Here coverage refers to code coverage.

Rothermel et al. proposed Fault exposing potential-total and Fault exposing potential- additional

https://doi.org/10.15623/ijret.2018.0712009 Received: 22-10-2018, Accepted: 01-12-2018, Published: 24-12-2018

The branch total approach prioritizes the test cases refers to the number of branches covered by the test case, and branch additional number of branches covered by individual test cases, same with statement -total and statement additional approaches.

The fault exposing potential is a metric measured using a program mutation, a seeded change will be introduced and create a new modified program and this mutant will be killed or uncovered by the testcase, by revealing the difference between the original and modified program. The score will be total mutants uncovered by total mutants introduced.

Li et al. applied various meta heuristics for a test case prioritization [20], they have considered hill climbing algorithm, genetic algorithm, a greedy algorithm, and additional greedy algorithm and compared with random prioritization. They concluded that additional greedy algorithms most efficient in general and they have measured the efficiency based on APBC average percentage of block coverage instead of Average percentage of fault detection.

4. CONCLUSION

In this paper, we presented the work done in the areas of test case selection and prioritization with brief explanation of each technique, this paper also describes that test case selection and prioritization are closely related and provides a holistic view of available techniques in the literature. It is also clear that so much research has been done on this topic and still researchers are interested in taking the techniques to the next level by incorporating latest techniques.

Regression testing is very expensive, and many techniques have been proposed for reducing the cost of regression testing. And many researchers proposed many different techniques and the challenge is to select the most cost-effective technique for a regression testing session. And a further works needs to be done in choosing the test cost effective test selection and prioritization techniques using latest techniques, and Adaptive test prioritization strategies are one of them.

REFERENCES

[1] https://www.utest.com/articles/reducing-the-cost-of-regression-testing-in-agile-development

[2] Marwah Alian , Dima Suleiman, Adnan Shaout Test Case Reduction Techniques – Survey (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 5, 2016

[3] https://www.justinmind.com/blog/reducing-software-development-lifecycle-costs-5-top-tips/

[4] http://www0.cs.ucl.ac.uk/staff/M.Harman/stvr-shin-survey.pdf

[5] http://glossary.istqb.org/search/regression%20testing

[6] G. Rothermel and M. Harrold. Selecting tests and identifying test coverage requirements for modified software. In Proceedings of the International Symposium on Software Testing and Analysis, pages 169–184, August 1994.

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

[8] Swarnendu Biswas and Rajib Mall,Regression Test Selection Techniques: A Survey,Informatica 35 (2011) 289–321

[9] H Agrawal,J Horgan,E Krauser, and S.London "Incremental regression testing",in IEEE International conference on Software Maintenance,1993.

[10] 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,pages 290–300, November 1990.

[11] Y. Chen, D. Rosenblum, and K. Vo. TestTube: A system for selective regression testing. In Proceedingsof the 16th International Conference on Software Engineering, pages 211–222, May 1994.

[12] J. Laski and W. Szermer. Identification of program modifications and its applications in software maintenance. In Proceedings of the Conference on Software Maintenance, pages 282–290, November 1992.

[13] 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, pages 134–142, 1998

[14] Kamna Solanki,Yudhvir Sing,"Novel Classification of Test Case Prioritization Techniques",International Jaournal of Computer Applications-August 2014.

[15] Yoo S, Harman M. Pareto efficient multi-objective test case selection. Proceedings of International Symposium on Software Testing and Analysis (ISSTA 2007), ACM Press, 2007; 140–150.

[16] Elbaum SG, Malishevsky AG, Rothermel G. Incorporating varying test costs and fault severities into test caseprioritization. Proceedings of the International Conference on Software Engineering (ICSE 2001), ACM Press, 2001; 329–338.

[17] S. Yoo , M. Harman, Regression testing minimization, selection and prioritization: a survey, Software Testing, Verification & Reliability, v.22 n.2, p.67-120, March 2012 [doi>10.1002/stv.430]

[18] Sherriff M, Lake M, Williams L. Prioritization of regression tests using singular value decomposition with empirical change records. Proceedings of the The 18th IEEE International Symposium on Software Reliability (ISSRE 2007), IEEE Computer Society: Washington, DC, USA, 2007; 81–90.

[19] Hema Srikanth, Laurie Williams, Jason Osborne, "System Test Case Prioritization of New and Regression Test Cases," in 4th International Symposium on Empirical Software Engineering, 2005, pp. 62-71.

https://doi.org/10.15623/ijret.2018.0712009 Received: 22-10-2018, Accepted: 01-12-2018, Published: 24-12-2018

[20] Li Z, Harman M, Hierons RM. Search Algorithms for Regression Test Case Prioritization. IEEE Transactions on Software Engineering 2007; 33(4):225–237.

BIOGRAPHIE

Kiran Jammalamadaka, received his MCA from Acharya Nagarjuna University, India. He is currently associated with GE India as a Software Manager. His interests include Software engineering, Agile->Scrum and automated debugging.