Indian Journal of Science and Technology

An Optimal Technique for Reducing the Effort of Regression Test

T. Prem Jacob[1*] and T. Ravi[2]

[1]Research Scholar, Sathyabama University, Chennai-119, India; premjac@yahoo.com
[2]Principal, Srinivasa Institute of Engineering & Technology, Chennai- 56, India; travi675@yahoo.com

Abstract

Regression test selection techniques are proposed often but are many times inaccurate when used with larger systems. The proposed new selection technique will be safer, more precise, and can handle the object-oriented features even in larger systems through its phases. Selecting the subset of the test case from the existing test suite is an important problem in regression testing and is addressed in the regression test selection technique. Safe regression test selection technique selects and identifies the program parts that are affected by the change. The test selection is performed by matching the identified change information with the coverage information. A tool is implemented that reduces the testing effort efficiently and the result shows that it can achieve considerable savings in the regression testing time.

Keywords: Test Selection, Testing, Software Maintenance, Regression Testing, Software Evaluation.

1. Introduction

Regression testing is an expensive task in the maintenance of retesting the modified programs that helps to ensure that those changes performed on the software will not have any negative impact on the reliability of the software. As the software evolves the regression testing will be applied to the modified software versions which provides the confidence that changed parts may behave as expected and the changes which have not introduced any unexpected faults which are known as the regression faults. Even for the small changes that has been made the whole test suite has to be re-executed in the program that has been modified. In the scenario of the regression testing, D is the developer of the software product i.e. P, where the latest version is tested using the test suite i.e. T which are then released. Once the latest version is tested the developer may have to perform some modifications. At the time of maintenance, the developer D modified the program P for fixing the faults and adding any new features [1]. Since the client can change the requirements at any stage. After the changes are performed, the developer will obtain the new software version P′ and it has to be regression tested before committing those changes to the repository before the release of the software product.

If the retest all approach is used will consume large amount of time and the resources which in turn will increase the cost of performing the regression testing. The developer has to face the problem for the selection of the appropriate subset i.e. T′ of the test T for rerunning on the program P` and this process is called as an RTST (Regression Test Selection Technique). This selection technique will select only the subset of the test case from the entire test suite. RTST has to rerun each of the test cases from the test suite T on the program P′, to select the test case T′ which is equal to the test suite T. As we have discussed earlier retest all approach where all the test cases are rerun in test suite T which will be expensive. Suppose

*Corresponding author:
T. Prem Jacob (premjac@yahoo.com)

if there is small changes between the program P and the modified version P′, which may lead to unnecessary effort. So we have to select a better technique. RTST technique will use information's from program P, modified versions P′, test suite T for selecting a subset of T for testing P′. With this technique considerable savings can be obtained since the testing effort can be reduced when compared to the retest all approach.

The RTST technique that selects each test case from the test suite which may behave differently in modified and original software versions [2]. Precision and efficiency of the RTST technique which are related to the granularity level in which these techniques operates. For the RTST technique safety is more important which guarantees that the subset T′, contains all the test cases which may able to reveal the regression faults that occur in P′. The techniques which work in a higher abstraction level. If the RTST technique has to be more cost effective than the cost for performing rerunning the subset of the selected test case should be less when compared to the cost for rerunning the test suite completely. In the cost models for the regression testing cost terms depend on some specific scenario. The studies have shown that the existing techniques will be more cost-effective and then studies are performed using the subjects of some limited size [3]. Consider the example for the test suite which requires the human interaction, savings should account human effort to be saved. The precise technique will be too expensive for using on larger systems. The safe techniques that exist are not so cost effective if it is applied to the large software and the efficient techniques will be more imprecise which achieves little savings in the testing effort.

A new algorithm for RTST is presented that handles the features of object-oriented language which will be more precise and safe for the larger systems. The algorithm has two phases that are partitioning, selection. In partitioning phase a high-level of the graph representation for the program P, P′ is created. The analysis goal is for identifying, based on the information in the changed classes which interface the program parts P, P′ for further analysing. A detailed graph is built in the algorithm's selection phase for the identified program parts P, P′. The graphs are analysed for identifying the differences between programs which selects for the rerun of the test cases that are in T, which traverse these changes. Although these techniques which are defined in the Java language and it can also be adapted for object-oriented language.

2. Related Work

There are different techniques developed for the regression testing of the software. Ren technique can identify the test cases that are affected by the code change, which test cases can affect each of the test case. These techniques focus only on the unit test cases. Most of the approaches will be based on the identification of the differences in the old and the new versions of program and the selection is based on the difference with the coverage information of the test case.

Our technique can able to handle the object oriented features and it does not require analyzing the entire system and requires only the identified partition at the initial phase. The advantage of our partitioning can able to compute the simple dependences and it postpones those expensive analysis at the second phase.

3. The Two Phases of the RTST Techniques

In this technique we combine the RTST effectiveness which is precise, it may be not efficient for the larger systems with the efficiency of the techniques which works on a higher level and may be imprecise. This can be done using the two phase approach which performs an initial analysis at a high level which identifies the system parts that has to be analysed further, in-depth analysis of the parts identified that selects those test cases which are in T are rerun. In partitioning phase where the technique can analyse those programs for identifying aggregation, hierarchical and it uses the relationship between the interface and the classes [4]. This technique which uses those information's about those relationships and the information about the classes and the interfaces have changed syntactically, for identifying the program parts which may get affected by those changes between the program P, P′.

The output of this phase will be the subset of those interfaces and the classes in those programs. In selection phase this technique which the input as the partition that are identified in the first phase. The test selection at the edge level selects the test case just by analysing the changes and the coverage information's at a level of flow of the control between the statements [5, 6]. Due to the partitioning that is performed at the phase one, expensive analysis, low level is performed to the small fraction in the whole program[7, 8]. Since a partial portion of the programs is analysed, it is performed under safe assumptions because, the partitioning will identify all the interfaces and the classes whose behaviour changes due to modifications to the program P, Edge-level technique which is used in selection phase will be safe.

3.1 The Partitioning Phase

This approach first phase will performs the high-level analysis for program P and the modified program P′ and to the relationships between the interface and the classes from the program.

3.2 The Syntactic Change-accounting

Without losing generality, we have classified the program changes as two groups, changes in statement level, change in declaration level. The changes in statement level consist of addition, modification, deletion of the executable statements. This change can be handled easily by the RTST, each of the test case which traverse the newly modified part of the code has to be re-executed.

A change in the declaration level consists of modifications in a declaration. The example of these changes will be the modifications in the type of the variable, removal or addition of any method, the inheritance relationship modification, the change of the type in the catch clause, the modifiers list change. These changes will be problematic for the RTST than the changes in the system level because they can affect program behaviour indirectly in the non-obvious way. Changes in the declaration level have complex effects compared to the changes in the statement level and if it is not handled suitably will cause the RTST technique unsafe, imprecise or both [9].

3.3 The Partitioning Algorithm

The input for the algorithm is set of the syntactic-changed types in C, two Interclass Relation Graph's, original version, modified versions of the program.

Step 1: Add the partition part which involves the changed types
Step 2: Add each type to a temporary set
Step 3: Add the types in the temporary set of partition
Step 4: Return partition

3.4 The Selection Phase

In this second phase we compute the change information just by analysing those types that are identified in the first phase, then we perform the test selection just by matching change information computed with the coverage information.

4. Performing the Test Selection

Whenever testing the program P, the testers will measure their coverage of the test suite i.e. T, for accessing the test suite adequacy. Coverage will be computed for the program entities like edges, statements. For each of the test case t which is in T the information is recorded on which entities to the program P that are executed by the test case t. Those coverage information that is collected automatically, using the coverage tools and it is represented as the coverage matrix, like one row for each entity, one column for each test case [10]. From the investigation of the efficiency and effectiveness of this technique it is applied to the medium and the larger systems. We present a tool DejaVu, which implements this technique on the set of those subjects. From the study we have investigated that how much percentage of program which is under test is selected by the partitioning technique, and how the overall RTST cost gets affected [11, 12]. We have investigated how much we have gained in precision when this technique operates at high abstraction level and the overall savings that our technique has achieved in the process of regression testing.

The DejaVu architecture consists of three components InsECT, DEI, Selector as in figure 1. The InsECT is an extensible, modular, coverage analyzer developed using java. It is used to gather the information on edge-coverage of the program when it is executed against the test suite. Dangerous Edge Identifier (DEI) and the Selector which is used to implement the two phased technique.

As subjects we have used three medium to large sized programs like JABA, DAIKON, JBoss. From these three programs five consecutive version are extracted and we stimulated the way the regression testing will occur. In the experimental design we have modified the DejaVu so that once the partition is identified at phase1 and it skips the phase 2, in turn selects all the test cases by using the partition. For these techniques we have measured the test suite selected, execution time for the test suite selected, the analysis time.

From the empirical studies that are performed using the tool on Java subjects that ranges between 70 to 600 KLOC. From these studies we have shown that our technique is more efficient compared to the precise existing

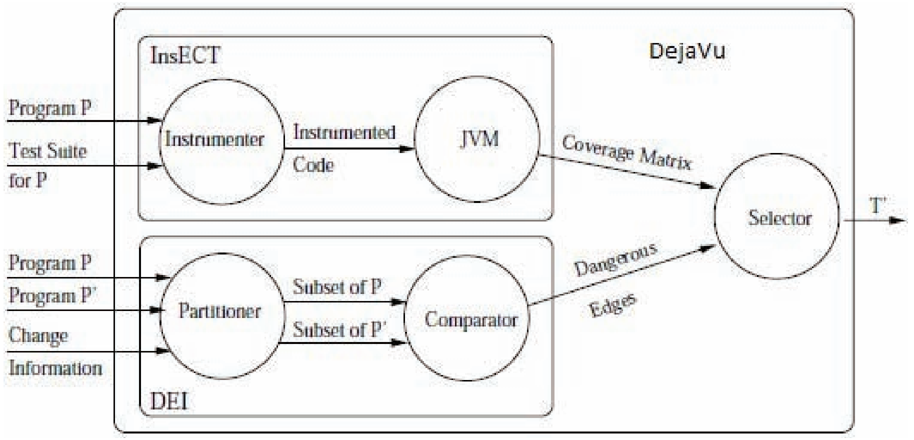

Figure 1. Architecture of DejaVu regression-test-selection-system.

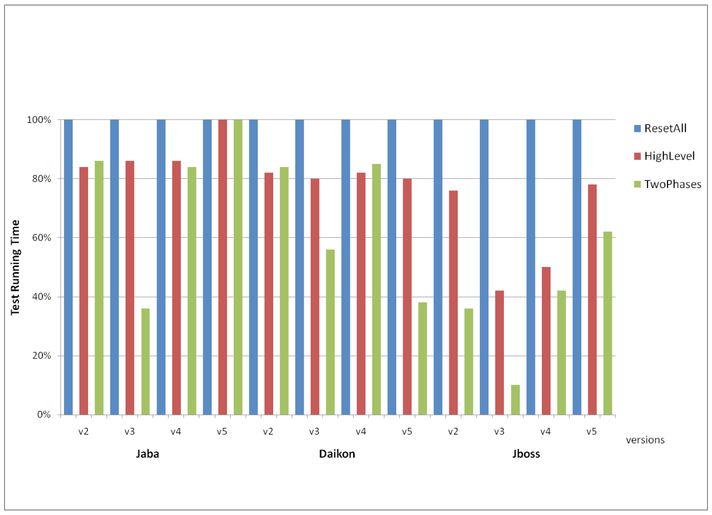

Figure 2. The Overall time taken for the regression testing process.

techniques, which operates to a fine abstraction level i.e. 89% on the average of those average subjects. It also showed the selection have achieved some considerable savings in the overall time for regression testing. Of these three subjects (retest all, high level, two phases) this technique has saved an average 19%, 36%, 63% in the time taken for regression testing.

5. Results and Discussion

We have empirically investigated the efficiency and effectiveness of our technique by applying to some larger systems and by developing the tool to implement this technique, we have performed a study on these subjects. In this study we have explored how much percentage will be selected by the partitioning technique from the program and analysed how the RTST costs gets affected by this, how much precision is gained, the savings that is achieved by our technique in the process of regression testing. The advantage of our partitioning can able to compute the simple dependences and it postpones those expensive analysis at the second phase.

6. Conclusions

From the new technique presented for the test selection in the Java software which is designed for larger systems. One technique which is based generally on the two phase approach, where the phase one that performs fast, high level analysis for identifying the various parts of the system which may get affected by those changes. In the phase two that performs the low level analysis of the parts for performing the test selection precisely. This paper performed the study on three medium to larger systems subjects for investigating the effectiveness and efficiency of our technique. These studies have produced an encouraging results, which can able to produce some considerable time saved at the time of regression testing as in figure 2. The results led us in an interesting direction of research for the further improvement in our RTST technique. In the future work we can work on for improving efficiency of the tool used and performing the controlled experiment for the large versions.

7. References

1. Rothermel G, Untch R H et al. (1999). Test case prioritization: an empirical study, Proceedings of the International Conference on Software Maintenance, 179–188.
2. Jacob T P (2013). Regression testing: Tabu search technique for code coverage, Indian Journal of Computer Science and Engineering, vol 4, No.3, 208–215.
3. Elbaum S, Rothermel G et al. (2004). Selecting a cost-effective test case prioritization technique, Software Quality Control, vol 12, No. 3, 185–210.
4. Jacob T P, and Ravi T (2013). Optimal regression test case prioritization using genetic algorithm, Life Science Journal, vol 10(3), 1021–1033.
5. Rothermel G, and Harrold M J (1997). A safe, efficient regression test selection technique. ACM TOSEM, vol 6(2), 173–210.
6. Bible J, Rothermel G et al. (2001). A comparative study of coarse and fine-grained safe regression test selection techniques. ACM TOSEM, vol 10(2), 149–183.
7. Walcott K R, Soffa M L et al. (2006). Time-aware test suite prioritization, International Symposium on Software Testing and Analysis, 1–11.
8. Do H, Elbaum S G et al. (2005). Supporting controlled experimentation with testing techniques: an infrastructure and its potential impact, Empirical Software Engineering, vol 10(4), 405–435.
9. Li Z, Harman M et al. (2007). Search algorithm for regression test case prioritization, IEEE Transactions on Software Engineering, vol 33, No. 4, 5–7.
10. Jeffrey D, and Gupta N (2007). Improving fault detection capability by selectively retaining test cases during test suite reduction, IEEE Transactions on software Engineering, vol 33, No. 2, 122–127.
11. Li Z, Harman M et al. (2007). Search algorithms for regression test case prioritization, IEEE Transactions on Software Engineering, vol 33, No. 4, 225–237.
12. Kim J M, and Porter A (2002). A history-based test prioritization technique for regression testing in resource constrained environments, Proceedings of the 24th International Conference on Software Engineering, 119–129.

8. Authors Biography

T. Prem Jacob received the B.E degree in Computer Science and Engineering from C.S.I Institute of Technology, Manonmaniam Sundaranar University, Nagercoil, India in 2004 and M.E degree in Computer Science and Engineering from Sathyabama University, Chennai, India in 2006, where he is currently working towards the Ph.D. degree in Computer Science and Engineering at Sathyabama University, Chennai, India. He is an Assistant Professor of Computer Science and Engineering in Sathyabama University and he has more than 7 Years of Teaching Experience. He has participated and presented many Research Papers in International and National Conferences. His area of interests includes Software Engineering, Data mining and Data warehouse.

Dr. T. Ravi, Principal of Srinivasa college of Engineering & Technology, Chennai. He has graduated in computer science and Engineering from Madurai Kamaraj University, Masters and Ph.D in computer Science and Engineering from Jadavpur University, Kolkata. He has more than 20 years of teaching experience in various engineering institutions in Tamil Nadu. More than 25 research papers are published in International & National Journals and conferences and also 5 text books are published through various publications. He is the Recognised Research Supervisor in Anna University and Sathyabama University Chennai and MS university, Tirunelveli.