

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320-088X

IJCSMC, Vol. 3, Issue. 7, July 2014, pg.161 – 167

RESEARCH ARTICLE



A Dynamic Programming Based Approach for Test Sequence Generation

Richa Moudgil¹, Swati²

¹Student, M.Tech (CSE), Meri College of Engineering and Technology, Sampla, Haryana
Richamoudgil2010@gmail.com

²Asstt. Prof., M.Tech (CSE), Meri College of Engineering and Technology, Sampla, Haryana
swati_thakral@yahoo.co.in

Abstract: The software reliability is based on effective software testing. To perform the optimized testing for a software system, a cost effective test sequence is required. In this work, a two stage model is presented for test sequence generation. In first stage of this model, the prioritization is applied to test cases under different vectors. The vectors included in this work are fault based analysis and module interaction analysis. Once the prioritization is applied, the next work is to apply the dynamic programming approach to obtain the optimal test sequence. The work is analyzed under different prioritization cases, and the analysis results show the effect of different prioritization methods for test sequence generation.

Keywords: Prioritization, Dynamic Programming, Cost Analysis, Module Interaction

I. INTRODUCTION

When the test plan is prepared for a software system, it is required to estimate the cost of testing so that the effective cost estimation will be performed over it. The effective test sequence reduces the overall test cost. The software cost estimation is one of the most required phenomenon associated with software system to analyze the software system cost along with the estimation of optimal test sequence. In this work, an effective approach is defined for test sequence generation and cost estimation. To generate the optimal test sequence it is required to categorize these test cases based on the criticality level. There are number of different vectors based on which the test case criticality can be identified. In this work, an effective approach is defined for test case prioritization[1][2][3]. The basic flow of the test sequence generation is shown in figure 1.

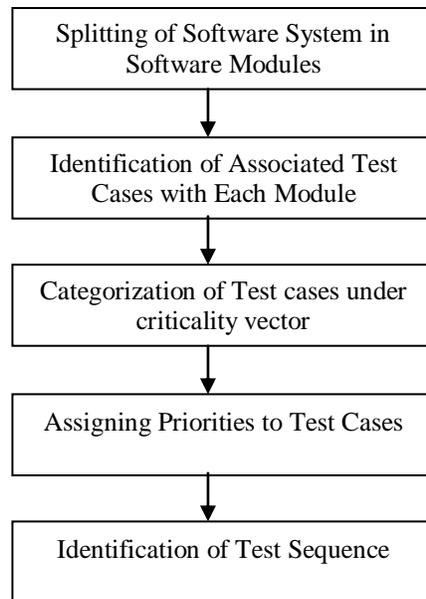


Figure 1 : Flow of Test Sequence Generation

As shown in the figure, the primary step for the software test plan is to represent the complete software system in terms of modules or the procedure. On these modules the unit testing is performed. As the individual modules are analyzed, the integration testing is performed while combining these modules. Finally, when all these modules get integrated, the system testing is applied over it for testing the complete software system. All these test stages are associated with software module. Once the modules are identified, the next work is to identify the associated test cases. The associated test cases are here defined in terms of different errors, exceptions and warning generated from the module or the system[4][5][6].

Based on the study of these modules or the test cases, the criticality of the test case is defined. There are number of approaches for identification of test case criticality. Some of these approaches are shown in figure 2.

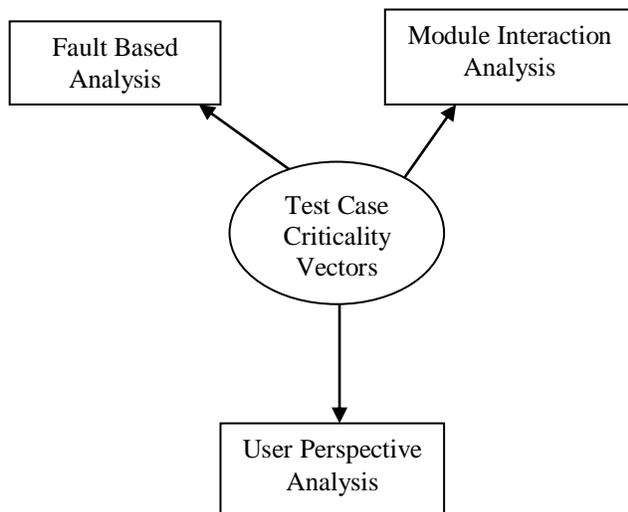


Figure 2 : Test Case Criticality Vectors

Here figure 2 is showing the test case criticality vectors. These vectors include the Fault Based Analysis, Module Interaction Analysis and User Perspective Analysis. These vectors are used to decide or identify the test case criticality.

After identifying the criticality vector for software system and the relative criticality value, the next work is to assign the priority to the test cases based on which the test sequence identification process is carried on. The final stage of work is to generate the effective test sequence based on which the software test cases will be applied over the software project. The optimality is here identified under the cost vector.

In this section, the basic model of the test sequence generation is defined. The model is defined in terms of sub stages. In section II, the work defined by the earlier researchers for test sequence generation is defined. In section III, the proposed work model is presented. In section IV, the results obtained from the work are presented and discussed. In section V, the conclusion obtained from the work is presented.

II. EXISTING WORK

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.

Kaur, Goyal [9] proposed algorithm to prioritize test cases using Genetic Algorithm. Here, different prioritization approach had been analyzed, namely: total fault coverage with in time constrained environment and amount of code coverage on different examples and their finite solution obtained. The elaborations of results are shown with the help of APCC values. The APCC had been calculated for example for code coverage testing to evaluate the usefulness of the proposed algorithm. The algorithm is solved manually and is a step towards Test Automation. Wong *et al.* [10] proposed a modification-based technique followed by test set minimization or prioritization to determine which regression tests should be rerun. Stated differently, only regression tests in (I) T 00 MB obtained by modification + minimization, or (II) the top N from T 00 P obtained by modification +prioritization are selected. Three metrics, size reduction, precision, and recall, are used to examine the goodness of using (I) and (II). These metric values depend not only on the nature of the regression test suite but also the extent and the locations of the modifications. Unlike many other proposed test selection techniques, ours is supported by a tool called ATAC. As explained in Section 2.3, we have not experienced any excessive computing time in doing test set minimization. If the exact solution is not obtained in reasonable time, ATAC provides an approximate minimized set using "greedy" heuristics. Kumar *et al.* [11] proposed a combined approach by which the stated problems are resolved in effective manner. By this approach, tester identified the appropriate paths for test case execution. The proposed approach enable tester to execute the test cases in order to increase their effectiveness to find faults taking minimum efforts. This approach was used in regression testing to choose an appropriate subset of test cases by using elitist GA, among a previously run test suite for a software system, based on the information about the

modifications made to the system for enhancement. Li, *et al.* [12] proposed the changes by using control-flow analysis technique and comparing the paths in new composite service version and the old one using extensible BPEL flow graph (or XBFG). Message flow is appended to the control flow so that XBFG can describe the behavior of composite service integrally. The binding and predicate constraint information added in XBFG elements was used in path selection and test case generation. Theory analysis and case study both show that the approach was effective, and test cases coverage rate is high for the changes of processes, bindings and interfaces. Reddy, Kumar [13] proposed a new technique by which we can reduce the regression test phase by using a variety of inputs to determine the appropriate smaller number of test cases out of entire large test suite. In this paper they developed a new regression optimization technique, which reduces the regression cycle time and improves the quality of testing. Albertins *et al.* [15] described related works on test case prioritization and illustrates the variety of criteria applied to test sequence generation.

III. RESEARCH METHODOLOGY

The presented work is defined as an effective model for test sequence generation using dynamic programming approach the presented work is divided in three main stages defined in figure 3

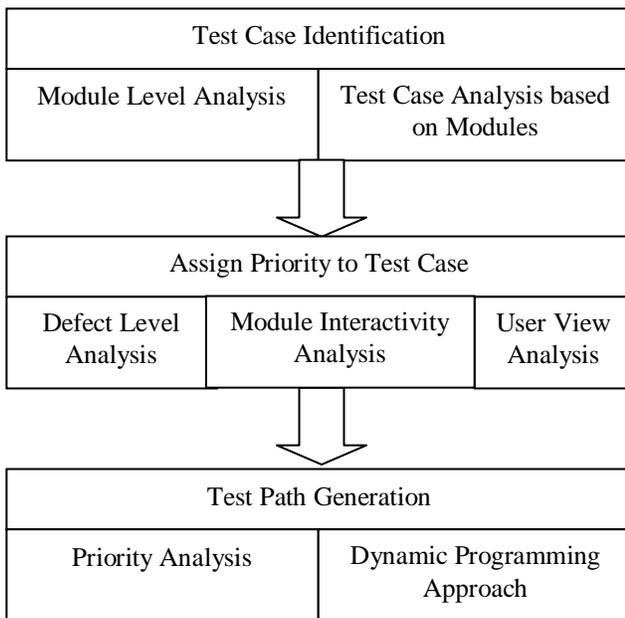


Figure 3 : Proposed Model

In this work, 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.

It is required to define the event that can affect the available code or the related test cases. With each event we define the affected test cases. The test cases affection is represented as use case. It also defines as the event the particular test

cases will be required to perform or not. If it is required it will check wither it will be uses in same or some modification is required. After the use case is assigned to the available test cases the next work is to assign a new sequence of test case implementation. This work will be performed dynamically by keeping the existing test cases in mind as well as by observing the criticality level as well the use cases of the particular test case.

Table 1 : Algorithm

```

Algorithm
1. {
2. Divide the Software Project in the term of software Module
   and define The relative Test Cases with Each Software
   Module
3. For i=1 to Length(TestCases)
4. {
5. TestCasetAtModule(i)=SetAssociatedTestCases
6. }
7. Identify the Individual Test Cases for the Module
8. For i=1 to Length(Modules)
9. {
10. For j=1 to Module(i).TestCount
11. {
12. TestCase(I,j).Priority)=AssignPriority(High,Medium,Low)
13. }
14. For i=1 to TestCase.Length
15. {
16. Cost(i)=IdentifyTestCost * TestCaset(i).Priority;
17. }
18. for i=Length(Module)-1 to 1
19. {
20. Identify the Minimum Cost Test Case For Module(i)
21. Cost=Cost+MinCost
22. }
23. Return Cost
24. }
    
```

IV. RESULTS

The presented work is implemented in matlab environment for test sequence generation under dynamic programming and prioritization approach. The work is tested for different prioritization sequence and based on it the test sequence cost estimation is performed. As the general case we have assigned the random cost to each test case and perform the analysis based on this random cost assignment. The output driven based on this assignment is shown as under.

a) The obtained Test Sequence of this random cost assignment is given as

3 9 4 1 10 7 8 5 6 2

b) The cost driven from the on given approach is given as

Cost = 11.0547

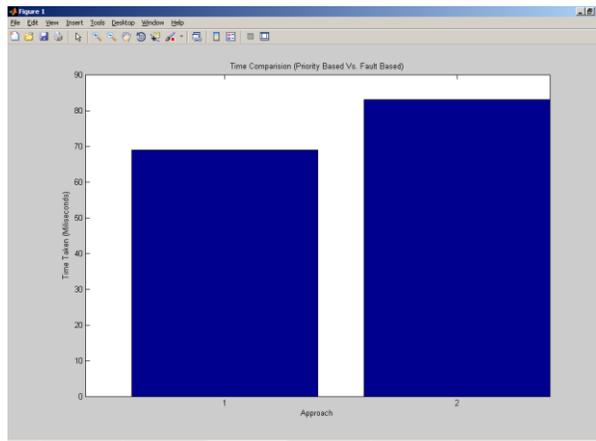


Figure 4 : Test Cost Estimation

Here figure is showing the time based comparative analysis between the priority based test path analysis and the cost based test path analysis. Here x axis represent these two approaches and y axis represents the time taken by the approach. The time is given in milliseconds. We can see that the fault based analysis is less efficient then priority based analysis

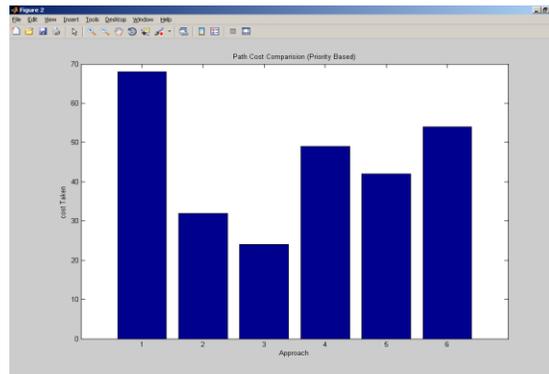


Figure 5 : Cost Analysis (Different Prioritization Approach)

Here figure 5 is showing the comparative analysis of different cost testing path cost in case of priority based test path estimation. Here we have considered the priority between 1 and 3. case 1 represents when all priority are 1, case 2 represents when all priorities are 2 an case 3 represents when all priorities are 3. case 4 represents the priorities assigned in ascending order and case 5 represents the priority assignment in descending order. Case 6 represents the random assignment of the priorities. Here y axis represents the cost of test path.

V. CONCLUSION

In this present paper, an effective test sequence generation approach is defined under dynamic programming approach. The work is presented as two stage model to assign the priorities to test cases and to generate the optimal test sequence. The obtained results show the cost effective sequence generation.

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] Last 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] Kaur A., Goyal S.," A genetic algorithm for regression test case prioritize using code coverage", International Journal on Computer Science and Engineering (IJCSSE), ISSN: 0975-3397 Vol. 3 No. 5 May 2011.
- [10] Wong W., Agrawal. H.," A Study of Effective Regression Testing in Practice", 8th IEEE International Symposium on Software Reliability Engineering (ISSRE'97), pp 264-274, Albuquerque, NM, November 1997.
- [11] Kumar A., Tiwari S., "Generation of Efficient Test Data using Path Selection Strategy with Elitist GA in Regression Testing", 978-1-4244-5540-9/10/\$26.00 ©2010 IEEE
- [12] Li B., Qiu D., "Automatic Test Case Selection and Generation for Regression Testing of Composite Service Based on Extensible BPEL Flow Graph", 26th IEEE International Conference on Software Maintenance 978-1-4244-8675-1/10\$26.00 © 2010 IEEE
- [13] Reddy S., Kumar S.,"An effective approach to regression test optimization technique", Indian Journal of Computer Science and Engineering (IJCSSE), ISSN: 0976-5166 Vol. 2 No. 5 Oct-Nov 2011.
- [14] Albertins L., Sampaio A., "A Permutation Technique for Test Case Prioritization in a Black-box Environment", 2nd Brazilian Workshop on Systematic and Automated Software Testing, page no. 1-10, 2008