Regression Testing: Techniques for Test Suite Minimization

Amita\(^a\), Kamal Ranga\(^b\)

\(^a\)Research Scholar, Department of Computer Science and Engineering, Ganga Institute of Technology & Management, Kablana
\(^b\)Assistant Professor, Department of Computer Science and Engineering, Ganga Institute of Technology & Management, Kablana

ABSTRACT

Software testing occurs simultaneously during the software development to detect errors as early as possible and to guarantee that changes made in software did not change the system negatively. However, during the development phase, the test suite is efficient and tends to increase in size. Due to the resource and time constraints for re-executing large test suites, it is important to develop techniques to reduce the effort of regression testing. Several approaches have been studied to reduce the effort of regression testing: test suite minimization, selection, and prioritization. Test suite minimization techniques aim at identifying and eliminating redundant test cases from the suite. Test suite selection techniques identify a subset of test cases from suite, necessary to re-test the changes in the software. Test suite prioritization techniques schedule test cases for execution in an order to increase the early fault detection.

Keywords - Black box, Regression testing

1. INTRODUCTION

Regression testing is performed when changes are made to existing software; the purpose of regression testing is to provide confidence that the newly introduced changes do not obstruct the behaviours of the existing, unchanged part of the software. It is a complex procedure that is all the more challenging because of some of the recent trends in software development paradigms. For example, the component based software development method tends to result in use of many black-box components, often adopted from a third-party. Any change in the third-party components may interfere with the rest of the software system, yet it is hard to perform regression testing because the internals of the third-party components are not known to their users. The shorter life-cycle of software development, such as the one suggested by the agile programming discipline, also imposes restrictions and constraints on how regression testing can be performed within limited resources.
Naturally, the most straightforward approach to this problem is to simply execute all the existing test cases in the test suite; this is called a retest-all approach. However, as software means it may be prohibitively expensive to execute the entire test suite. This limitation forces consideration of techniques that seek to reduce the effort required for regression testing in various ways. A number of different methods have been studied to aid the regression testing process. The three major branches include test suite minimisation, test case selection and test case prioritisation. Test suite minimisation 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 prioritisation concerns the identification of the ‘ideal’ ordering of test cases that maximises desirable properties, such as early fault detection. Existing empirical studies show that the application of these techniques can be cost-effective.

### 1.1 Multi-Objective Regression Testing
Regression testing is a complex and costly process that may involve multiple objectives and constraints. For example, the cost of executing a test case is usually measured as the time taken to execute the test case. However, there may be a series of different costs involved in executing a test case, such as setting up the environment or preparing a test input, each of which may be subject to a different constraint. Existing techniques also assume that test cases can be executed in any given order without any change to the cost of execution, which seems unrealistic. Test cases may have dependency relations between them. It may also be possible to lower the cost of execution by grouping test cases that share the same test environment, thereby saving set-up time.

Considering the complexity of real-world regression testing, existing representations of problems in regression testing may be over-simplistic. Indeed, most of the published empirical studies rely on relatively small-scale academic examples. Even when real-world programs are studied, they tend to be individual programs, not a software system as a whole. Larger software systems do not simply entail larger problem size; they may denote a different level of complexity.

### 1.2 Problem of Test Oracle and Its Cost
Test oracles present a set of challenging problems for software testing. It is difficult to generate them automatically, they often require human efforts to verify and the cost of this effort is hard to estimate and measure. The oracle cost has been considered as a part of cost models [17], but has not been considered as
Regression Testing: Techniques for Test Suite Minimization

a part of the process of minimisation, selection and prioritisation itself. Since regression testing techniques seek to efficiently re-use existing test cases, information about the cost of verifying the output observed with the existing test suite may be collected across versions. This can be incorporated into the existing regression testing techniques.

While a test oracle and its cost may be seen as yet another additional objective that can be considered using a multi-objective approach, it can be argued that the test oracle will present many interesting and exciting research questions in the context of regression testing and, thus, deserves a special treatment in its own right. This is because, compared to other testing cost such as the physical execution time of test cases, the test oracle cost is closely related to the quality of testing. Moreover, unlike some costs that can be reduced by using more advanced hardware, the cost of oracle verification derives from human effort and is, therefore, harder to reduce. These characteristics make the issues related to test oracles challenging but interesting research subjects.

1.3 Tool Support

Closely related to the issue of technology transfer is the issue of tool support. Without readily Available tools that implement regression testing techniques, practical adoption will remain limited. One potential difficulty of providing tool support is the fact that, unlike unit testing for which there exists a series of frameworks based on the x Unit architecture, there is not a common framework for the regression testing process in general. The closest to a common ground for regression testing would be an Integrated Development Environment (IDE), such as Eclipse, with which the x Unit architecture is already integrated successfully. A good starting point for regression testing techniques may be the management framework of unit test cases, built upon x Unit architecture and IDEs.

2. BACKGROUND

This section introduces the basic concepts and definitions that form a nomenclature of regression testing and minimisation, selection and prioritisation techniques.

2.1 Regression Testing

Regression testing is performed between two different versions of software in order to provide confidence that the recently introduced features of the System under Test (SUT) do not interfere with the existing features. While the exact details of the modifications made to SUT will often be available, they may not
be easily accessible in some cases. For example, when the new version is written in a different programming language or when the source code is unavailable, adaptation data will be unavailable.

The following notations are used to describe concepts in the context of regression testing. Let \( P \) be the current version of the program under test, and \( P' \) be the next version of \( P \). Let \( S \) be the current set of specifications for \( P \), and \( S' \) be the set of specifications for \( P' \). \( T \) is the existing test suite. Individual test cases will be denoted by lower case: \( t \). \( P(t) \) stands for the execution of \( P \) using \( t \) as input.

2.2. Distinction between Classes of Techniques

It is necessary at this point to establish a clear terminological distinction between the different classes of techniques described in the paper. Test suite minimisation techniques look for reduce the size of a test suite by eliminating redundant test cases from the test suite. Minimisation is sometimes also called ‘test suite reduction’, meaning that the elimination is permanent. However, these two concepts are essentially interchangeable because all reduction techniques can be used to make a temporary subset of the test suite, whereas any minimisation techniques can be used to permanently eliminate test cases. More formally, following Rothermel et al. [8], the test suite minimisation is defined as follows:

**Definition1. Test Suite Minimisation Problem**

**Given:** A test suite, \( T \), a set of test requirements \( \{r_1, \ldots, r_n\} \), that must be satisfied to provide the desired ‘adequate’ testing of the program, and subsets of \( T \), \( T_1, \ldots, T_n \), one associated with each of the \( r_i \)s such that any one of the test cases \( t_j \) belonging to \( T_i \) can be used to achieve requirement \( r_i \).

**Problem:** Find a representative set, \( T' \), of test cases from \( T \) that satisfies all \( r_i \)s.

The testing criterion is satisfied when every test requirement in \( \{r_1, \ldots, r_n\} \) is satisfied. A test requirement, \( r_i \), is satisfied by any test case, \( t_j \), that belongs to the \( T_i \), a subset of \( T \). Therefore, the representative set of test cases is the hitting set of the \( T_i \)s. Furthermore, in order to maximise the effect of minimisation, \( T' \) should be the minimal hitting set of the \( T_i \)s. The minimal hitting set problem is an NP-complete problem as is the dual problem of the minimal set cover problem [9].

While test case selection techniques also seek to reduce the size of a test suite, the majority of selection techniques are modification-aware. That is, the selection is not only temporary (i.e. specific to the current version of the program), but also focused on the identification of the modified parts of the program. Test cases are selected because they are relevant to the changed parts of the SUT, which typically involves a
Regression Testing: Techniques for Test Suite Minimization

white-box static analysis of the program code. Throughout this, the meaning of ‘test case selection problem’ is restricted to this modification-aware problem.

3. LITERATURE SURVEY

In year 1997, Wong, W.E.; Bell core, Morristown, NJ, USA; Horgan, J.R.; London, S.; Agrawal, H. performed a work,” A study of effective regression testing in practice ” The purpose of regression testing is to ensure that changes made to software, such as adding new features or modifying existing features, have not adversely affected features of the software that should not change. Regression testing is usually performed by running some, or all, of the test cases created to test modifications in previous versions of the software. Many techniques have been reported on how to select regression tests so that the number of test cases does not grow too large as the software evolves.

In year 2001, Rothermel,G. performed a work, “Prioritizing test cases for regression testing” Test case prioritization techniques schedule test cases for execution in an order that attempts to increase their effectiveness at meeting some performance goal. Various goals are possible; one involves rate of fault detection, a measure of how quickly faults are detected within the testing process. An improved rate of fault detection during testing can provide faster feedback on the system under test and let software engineers begin correcting faults earlier than might otherwise be possible. One application of prioritization techniques involves regression testing, the retesting of software following modifications; in this context, prioritization techniques can take advantage of information gathered about the previous execution of test cases to obtain test case orderings.

In year 2002, Jung-Min Kim, performed a work, “A history-based test prioritization technique for regression testing in resource constrained environments” Regression testing is an expensive and frequently executed maintenance process used to revalidate modified software. To improve it, regression test selection (RTS) techniques strive to lower costs without overly reducing effectiveness by carefully selecting a subset of the test suite. Under certain conditions, some can even guarantee that the selected test cases perform no worse than the original test suite. This ignores certain software development realities such as resource and time constraints that may prevent using RTS techniques as intended (e.g., regression testing must be done overnight, but RTS selection returns two days’ worth of tests). In practice, testers work around this by prioritizing the test cases and running only those that fit within existing constraints.

In year 2004, Rothermel,G. performed a work, “Selecting a Cost-Effective Test Case Prioritization Techniques” Regression testing is an expensive testing process used to validate modified software and
Regression Testing: Techniques for Test Suite Minimization

detect whether new faults have been introduced into previously tested code. To reduce the cost of regression testing, software testers may prioritize their test cases so that those which are more important, by some measure, are run earlier in the regression testing process. One goal of prioritization is to increase a test suite's rate of fault detection. Previous empirical studies have shown that several prioritization techniques can significantly improve rate of fault detection, but these studies have also shown that the effectiveness of these techniques varies considerably across various attributes of the program, test suites, and modifications being considered.

In year 2007, Cohen, M.B; Woolf, K.M, “Combinatorial Interaction Regression Testing: A Study of Test Case Generation and Prioritization.” Regression testing is an expensive part of the software maintenance process. Effective regression testing techniques select and order (or prioritize) test cases between successive releases of a program. However, selection and prioritization are dependent on the quality of the initial test suite. An effective and cost efficient test generation technique is combinatorial interaction testing, CIT, which systematically samples all t-way combinations of input parameters. Research on CIT, to date, has focused on single version software systems.

3.1 Discussions

In all these surveys’ of regression testing, test case prioritization technique, test case selection techniques are mostly used. In this 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 and test case prioritisation concerns the identification of the ‘ideal’ ordering of test cases that maximises desirable properties, such as early fault detection. Test suite minimisation techniques seek to reduce the size of a test suite by eliminating redundant test cases from the test suite.

4. CONCLUSIONS

It is also clear that the research community is moving towards assessment of the complex trade-offs and balances between different concerns, with an increase in work that considers the best way in which to incorporate multiple concerns (cost and value for instance) and to fully evaluate regression testing improvement techniques.

This focus on empirical methodology is one tentative sign that the field is beginning to mature. The trend analysis also indicates a rising profile of publication, providing evidence to support the claim that the field
Regression Testing: Techniques for Test Suite Minimization

continues to attract growing attention from the wider research community, which is a positive finding for those working on regression test case minimisation problems.

REFERENCES

Regression Testing: Techniques for Test Suite Minimization