A Fault Processed Genetic Approach for Test Path Optimization

Abhinesh[1], Yogesh[2]

[1]Student, UIET, M.D.U., Rohtak, Haryana, India
[2]A.P., UIET, M.D.U., Rohtak, Haryana, India

ABSTRACT

Software testing is the primary task to control the software cost and quality. With each process stage of SDLC, a testing phase is included to it. Before applying the testing, it is required to setup the sequence of execution of test cases. The test sequence must be cost adaptive so that the testing time and cost will be reduced. In this work, a priority adaptive genetic approach is defined to generate the optimized test sequence. The model evaluated the test case weights based on fault coverage and the module interactivity. Once the cost of individual test case is identified, the genetic model is applied to identify the optimized test sequence. The defined model is implemented in matlab environment. The implementation results show that the model has generated the cost effective test sequence.

Keywords: Path Testing, Genetics, Cost Optimized, Priority Driven, Fault Based.

1. INTRODUCTION

Almost everything that is in use has a component of software in it. Today, in a typical workplace (e.g. Home), everyone uses a computer and software. Software today is as common as electricity was in the early part of the last century. Almost every gadget and device we have at home and at work is embedded with software. Mobile phones, televisions, wrist watches etc. all have embedded software. Now, software is being used in some missions where failure is not acceptable. There are number of associated application area including the Banks, insurance, Cars etc. These are the real time system application that requires adapting the reliability so that the continuous application rule will be obtained without any fault. To improve the system reliability software needed to be tested before it is delivered. There are some critical applications of software testing that are hardly time bounded such as nuclear reactor monitoring, flight control etc. These kinds of systems collect the critical aspects of software testing under the development model. The software testing enables the systematic and rule based development process under the software plan boundaries so that development will be performed.

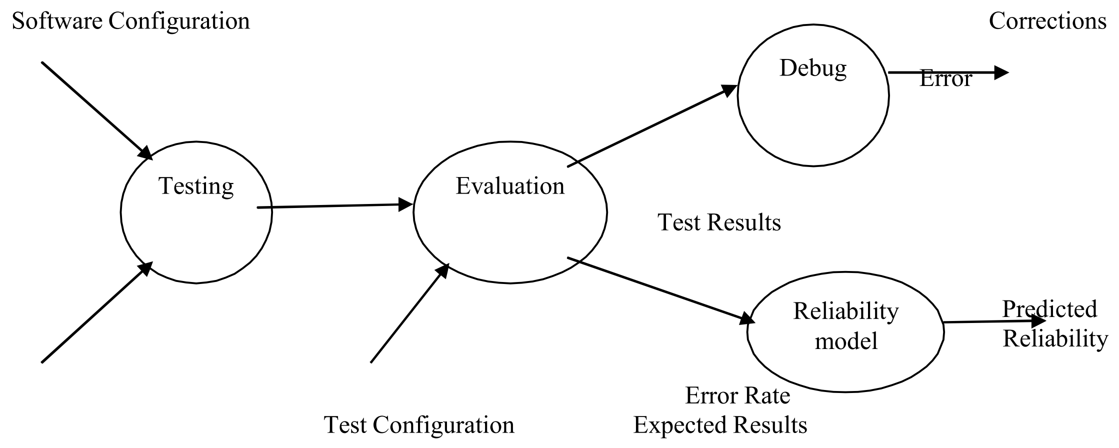

Figure 1: Test Information Flow

Software testing results come under the evaluation phase where the delivered software project or the developed software product or service is compared with planned software product or service. In this phase, the analysis of software system is done under the error analysis and bug rate analysis. To identify the bugs, different levels of testing is performed by different groups. These groups include the developers, managers, testers and the analyzers work and effort to identify the software error. The bugs are identified at module level, system level, integration level and environment level. Once the bugs are identified, the fix operations are directed over it. To identify the software reliability, these bugs are identified in terms of rate of bug occurrence. Such model is called software reliability model where the reliability is predicted.

A) Black Box or Functional Testing

Black box testing is the testing of a piece of software without regard to its underlying implementation. Specifically, it dictates that test cases for a piece of software are to be generated based solely on an examination of the specification (external description) for that piece of software. The goal of black box testing is to demonstrate that the software being tested does not adhere to its external specification.

B) White Box Testing

White box testing also known as glass box testing .It is a test case design method that uses the control structure of the procedural design to derive test cases [7]. Using white box testing methods, the software engineer can derive test cases that

1. Guarantee that all independent paths within a module have been exercised at least once.
2. Exercise all logical decisions on their true and false sides.
3. Execute all loops at their boundaries and within their operational bounds.
4. Exercise internal data structures to assure their validity.

2. RELATED WORK

Author [1] has defined a work on the analysis of software system for different structural and object oriented metrics. This metric based component estimation along with relationship analysis is performed under the procedure based analysis. Author discussed the metrics such as LOC, cyclomatic complexity, cohesion and coupling metrics. Author[2] has defined estimation on software products to analyze the software system under defect analysis for object oriented software system. Author[3] has presented resource based software estimation scheme for software quality analysis. Author defined budget analysis approach to improve the software product analysis. Author performed analysis under different testing aspects. Author[4] has defined an improved metrics based complexity model for object oriented programming. Author defined the complexity analysis under multiple aspects so that effective software development under method analysis will be performed. A computational system for the software system under the software development rules was defined by the author under cost, timeline and quality analysis[5].

Author[6] defined the software process and software design mechanism to analyze the software system under software evolution and software management analysis. Author defined the specific development model so that the development process will be improved. Author[7] defined the software measurement analysis under the defined framework so that the software measurement validation is performed. Author defined the structural model for software development so that the attribute relation analysis will be performed. Author[7] has presented an effective approach for software development using object oriented programming. Author defined the development model for C++ programs. Author[8] presented an object oriented model to analyze the software system under prediction and maintenance analysis so that the effort based software modeling will be done. Author presented the solution analysis under the bounds of estimation and perdition systems. Author[9] denied the regression analysis under multi linear system for real data representation and management. Author defined the effort and error rate analysis so that the effective development will be performed. Author[10] defined the software management model under the current and identical requirements of the system. Author has defined an advantage and drawback analysis system for object oriented programming and metrics. Author analyzes the system under fault proneness so that the low level system design will be configured.

Author[11] has defined an effective complexity analysis model for software development. Author defined the complexity analysis and measurement so that the effective development model will be presented. Author[12] has presented a work on random testing to generate the optimized path so that the effective software testing will be obtained. The path generation is here done under execution modeling and constraints analysis. Author[13] has presented a work on path selection approach causing graph partition so that the communication delay will be reduced and testing impact will be improved. Author[14] has presented a work on path optimization under regression testing so that the static path generation with coverage testing will be obtained. Author[15] has presented a work on defect analysis based delay estimation approach for path correlation analysis.

3. RESEARCH METHODOLOGY

The presented work is about to generate the effective test cases and the test data so that the effective test case processing will be done. The presented work will be defined in three main stages. In first stage, the software project will be divided in smaller zones based on graph level analysis. This division will be based on number of modules and connectivity covered in each zone. In second phase, the fuzzy integrated criticality analysis will be applied for test cases and the module to generate a weighted priority assignment. In the final stage, the fuzzy-genetic method will be applied to generate the optimized test sequence. The proposed fuzzy-genetic adaptive work model is shown in figure 2.

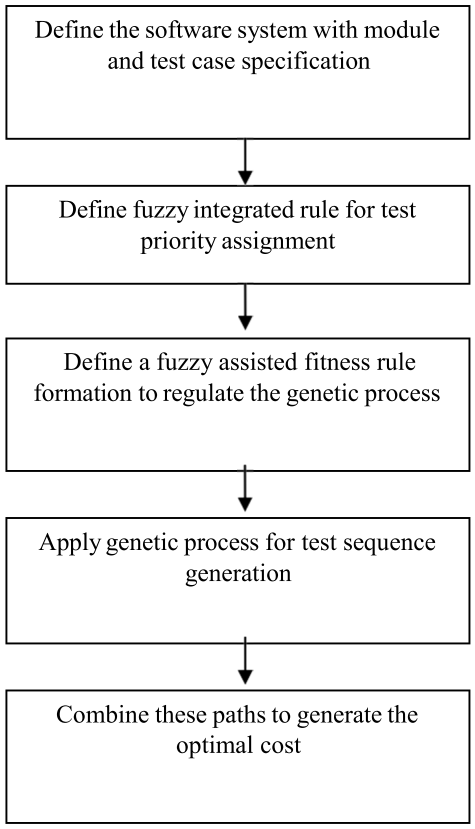
Define the software system with module and test case specification
Define fuzzy integrated rule for test priority assignment
Define a fuzzy assisted fitness rule formation to regulate the genetic process
Apply genetic process for test sequence generation
Combine these paths to generate the optimal cost

Figure 2: Proposed Model

Here figure shows that the model will first represent the software system in small modules. For each module, the test cases will be defined. The test case prioritization will be applied in each case using fuzzy based approach. The fuzzy integration is here done as the fitness rule. Based on this evaluation, the genetic model is applied to minimize the cost of test sequence generation. The method divided the project into smaller zones and applied the genetic to them in combined form. The fuzzy rule is defined to perform the cost evaluation and test case prioritization.

In this section, the algorithmic model to perform the communication over the trust nodes is defined. The presented work is simulated on a random network. The comparative results are shown in next section.

4. RESULTS

We have analysed this test cases based on the cost of the test case. The cost depends on the occurrence and the detection of the software fault in a particular module. Here we have assigned the test cost in different ways to perform the analysis. The implementation of work is done in matlab environment. Here the analysis is given on the basis of the Test Cost shown in figure 3. Here figure is showing the estimation of test sequence cost for different set of test cost assignment. The genetic configuration is defined with 200 iterations specification. The cost assignment is done using four different methods called random cost assignment, cost assignment in increasing order, decreasing order and in defined range.

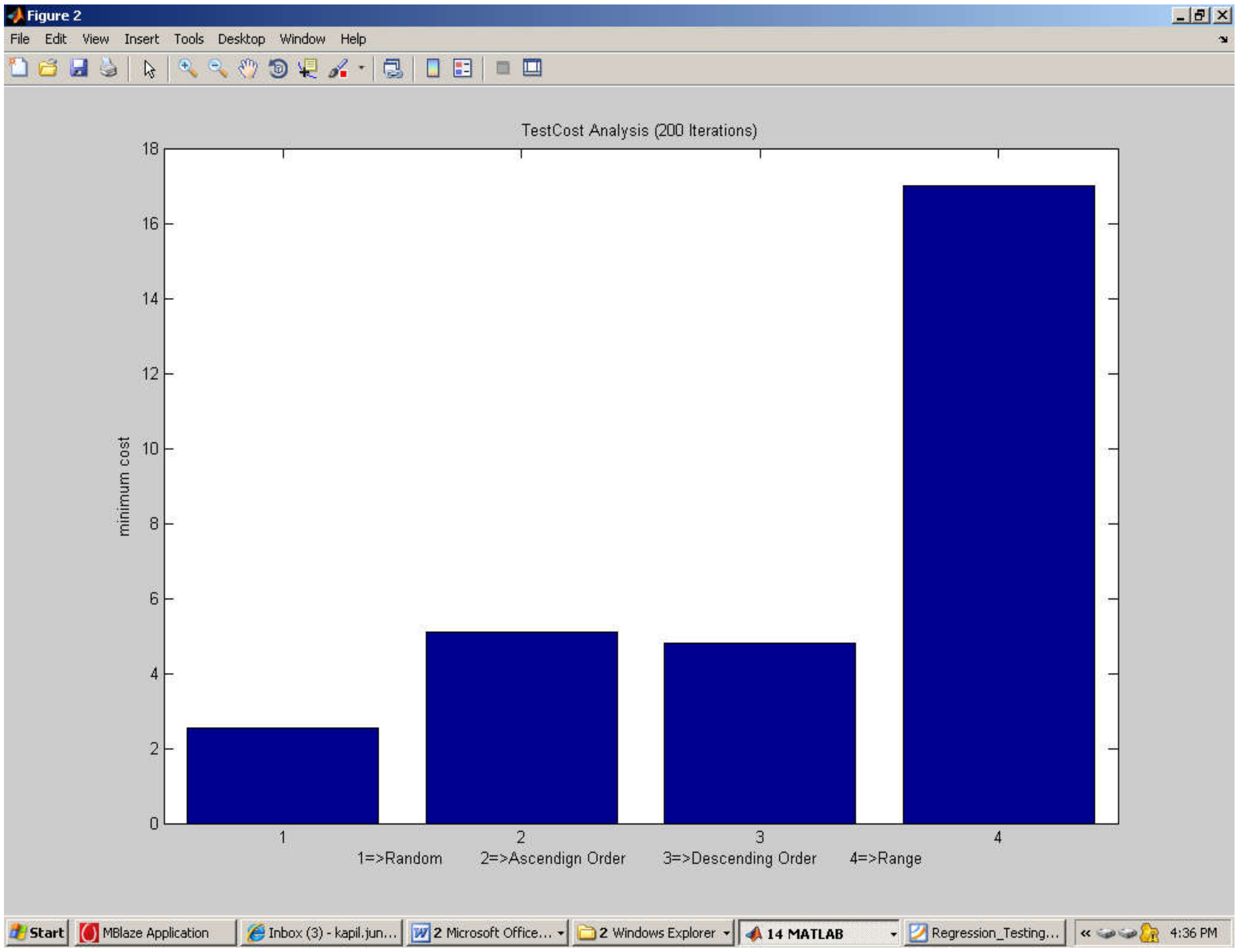

Figure 3: Test Cost Analysis (All Methods/200 Generations)

Here figure 3 is showing the optimized cost results obtained using fuzzy-genetic based approach. The cost assignment of test cases is done in four different methods and based on which the priority is applied. The figure shows that the cost of random cost optimization is least whereas when a range specific cost is assigned, it takes higher cost on aggregate. The method is here implemented for about 200 iterations.

5. CONCLUSION

To improve the software reliability it is required to analyze the software system under different aspects. This analysis can be done under the specification of application behavior and process flow. Software testing defines the process to analyze the software system and improve the reliability. In this present work a zone adaptive genetic programming model is defined for route optimization. The work is here defined to generate a priority based cost adaptive route so that the overall test sequence will be optimized.

REFERENCES

[1] Sommerville,"Software Engineering", 7th edition, pp 1-31, 2004.
[2] "Software Engineering 2004", a Volume of the Computing Curricula Series, The Joint Task Force on Computing Curricula IEEE Computing Society and Association for Computing Machinery, 2004.
[3] DeMarco, Tom, "Controlling Software Projects", Yourdon Press, New York, 1986.
[4] R. S. Pressman,"Software Engineering, A Practitioner's Approach", Sixth Edition, Mc Graw. Hill, 2005.
[5] Gurdev Singh, Dilbag Singh, and Vikram Singh," A Study of Software metrics", IJCEM International Journal of Computational Engineering & Management, Vol. 11, January 2011.
[6] Linda L. Westfall Principle Software Measurement Services Plano, TX 75075," Seven Steps to Designing a Software Metric.
[7] Albrecht, A. J. and J. E. Gaffney. Jr. "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", IEEE Trans. Software Eng. SE-9, 6, pp. 639-648, Nov. 1983.

[8] T. J. McCabe, “A Complexity Measure,” ICSE '76: Proceedings of the 2nd international conference on Software engineering, 1976.
[9] Lou Marco, "Measuring software complexity", Enterprise Systems Journal, 1997.
[10] Kafura, D. and G. R. Reddy. “The Use of Software Complexity Metrics in Software Maintenance”, IEEE Trans. Software Eng. SE-13.3 (March 1987), pp. 335-343.
[11] Everald E. Mills, "Software metrics", Software Engineering Institute, 1988.
[12] Chhikara, Chhillar, and Khatri,” Evaluating the Impact of Different Types of Inheritance on the Component Driven Software metrics”, International Journal of Enterprise Computing and Business SystemsISSN: 223-8849 Volume 1 Issue 2 July 2011.
[13] Barbara A. Kitchenham, Robert T. Hughes, and Stephen G. Linkman,” Modeling Software Measurement Data”, IEEE Transactions on Software Engineering.
[14] Gene F. Walters, James A. McCall,” Software Quality Metrics for Life-Cycle Cost Reduction”, IEEE Transactions on Reliability, vol. r-28, no. 3, August 1979.
[15] June Verner and Graham Tate,” A Software Size Model”, IEEE Transactions onSoftware Engineering, vol. 18, no. 4, April 1992.