Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 3, March 2015, pg.259 – 264

RESEARCH ARTICLE

An Adaptive Fault Analysis Approach for Test Path Optimization

Deepti
Student, M.Tech (CSE), Gurgaon College of Engineering, Gurgaon
Haramveer Yadav
HOD, M.Tech (CSE), Gurgaon College of Engineering, Gurgaon

Abstract- To improve the software reliability, the major requirement is to test it under different aspects in terms of test cases and constraints. A software product is described as the number if relative stages and each stage can be analyzed by number of associated test cases. Testing each test case increases the cost and time of testing process. Because of this, there is the requirement to perform effective test case selection so that each stage will be analyzed effectively. In this present work, an adaptive approach is defined to generate cost effective test sequence under test criticality analysis. The obtained results show that the presented work has reduced the overall cost of testing the software system.

Keywords: Testing, Fault Analysis, Black box Testing. Failure

I. INTRODUCTION

One of the major vectors to describe a software system is in terms of its quality. To ensure this quality, there is the requirement to analyze the software system under different testing techniques. The reliability of the system can be described in terms of its robustness against the failure or fault identification over the software system. More probabilistic the software system is against this kind of attack, more reliable the system will be considered. This fault based analysis is able to identify the software criticality so that reliable action will be taken against this criticality so that the software system effectiveness will be improved. A software system can have different kind of faults in the software system. These fault associated vectors are described in figure 1. Each kind of faults are considered as the metrics that are required to identify during the testing process. Each of these faults are described along with the module. The module criticality also affects the criticality of the software system itself. Blackbox testing is able to identify these all kind of errors, faults and failures over the software system.

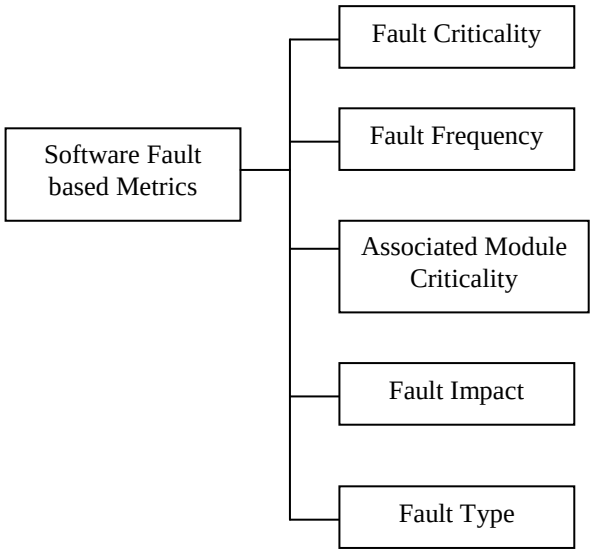

Figure 1 : Fault Criticality Type

Black Box is the testing process defined on the ready software product. The implementation of the software system is here tested based on input values. The software system itself it for different values along with associated constraint. This examination is performed respective to the description of the tested element with block level specification to identify the error cases and the cases with good results. This testing approach is able to identify interface error, process error, performance challenges, module level error identification etc. There are number of associated approaches comes under blackbox testing given here under

- Boundary Value Analysis
- Equivalence Class Partitioning
- Stress Testing
- Interface Testing

The boundary value analysis identifies the limit range analysis on software system for different values. These values can be identified as the valid and invalid data ranges so that the valid process identification and error case identification will be done. This approach itself includes zero value analysis, negative value analysis to generate the possible cases. The equivalence partitioning is defined to drive the test cases under different classes for the equivalence class formation so that the partitioning over the class level for different input domains will be obtained. The design level analysis comes under this testing category so that reliable test cases can be generated over the software system.

The stress testing includes the performance testing in critical situation. This critical situation can be identified in terms of heavy load over the system or in case of restricted resource availability. The resource and the load ratio is the main constraint to analyze such kind of testing over the software module or over the software system. Other kind of testing includes in these software system is interface testing. This testing approach is able to identify relationship between different modules. If the system is connected to some other environment or architecture, the testing will check that for different data values.

In this paper, the black box testing is defined under fault cases and defined an adaptive approach to generate the optimal test sequence. The test case prioritization is here defined using fault based analysis. In this section, the significance of black box testing and fault based analysis is described. In section II, the work defined by earlier researchers is discussed. In section III, the proposed work is explained along with algorithmic approach. In section IV, the results obtained from the work are discussed. In section V, the conclusion obtained from the work is presented.

II. EXISTING WORK

Different researchers included their contribution to perform black box testing using different approaches. Some of their achievements are discussed in this section. 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**.** Tang *et al.* [2] proposed first trial to discuss the effects of test suite reduction under selective-form regression testing. Here the total set of test requirements were divided into two subsets: one was the set of concerned test requirements which ought to be covered by current regression test cases; the other was the set of irrelevant test requirements which avoided to save the test effort, ease the failure analysis, enable missing parts and stubs, or avoid failure protecting codes. Author have been defined the multi-objective test suite reduction problem to model the selective coverage test suite reduction, and put forward the HATS algorithm to solve the problem. HATS have been applied three strategies to select the best-for-now test case(s) which proposed the maximum fitness value, or is the only one left to cover certain concerned test requirement(s). Experimental results has been compared with the total-coverage test suite reduction techniques, with proper settings of α, the HATS have been largely decrease the size of the regression test bucket. Tao *et al.* [3] proposed how to apply hierarchical slicing technique to regression test selection. Experimental studies indicate that the approach provide selected test cases step by step, and can improve the precision of regression test selection. From high level to low level of program, this approach was a stepwise procedure of regression test selection.

For a large-scale object-oriented program, hierarchical slicing was used to select regression test cases step by step. Chen *et al.* [4] proposed a dependence analysis based test case prioritization technique for Web Service regression testing. First, they analyzed the dependence relationship using control and data flow information in an orchestration language: WS-BPEL. Then they construct a weighted graph. After that, they prioritize test cases according to covering more modification-affected elements with the highest weight. Finally authors conduct a case study to illustrate the applicability of method**.** A weighted dependence propagation model was proposed to facilitate prioritization process.

Yang and Wu [5] proposed a regression testing method for Composite Web service. The method can instruct the tester in locating fault in Composite Web Service, and make test data and test behavior independent of each other. Author developed a prototype, and put a system under testing into practical use, the result indicates the method is effective to Composite Web. Engström [6] 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 [7] 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' behaviour. Third, it analyzes the identified differences and presents them to the developers. Do, Mirarab [8] 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.* [10] 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 behaviours can be well understood and test cases can be selected in a proper way to reduce the test suite effectively. Hsu [11] presented many data mining applications that use optimization and classification. The current literature contains several general observations about the generation of solutions using a genetic algorithm: GAs is sensitive to deceptivity, the irregularity of the fitness landscape. This includes locally optimal solutions that are not globally optimal; lack of fitness gradient for a given step size; and jump discontinuities in fitness.

III. RESEARCH METHODOLOGY

One of the major aspect to analyze a software system under the testing process is the path testing. The path testing can be defined either a white box or the black box testing approach. In this presented work, we have defined the fault analysis based on the path testing. The presented work is divided in two main phases. In first phase, each code block will be defined respective to the different test cases and based on these test case analysis some priorities will be assigned to these test cases. The main focus of presented work is about to analyze the prioritization process. In this work fault based analysis approach is defined for prioritization. Once the prioritization will be assigned, the second phase will define the optimal sequence of the implementation of these test cases. For this sequence generation the genetic approach is analyzed based on the associated vector. Where the prioritization is taken as the main criteria to decide the next generation sequence. The presented work is the analytical work that will be done under some examples and based on the analysis the results will be driven.

In this present work a software module criticality is analyzed on the associated fault. The fault vector can be defined to generate the criticality so that more adaptive analysis over the software system will be performed. The work is here defined to analyze the modules based on the associated test vector and each test vector is able to identify different kind of fault over the system. The criticality of the fault itself represent the priority to the associated test case. These associated faults are defined here under different vectors shown in table 1.

Table 1 : Software Fault Association

Fault Type	Priority
Warning	Low
Error	Medium
Fault	Medium
Failure	High
Independent	Low
Dependent	High
Application Based	Low
System Based	High

The work is defined in a layered model. At the initial stage, the software system is defined in terms of smaller modules. Once the modules are described, the identification of the test cases associated with particular module is done. A software module can be associated with one or more test case. Once the test cases are identified, the next work is to perform the test cost analysis using different approaches. One of such approach is to perform the estimation base of fault analysis. The software fault analysis is defined as the classification of the test case criticality as well as relative module criticality. A test case is defined with higher priority if it is representing some fatal error or the fault. If the test is defining some normal fault then the priority will be medium. If the test case is resolving some warning then the priority will be lowest. Once the priorities are assigned to these test cases, the cost estimation is also done based on priority analysis.

The presented work is about to derive the optimal sequence of the test case generation under different prioritization approaches. The presented work will provide a cost effective sequence of the test case generation and the analysis. The work is also effective to analyze the software under different point of views so that the effectiveness of the code and test cases can be obtained.The path testing is a very beneficial white box testing that play an important role to select the actual code test sequence as well as to perform the code optimization. The code optimization is actually performed in terms of detection of unusable code from the test sequence. There are number of aspects according to which the code sequence is decided. Better the test sequence is estimated; better the testing will be performed. It will also affect the testing code and test case reusability. The proposed work is the implementation of approach that will work the basis of different priotization vectors and to generate the optimal sequence of test case generation. The flow of work is shown in figure 2.

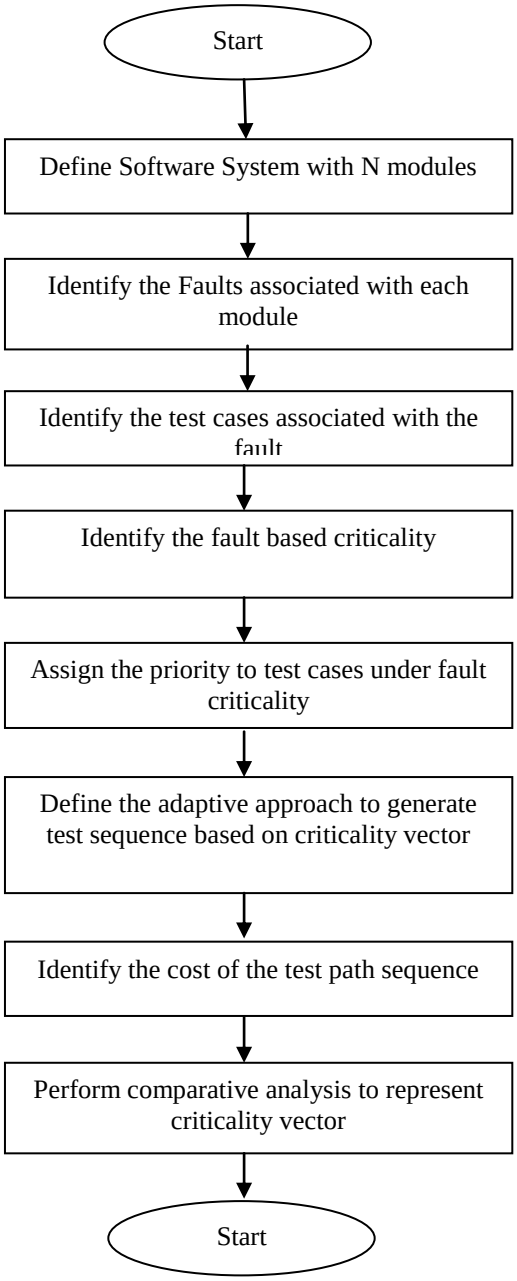

Figure 2 : Flow of Work

IV. RESULTS

The presented work is been implemented in Matlab 7.8 Environment. The work is here defined with dummy module set with assignment of the priority under fault vector. The proposed approach is here been implemented to obtain the fault sequence and to generate the optimal sequence results. The obtained results are here analyzed under cost vector.

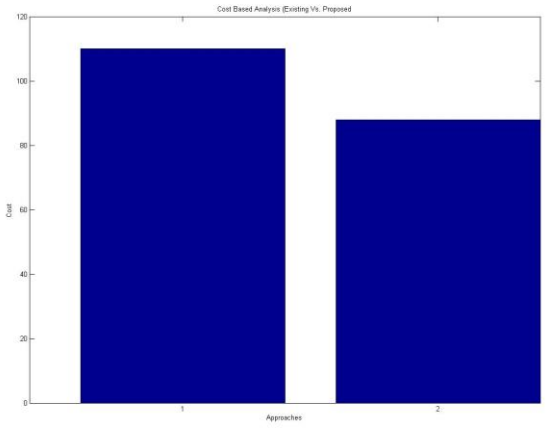

Figure 3 : Cost Based Analysis

Here figure 3 is showing the cost based analysis of prsented work. The obtained results shows that the presneted work has reduced the cost of optimal sequence generation under fault based analysis.

V. CONCLUSION

In this present work, an fault adaptive approach is defined to generate the optimal test sequence under black box testing approach. The work is here based on the prioritization assigned to the test cases under fault based analysis. The obtained results shows the effective sequence generation for test sequence optimization.

References

[1] Last M., Eyal S., and Kandel A.,"Effective Black-Box Testing with Genetic Algorithms",2005

[2] Gu1 Q., Tang B.," Optimal Regression Testing based on Selective Coverage of Test Requirements", International Symposium on Parallel and Distributed Processing with Applications,978-1-7695-4152-7/10 $26.00 © 2010 IEEE

[3] Tao C., et al.," An Approach to Regression Test Selection Based on Hierarchical Slicing Technique", 34th Annual IEEE Computer Software and Applications Conference Workshops, 978-0-7695-4105-1/10 $26.00 © 2010 IEEE.

[4] Gu Q., Chen Q.," Optimal Regression Testing based on Selective Coverage of Test Requirements", International Symposium on Parallel and Distributed Processing with Applications, 978-0-7695-4190-7/10 $26.00 © 2010 IEEE

[5] Bo Yang, Ji Wu, Chao Liu, Luo Xu [2010], "A Regression Testing Method for Composite Web Service".

[6] 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

[7] 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

[8] 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

[9] King Chun Foo, Zhen Ming Jiang, Bram Adams [2010], "Mining Performance Regression Testing Repositories for Automated Performance Analysis".

[10] 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

[11] Hsu W., "Genetic Algorithms", Kansas State University, USA.

