RESEARCH ARTICLE OPEN ACCESS

An Approach for Usage and Dataflow Coverage in Regression Testing

Y. Laxmi Prasanna*
*(Department of Computer Science, Anurag Engineering College, Kodad)

ABSTRACT
Regression testing is the activity of retesting a program after it has been modified to gain confidence that existing, changed, and new parts of the program behave correctly. This activity is typically performed by rerunning, completely or partially, a set of existing test cases (i.e., its regression test suite).In this thesis we proposed a frame work based approach that covers the component usage and data flow variations between two versions. The approach of proposed model is a frame work that monitors the applications activities and flow of execution. This approach shows the statistical information of both versions such as execution flow statistics, execution time statistics, and coverage statistics.

Keywords: Regression testing , Regression Test Selection, Software Maintenance.

I. INTRODUCTION

Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of obsolete capabilities. These modifications in the software may cause the software to work incorrectly and may also affect the other parts of the software, so to prevent this Regression testing is performed. Regression testing is used to *revalidate* the modifications of the software. Regression testing is an expensive process in which test suites are executed ensuring that no new errors have been introduced into previously tested code. In section 2 of this paper we have broadly shown various types of regression testing techniques and further discussed classifications of these types given by various authors, then moving into the details of selective and prioritizing test cases for regression testing, discussing search algorithms for test case prioritization. In section 3 we have discussed the approaches which may be used to compare various regression testing techniques and challenges faced by these approaches.

II. REGRESSION TESTING

Regression testing is defined [1] as “the process of retesting the modified parts of the software and ensuring that no new errors have been introduced into previously tested code”. Let P be a program [2], let P′ be a modified version of P, and let T be a test suite for P. Regression testing consists of reusing T on P′, and determining where the new test cases are needed to effectively test code or functionality added to or changed in producing P′. There are various regression testing techniques (1) Retest all; (2) Regression Test Selection; (3) Test Case Prioritization; (4) Hybrid Approach. Figure 1 shows various regression testing techniques.

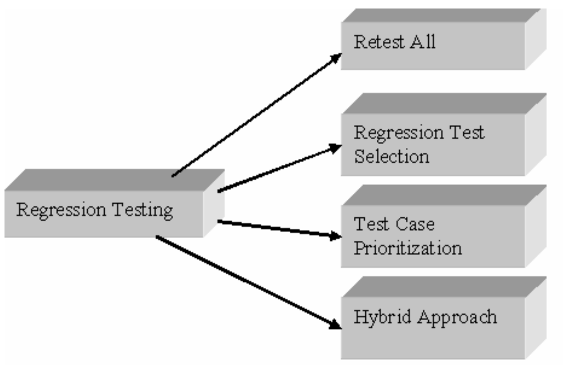

Fig1 Regression Testing Techniques

1. Retest

Retest all technique is very expensive as compared to techniques which will be discussed further as regression test suites are costly to execute in full as it require more time and budget.

2. Regression Test Selection (RTS)

Due to expensive nature of “retest all” technique, Regression Test Selection is performed. In this technique instead of rerunning the whole test suite we select a part of test suite to rerun if the cost of selecting a part of test suite is less than the cost of running the tests that RTS allows us to omit. RTS divides the existing test suite into (1) Reusable test cases; (2) Retestable test cases; (3) Obsolete test cases. In addition to this classification RTS may create new test cases that test the program for areas which are not covered by the existing test cases. RTS techniques are broadly classified into three categories [1].
1) Coverage techniques:_they take the test coverage criteria into account. They find coverable program parts that have been modified and select test cases that work on these parts.

2) Minimization techniques: they are similar to coverage techniques except that they select minimum set of test cases.
3) Safe techniques: they do not focus on criteria of coverage, in contrast they select all those test cases that produce different output with a modified program as compared to its original version.

3. Test Case Prioritization

This technique of regression testing prioritize the test cases so as to increase a test suite‘s rate of fault detection that is how quickly a test suite detects faults in the modified program to increase reliability. This is of two types:(1) General prioritization[3] which attempts to select an order of the test case that will be effective on average subsequent versions of software .(2)Version Specific prioritization which is concerned with particular version of the software.

4. Hybrid Approach

The fourth regression technique is the Hybrid Approach of both Regression Test Selection and Test Case Prioritization. There are number of researchers working on this approach and they have proposed many algorithms for it. For example,
1) Test Selection Algorithm: proposed by Aggarwal et al. Implementation of algorithm [4]: (a) Input (b) Test Selection algorithm: Adjust module and Reduce module (c) output.
2) Hybrid technique proposed by Wong et al_which combines minimization, modification and prioritization based selection using test history [5].
3) Hybrid technique proposed by Yogesh Singh et_al_is based on Regression Test Selection and Test Case Prioritization. The proposed algorithm in detail can be studied in [6].

III. EXISTING SYSTEM

Existing approaches like "Incremental program testing using program dependence graphs", "Semantics guided regression test cost reduction", "Program slicing-based regression testing techniques and "Selecting tests and "identifying test coverage requirements for modified software" address this problem by defining criteria that require exercising single control- or data-flow dependences related to program changes.

Considering the effects of changes on single control- and data-flow relations alone does not adequately exercise either the effects of software changes or the modified behavior induced by such changes.

In the existing system, the tool called recover takes input as source code of the application .As it takes source code as input it does not concentrate on behavioral differences of old and new versions of softwares. The demerits of existing system are it does not support polymorphic version of components and dependency injection, and inversion of control.

Research Objective:

The results gained from empirical study conducted and claimed in literature "Recomputing Coverage Information to Assist Regression Testing", thus, motivate the need for our technique that provides usage coverage and dataflow coverage as the software evolves without requiring rerunning of all test cases as each software change is made. With the motivation gained from the literature discussed I would like to propose a frame work based approach that covers the component usage and dataflow in new version. The approach of the proposed model is that it is a framework that monitors the applications activities and flow of execution during that application’s execution time. Where as in the case of tool called RECOVER discussed in literature “Recomputing Coverage Information to Assist Regression Testing” the input will be the source code of the application. The aim of our proposal is the version differences will be identified during the runtime that improvises the evaluation of the component usage coverage and dataflow coverage.

IV. PROPOSED SYSTEM

We need to concentrate on analyzing two versions of the same program. The structural difference between the two programs must be small relative to the size of the program: only a few lines of code or a few procedures in the program should be different. We would like to develop techniques that take advantage of the similarities between the two programs, rather than use existing techniques to analyze the programs independently and compare the results. Because our goals include finding unanticipated side effects of changes, we cannot assume that an existing regression test suite is able to find all interesting behavioral differences.

Regression testing finds differences in behavior that were anticipated by the designers (or testers) and specifically checked. While regression test selection is a useful technique for reducing the cost of testing, it cannot reveal new differences that are not already tested by the suite. We also would like to be able to analyze undocumented programs that may not have test suites.

We assume we have a generator capable of producing a differentiating test case, but that it is not reasonable to do an exhaustive search of the input space. It is not necessary for all generated inputs to be valid; the search will eliminate inputs that both programs consider to be errors. If the difference in behavior is small relative to the input space, and we have a generator that can produce the right inputs, the analysis problem becomes one of performing a directed search to find inputs which reveal behavioral differences.

Proposed Model

Regression testing is the activity of retesting a program after it has been modified to gain

confidence that existing, changed, and new parts of the program behave correctly. This activity is typically performed by rerunning, completely or partially, a set of existing test cases (i.e., its regression test suite).
Given a program P and a modified version of the program P, the regression test suite can reveal differences in behavior between P and Pand, thus, help developers discover errors caused by the changes or by unwanted side effects of the changes introduced in P. There is much research on making regression testing more efficient by
(1) Identifying test cases in a regression test suite that need not be rerun on the modified version of the software,
(2) Eliminating redundant test cases in a test suite according to given criteria ordering test cases in a test suite to help find defects earlier.
Little research, however, has focused on the effectiveness of the regression test suite with respect to the changes. To evaluate such effectiveness, it is necessary, when performing regression testing, to
(1) Check whether existing test suites are adequate for the changes introduced in a program and, if not,
(2) provide guidance for creating new test cases that specifically target the (intentionally or unintentionally) changed behavior of the program.

Existing approaches like "Incremental program testing using program dependence graphs", "Semantics guided regression test cost reduction", "Program slicing-based regression testing techniques and "Selecting tests and "identifying test coverage requirements for modified software" address this problem by defining criteria that require exercising single control- or data-flow dependences related to program changes.
Limits of the solutions exist:

Considering the effects of changes on single control- and data-flow relations alone does not adequately exercise either the effects of software changes or the modified behavior induced by such changes.

The literature "Recomputing Coverage Information to Assist Regression Testing" that considered as motivation proposed a tool called recover. This tool identifies the version differences to assist the selection of test cases that covers updates in new version during the regression testing.

The literature "Recomputing Coverage Information to Assist Regression Testing" presented a technique that provides updated coverage data for a modified program without running all test cases in the test suite that was developed for the original program and used for regression testing. The technique is safe and precise in that it computes exactly the same information as if all test cases in the test suite were rerun.

Approach: To conduct empirical study we will develop a tool using java aspect model that evaluates the component usage coverage and dataflow coverage of any java application. The resultant statistical report would helps to identify the significant version differences.

Block Diagram

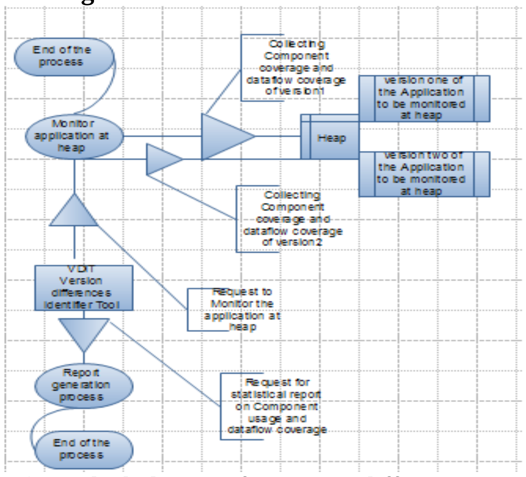
End of the process
Monitor application at heap
Collecting Component coverage and dataflow coverage of version1
version one of the Application to be monitored at heap
Heap
version two of the Application to be monitored at heap
Collecting Component coverage and dataflow coverage of version 2
VDIT Version differences Identifier Tool
Request to Monitor the application at heap
Report generation process
Request for statistical report on Component usage and dataflow coverage
End of the process

Fig 2 Block diagram for version differentiation

V. CONCLUSION

We conclude that novel framework helps to identify test case feasibility based on version differentials under regression testing. Recomputing coverage information assists regression testing. Recomputing is a frame work that extracts the component usage data flow coverage information from two concurrent versions of the software. This framework is accurate and scalable when compared to the tool called recoverage.

VI. FUTURE WORK

Further enhancement can be made by automating without selecting the updated coverage manually by the test engineer. By this we can still reduce the testing budget and maintenance.

In regression testing many techniques are proposed that provides an efficient way of selecting regression test suite without rerunning all test cases that was developed for old version. But still IT companies are not utilizing these techniques, because these techniques are not assuring completely and test engineers are still using manual testing and automation testing for safety.

REFERENCES

[1] K.K.Aggarwal & Yogesh Singh, *"Software Engineering Programs Documentation, Operating Procedures,"* New Age International Publishers, Revised Second Edition – 2005.
[2] Sebastian Elbaum, Praveen Kallakuri, Alexey G. Malishevsky, Gregg Rothermel, Satya Kanduri, *"Understanding the Effects of Changes on the Cost-Effectiveness of Regression Testing Techniques,"* Journal of Software Testing, Verification, and Reliability, 13(2) pages:65-83, June 2003.

[3] G. Rothermel, R.H. Untch, C. Chu, and M.J. Harrold, *"Prioritizing Test Cases for Regression Testing,"* IEEE Trans. Software Eng., vol. 27, no. 10, pages 929-948, Oct. 2001.

[4] K. K. Aggrawal, Yogesh Singh, A. Kaur, *" Code coverage based technique for prioritizing test cases for regression testing,"* ACM SIGSOFT Software Engineering Notes, vol 29 Issue 5 September 2004.

[5] W. E.Wong, J. [21] W. E.Wong, J. R. Horgan, S. London and H.Agrawal, *"A study of effective regression testing in practice,"* In Proceedings of the 8th IEEE International Symposium on Software Reliability Engineering (ISSRE' 97), pages 264-274, November 1997.

[6] Yogesh Singh, Arvinder Kaur, Bharti Suri, *"A new technique for version-specific test case selection and prioritization for regression testing,"* Journal of the CSI ,Vol. 36 No.4, pages 23-32, October-December 2006.

[7] Pavan Kumar Chittimalli and Mary Jean Harrold *"Recomputing Coverage Information toAssist Regression testing,"* IEEE Trans. Software Eng., vol. 35, no. 4, pages 452-469, July/Aug. 2009.