Volume 3, Issue 8, August 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Novel Fault Analysis Based Dynamic Programming Approach for Regression Testing

Neelam *
Student, N.C.College of Engineering Israna Panipat, Haryana, India

Sukhvir Singh
Associate Professor, N.C.College of Engineering Israna Panipat, Haryana, India

Abstract— Regression Testing is basically performed to identify and rectify the client problem after the implementation of software product in client environment. In such case, instead of performing the all testing stages in same way, an intelligent mechanism is been implemented to identify the optimized sequence of test generation. One of such approach is fault analysis based approach. It means the module having the fault will be tested first and relatively other modules will be tested. In this paper, at first the regression testing approach is explored and later on a fault analysis based dynamic programming approach is suggested to find optimal test sequence.

Keywords— Regression Testing, Fault Analysis, Optimal Sequence, Test Suite

I. INTRODUCTION

Regression testing is selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements. Regression tests are a subset of the original set of test cases. These test cases are re-run often, after any significant changes (bug fixes or enhancements) are made to the code. The purpose of running the regression test case is to make a "spot check" to examine whether the new code works properly and has not damaged any previously-working functionality by propagating unintended side effects. Most often, it is impractical to re-run all the test cases when changes are made. Since regression tests are run throughout the development cycle, there can be white box regression tests at the unit and integration levels and black box tests at the integration, function, system, and acceptance test levels. The following guidelines should be used when choosing a set of regression tests (also referred to as the regression test suite):

• Choose a representative sample of tests that exercise all the existing software functions;
• Choose tests that focus on the software components/functions that have been changed; and
• Choose additional test cases that focus on the software functions that are most likely to be affected by the change.

A subset of the regression test cases can be set aside as smoke tests. A smoke test is a group of test cases that establish that the system is stable and all major functionality is present and works under "normal" conditions. Smoke tests are often automated, and the selections of the test cases are broad in scope. The smoke tests might be run before deciding to proceed with further testing (why dedicate resources to testing if the system is very unstable). The purpose of smoke tests is to demonstrate stability, not to find bugs with the system.

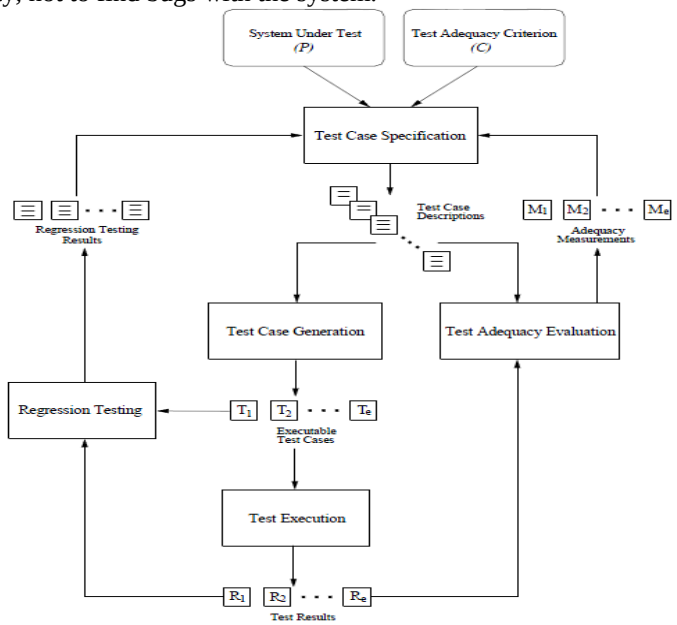

Figure 1 : Test Case Design

The most crucial phase in the software development life cycle is maintenance phase, in which the development team is supposed to maintain the software which is delivered to the clients by them. Software maintenance results for the reasons like error corrections, enhancement of capabilities, deletion of capabilities and optimization.

Software needs testing known as regression testing. 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.

II. LITERATURE REVIEW

Last *et al.* [1] proposed a new, GA-based approach to generate effective black-box test cases. From the case study, they conclude that the Fuzzy- Based Age Extension of Genetic Algorithm (FAexGA) is much more efficient for this problem than the two other evaluated algorithms (SimpleGA and GAVaPS). In this paper, they introduced a new, computationally intelligent approach to generation of effective test cases based on a novel, Fuzzy-Based Age Extension of Genetic Algorithms (FAexGA). The basic idea was to eliminate bad test cases that are unlikely to expose any error. The promising performance of the FAexGA based approach was demonstrated on testing a complex Boolean expression**.**

Engström [5] proposed a method to develop and evaluate strategies for improving system test selection in a SPL. *Method*: Initially industrial practices and research in both SPL testing and traditional regression test selection have been surveyed. Two systematic literature reviews, two industrial exploratory surveys and one industrial evaluation of a pragmatic test selection approach have been conducted. Jin and Orso [6] proposed a novel approach called Behavioral Regression Testing (BERT). Given two versions of a program, BERT identifies behavioral differences between the two versions through dynamical analysis, in three steps.

First, it generates a large number of test inputs that focus on the changed parts of the code. Second, it runs the generated test inputs on the old and new versions of the code and identifies differences in the tests' behavior. Third, it analyzes the identified differences and presents them to the developers.

Do, Mirarab [7] conducted a series of experiments to assess the effects of time constraints on the costs and benefits of prioritization techniques. Author first experiment manipulated time constraint levels and shows that time constraints to play a significant role in determining both the cost-effectiveness of prioritization and the relative cost-benefit trade-offs among techniques. Chun *et al.* [9] proposed performance analysts must manually analyze performance regression testing data to uncover performance regressions. The proposed approach was used to compare new test results against correlations pre-computed performance metrics extracted from performance regression testing repositories. Case studies show that our approach scales well to large industrial systems, and detected performance problems that are often overlooked by performance analysts. Zhang *et al.* [8] presented a new regression test selection technique by clustering the execution profiles of modification traversing test cases. Cluster analysis group program executions that had similar features, so that program behaviors can be well understood and test cases can be selected in a proper way to reduce the test suite effectively.

Duggal, Suri [12] described Regression testing is done in the maintenance phase of the software development life cycle to retest the software for the modifications it has undergone. Approximately 50% of the software cost is involved in the maintenance phase so researchers are working hard to come up with best results by developing new Regression Testing techniques so that the expenditure made in this phase can be reduced to some extent. This paper discussed Regression Testing techniques and further classified each one of them respectively as explained by various authors, explaining Regression Test Selection and Test Case Prioritization in detail with Search Algorithms for Test Case Prioritization. Through this paper author tried to, explain the complete structure of Regression Testing, areas of Regression Testing to make researchers understand its importance and scope and motivate new researchers who are planning to start their research "to work on it".

Mehmet and glu [3] proposed a cost-effective stopping rule using empirical Bayesian principles for a non homogeneous Poisson counting process used with logarithmic-series distribution (LSD) was derived and applied to digital software testing and verification. It was assumed that the software failures or branches covered, whichever the case may be, clustered at the application of a given test-case are positively correlated, i.e., contagious, implying that the occurrence of one software failure (or coverage of a branch) positively influences the occurrence of the next. Singh and Kumar [2] proposed simulation show that the proposed GAs with the specification that find solutions with better quality in shorter time. The developer used this information to search, locate, and set apart from other faults that caused the failures. While each of these areas for future consideration could be further investigated with respect to applicability for software testing, as demonstrated by the examples of this paper, the simple genetic algorithm approach presented in this paper provides in itself a useful contribution to the selection of test cases and a focused examination of test results.

III. REGRESSION TESTING

Test data are generated based on different situations, including:

- Different widget type, such as a text field which allows users to input whatever they want, a combo box which enumerates all the possible input values
- Different data types, such as integers for which boundary value based test data generation technique can be used, strings for which random test data generation technique can be used
- Different data source, such as data defined by database

- Different data attributes, such as independent or dependent

Regression testing is defined as "the process of retesting the modified parts of the software and ensuring that no new errors have been introduced into previously tested code". Figure 2 gives an example of the test cases generated from a test scenario.

There are various [3] regression testing techniques (1) Retest all; (2) Regression Test Selection; (3) Test Case Prioritization; (4) Hybrid Approach.

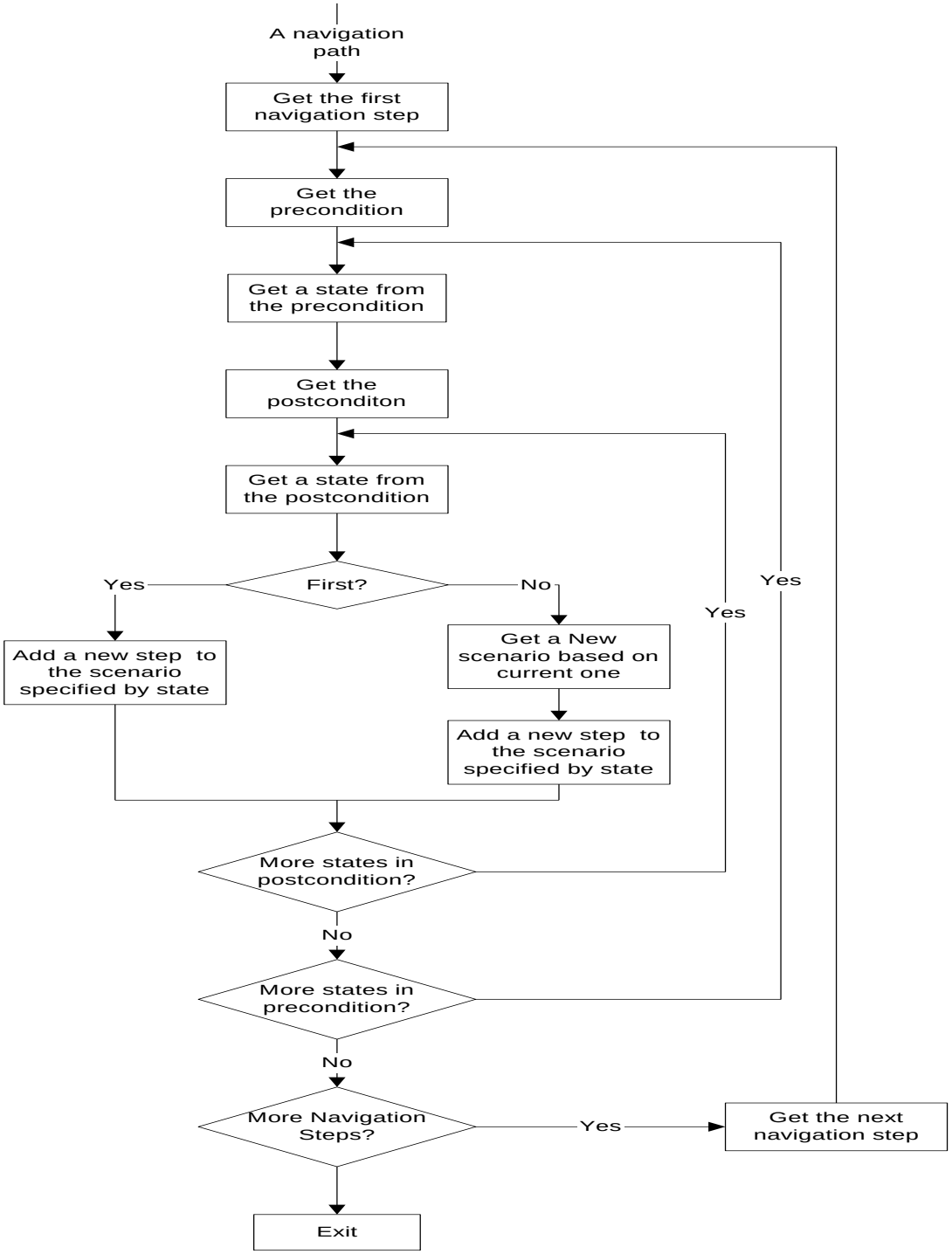

Figure 2 : Test Case Generation Example

A) Retest All:

Retest all method is one of the conventional methods for regression testing in which all the tests in the existing test suite are rerunned. So the 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.

B) 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.

C) 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 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.

D) 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 [3].

We are defining a new approach to assign the priorities to the test cases dynamically while performing the regression testing. The proposed approach is the try to reduce the test cases and assigning a new prioritization sequence. We need to define a database to maintain all the test cases respective to the project. The data will contain different kind of test respective to the criticality level. It will also define the position of the test cases in the data flow over the object. It also define either it is a function test or non function test.

Once all the test cases are defined the next work is assign the priorities to these test cases. The prioritization should be assigned according to the criticality of the test as well as the code on which the test is occurred. It also defines how frequent the test is. After considering an initial test cases sequence is generated.

E) Algorithm

Algorithm

```
1.  {
2.  Input a software program, system or module with N number of Stages
3.  Define the Test Cases Associated with Each Stage
4.  Assign the Cost to Each Test Case based on Program Analysis
5.  Assign the Priority Low to NonChangeable TestCases during the Regression Testing
6.  Assign the High priority to Changebale TestCases During the regression Testing
7.  For i=N-1 to 1
8.  {
9.  K=TestCasesAt(i)
10. For j=1 to K
11. {
12. Cost(I,j)=Cost(I,j) * Priority(Test(j))
13. }
14. }
15. for i=n-1 to 1
16. {
17. Find MinCost for TestLevel(i)+MinCost(i+1,n)
18. TotalCost=TotalCost+MinCost
19. }
20. Return TotalCost
21. }
```

IV. CONCLUSIONS

In this paper, we have explored the regression testing approach with all its stages and components. Paper also included the a new dynamic programming based approach under the fault and cost analysis.

REFERENCES

[1] Williams L., "Testing Overview and Black-Box Testing Techniques", page no.35- 59, 2006.
[2] Pressman, Roger S.,"Software engineering: a practitioner's approach" 5th edition, 2001.
[3] La**st** M., Eyal S., and Kandel A.,"Effective Black-Box Testing with Genetic Algorithms",2005
[4] Singh K., Kumar R., "Optimization of Functional Testing using Genetic Algorithms" ,International Journal of Innovation, Management and Technology, Vol. 1, No. 1, April 2010.
[5] Engström E., "Regression Test Selection and Product Line System Testing", Third International Conference on Software Testing, Verification and Validation, 978-0-7695-3990-4/10 $26.00 © 2010 IEEE
[6] Jin W., Orso A., "Automated Behavioral Regression Testing", Third International Conference on Software Testing, Verification and Validation 978-0-7695-3990-4/10 © 2010 IEEE
[7] Do H., Mirarab S., "The Effects of Time Constraints on Test Case Prioritization: A Series of Controlled Experiments", IEEE Transaction on Software engineering, vol. 36, no. 5, September, 0098-5589/10/ 2010 IEEE
[8] Foo K., Jiang Z., Adams B., "Mining Performance Regression Testing Repositories for Automated Performance Analysis",10th International Conference on Quality Software, 1550-6002/10 © 2010 IEEE

[9] Zhang C., *et al.* "An Improved Regression Test Selection Technique by Clustering Execution Profiles", 10th International Conference on Quality Software 1550-6002/10 © 2010 IEEE

[10] Kumar A., Tiwari S., Mishra K., "Generation of Efficient Test Data using Path Selection Strategy with Elitist GA in Regression Testing", 978-1-4244-5540-9/10 ©2010 IEEE

[11] Li B., *et al.*, "Automatic Test Case Selection and Generation for Regression Testing of Composite Service Based on Extensible BPEL Flow Graph", and 60773105, and partially by National High Technology Research and Development, 26th international conference on software maintenance, 978-1-4244-8628-1/10 ©2010 IEEE

[12] Kumar M., *et al.,"* Requirements based Test Case Prioritization using Genetic Algorithm", IJCST Vol. 1, Issue 2, December 2010.

[13] Rothermel G., Roland H.," Test Case Prioritization: An Empirical Study", International Conference on Software Maintenance, Oxford, UK, September, 1999, IEEE Copyright.

[14] Rus I., *et al.*, "Software Dependability Properties: A Survey of Definitions, Measures and Techniques". Fraunhofer Technical Report 03-110, January 2003.

