Improvement in Genetic Algorithm to Increase Error Detection Rate for Product Line Model based Testing

[1]Gurpreet Kaur

[1]Student Department of CSE, Lovely Professional University, Punjab, India

Abstract: Testing is a used to evaluate the system to discover that it satisfied with given requirements or it does not. Testing used to execute the system to find out any error, or missing requirements. Model based testing is a black box testing in which models are used to generate test cases. The online technique of Model based testing is used to generate test cases automatically. Sometimes faults are occurred in the test cases. In this paper, enhancement in genetic algorithm is done with supervised learning to remove faults from the test cases.

Keywords: Testing, test cases, Model based testing, Genetic algorithm, Results and comparison.

1. Introduction

Software Engineering [1] is the production of the software from the system specification. Software Engineering is an engineering application to design, development and maintenance of the software. It is a methodical application of technical knowledge, experience and methods to the design, implementation, testing and documentation of software or we can say engineering of the software in terms of design, implementation, testing and maintenance of the software.

Software testing [2][3] provides an independent view of system software which allows business for understanding risks in implementation of software. It is not a process to execute an application for finding errors but testing is also used to find failures in software so that errors are find out and corrected. Software testing itself is related to two processes called validation and verification. Verification is "the process of assessing a system to know that the software of a given phase of development satisfy the conditions forcing start phase" and Validation is "the process of assessing a system during or at the last of the process of development to determine that system is satisfied with the given needs or not".

Section 1.1explain Model Based Testing, 1.2 explain mutation testing, 1.3 explain software product line, 1.4 give details about the process of genetic algorithm. Section 2 give detail of related work, section 3 explain proposed methodology, section 4 contain experimental results and comparison and section 5 include conclusion and future scope.

1.1 Model Based Testing

Test suites are not obtained from source code but it is obtained from models. MBT is the form of black-box testing. Models used to representing desire behavior of System Under Test (SUT) and also represent test environment. MBT (Model based testing) is mainly used to generate test cases automatically [4]. Fig. 1.1 shows general Model Based Testing setting.

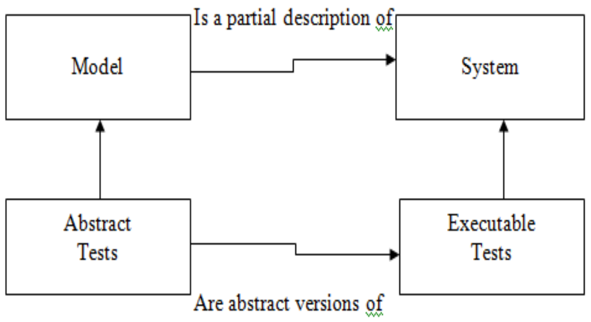

Fig. 1.1 Model Based Testing setting

Test case is obtained from model that test case is functional tests on the same level of hiding as model. An abstract test suite is on the incorrect level of hiding so it cannot be directly executed against an SUT. Executable test suites is necessary to obtained by abstract test suites. Executable test suites communicate with the System Under Test directly. This is done by mapping abstract test case to test cases which is concrete which suits for execution. Inside some MBT environments, some model can contain sufficient knowledge for directly generating executable test suites.

Advantages:-

- You have not necessary for writing new tests for every new aspect. When you have a model it becomes easy to re-generate the test cases than the manual test cases.
- Whenever add a new feature, one new action is added to state machine for running in combination with old actions. A small change automatically wave by whole suite of test cases. So, the designing is fluid.
- Design much and less coding.
- Tests continuously used to finding bugs.

1.1.1 MBT Techniques

Model Based Testing performed by two techniques. [5] a) Online; Model Based Testing tools connect directly to System Under Test(SUT) and dynamically test it. During execution test cases are generated dynamically. As test cases are generated it executes the tests and support long test runs. b) Offline; Test cases are generated from test models for later execution on SUT. It can be done by two ways: offline generation of executable tests which generate test cases as computer readable assets that can run automatically, and offline generation of manually deployable tests which generate test cases as human readable assets that can later used for manual testing. Test cases are generated automatically.

1.2 Mutation Testing

Mutation testing is used to detect the number of mutants in the model. The number of mutants found is known as the number of kill mutants otherwise live mutants. It calculates the mutation score. Mutation score is the ratio of number of kill mutants to the total number of mutants.[6]

$$\text{Mutation score} = \frac{\text{Number of kill Mutants}}{\text{Total Number of mutants}}$$

1.3 Software Product line

Software Product Line [7] [8] shares the common set of features to satisfy the market segment. Software produc line testing tests these common features. The products which come from the same chain but dressed differently and they share methods, processes are known as software product line. According to Barry Boehm's it is code reuse. Software product line provides practical guidance for business case. It produces the new product from the same development by adding or enhancing little. It makes the relationship between the code, architecture and production process. Parts of the software can be reused across the product line.

1.4 Genetic Algorithm

Genetic algorithm [9][10][11] developed by Goldberg was inspired by Darwin's theory of evolution. Genetic algorithm generates the solution for a problem. The GA is a stochastic global search method that mimics the metaphor of natural biological evolution. Genetic algorithm operates on a population of potential solutions applying the principle of survival of the fittest to produce (hopefully) better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

Individuals, or current approximations, are encoded as strings, chromosomes, composed over some alphabet(s), so that the genotypes (chromosome values) are uniquely mapped onto the decision variable (phenotypic) domain. The most commonly used representation in GAs is the binary alphabet {0, 1} although other representations can be used.

The process of genetic algorithm process is:

Step 1: Determine the number of chromosomes, generation, and mutation rate and crossover rate value

Step 2: Generate chromosome

Step 3: Process steps 4-7 until the number of generations is met

Step 4: Evaluation of fitness value of chromosomes by calculating objective function

Step 5: Chromosomes selection

Step 6: Crossover:- produce new chromosomes.

Step 7: Mutation

Step 8: New Chromosomes (Offspring)

Step 9: Solution (Best Chromosomes).

2. Related Work

Software Product Line techniques and tools allow the engineering of the software by reusing the software assets in a systematic way. For representing software product line, feature models (FM) [12] were introduced which abstractly modeled the common feature of the software assets. In common SPL, there are thousands of features which lead to the complex Feature Model. For instance, Linux Kernel FM has more than 6000 features [13]. Testing of SPL and Feature Model is difficult activity. A technique mutation analysis is used to evaluate the quality of the testing process. FM provides information to establish a mutation approach. It focuses on evaluating other testing process.

Test suits [14] [15] represent a set of software products and mutants can be considered as a fault. In Model Based Testing, it has been found that dissimilar test suits have higher fault detection than similar test suits. Removal of similar products reduces the size of the test suits by using similarity heuristics. To evaluate the degree of given test suits, use similarity heuristics to compare two products. An experiment conducted on both similar and dissimilar test suits towards feature model of different size which signify the higher ability of dissimilar test suits to detect the defect arise in the modified feature models. It evaluates the validity of similarity driven prioritization technique.

Testing a Software Product line is challenging due to combinatorial explosion of the number of products to consider. Testing of all products is feasible because the resources are limited. Reduction of number of products becomes necessary to test a reasonable value while trying to maximize the confidence in the products that are tested. Feature Model (FM) is used to test Software Product Line (SPL). Use mutation testing as a way to assess the similarity method. Feature Model represents the features of product line and the features represent the abstraction of software assets like functionality. Feature Model allows construction of software product by selecting features to be presented in the final product. Feature Model represent by Feature Diagram (FD). In Feature Diagram first make the hierarchy of features by graphical representation and then translate into prepositional logic.

In previous work, Feature Diagram of Mobile Phone with 10 features is taken to describe Product Line Testing. A product is said to be valid if it satisfy the Boolean formula of FM and invalid, if the products does not satisfy the formula of Feature Model. Formula of FM has clauses and literals. A clause is a constraint that has to be satisfied by given product and literal represent either a selected or unselected features. First make the hierarchy of features then change the hierarchy into clausal normal form (CNF). Mutation analysis approach evaluates the power of test cases to indicate behavior differences between the unaltered and altered artifact versions. The process of introducing mutants is called mutant analysis. While testing, if mutants can be detected then these mutants called killed otherwise live. The Mutation score is measured by the ratio of killed mutants by total introduced mutants. Distance metric is used in the base paper to evaluate the degree of similarity between any two products. Some formulas of feature model, clauses and literals are used to evaluate that the dissimilar test suits have higher mutant detection power than the similar ones.

Problem:-

- The testing process will be improved since the generated tests will be capable of finding all the introduced mutants.

3. Proposed Methodology

The model based testing is the technique to test the software through the model. The model has various test cases which are generated with the reverse engineering process. The product line testing has various communalities through which various end products are produced. Due to large size and associated communalities in the product line, it is very difficult to generate accurate test cases with the model-mutation based testing technique. In this work, improvement will be proposed in model-mutation based testing technique to reduce error in test cases generation. As described, model-mutation based technique take mutation values randomly and generate test case with best fitness function. Due to random selection of mutation values, error is raised at the time of test case generation. The proposed enhancement will be based on to select best mutation value for generation of test case. To select best value, technique of unsupervised learning will be applied which select value on the basis of type of software for which test cases are going to generate.

Figure 3.1 shows that there are five features of online shopping website are taken to generate test cases of these and detect error from these. We apply improved genetic algorithm to increase the error detection rate. The model based testing generates five test cases according to the five features of online shopping website.

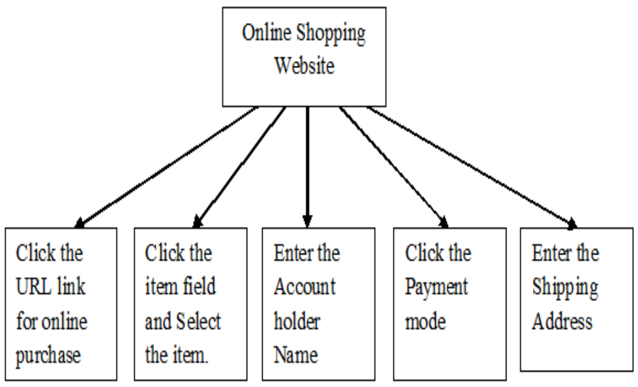
Online Shopping Website
Click the URL link for online purchase
Click the item field and Select the item.
Enter the Account holder Name
Click the Payment mode
Enter the Shipping Address

Figure 3.1: Features of online Shopping website

4. Experimental Results

Below table shows the experimental results of improved genetic algorithm. The number of errors detected in each test case according to the selection of feature is shown in table and also the error detection rate in test cases

Features Selected	Generate Test cases	% of Faults	Error Detection Rate
Click the URL link for online purchase.	One (1)	15.9110%	7.2689%
Click the URL link for online purchase. Click the item field and Select the item.	Two (1 and 2)	15.9110% 16.7360%	14.9141%
Click the URL link for online purchase. Click the item field and Select the item. Enter the Account holder Name	Three (1, 2 and 3)	15.9110% 16.7360% 19.9890%	24.078%
Click the URL link for online purchase. Click the item field and Select the item. Enter the Account holder Name. Click the Payment mode.	Four (1, 2, 3 and 4)	15.9110% 16.7360% 19.9890% 11.6310%	29.3591%
Click the URL link for online purchase. Click the item field and Select the item. Enter the Account holder Name. Click the Payment mode. Enter the Shipping Address	Five (1, 2, 3, 4 and 5)	15.9110% 16.7360% 19.9890% 11.6310% 22.8560%	39.8004%

5. Comparison with Existing Technique

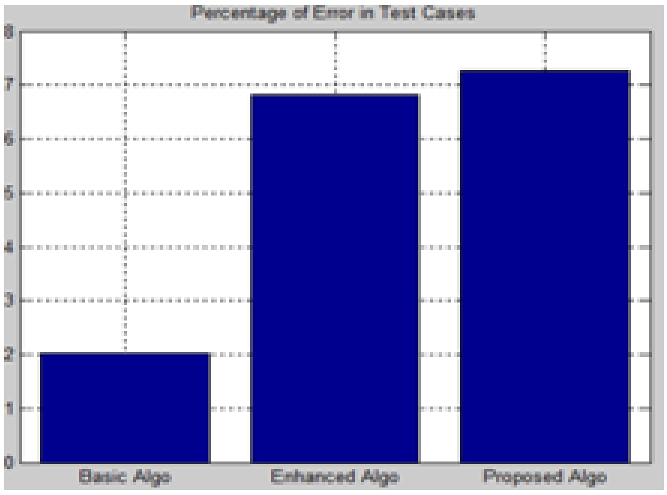

Figure 4.1: Comparison

As shown in figure 4.1, the bar graph shows the comparison of genetic and improved genetic algorithm. After selecting first feature first test case is generated automatically and Error detection rate of improved algorithm is greater than enhanced algorithm.

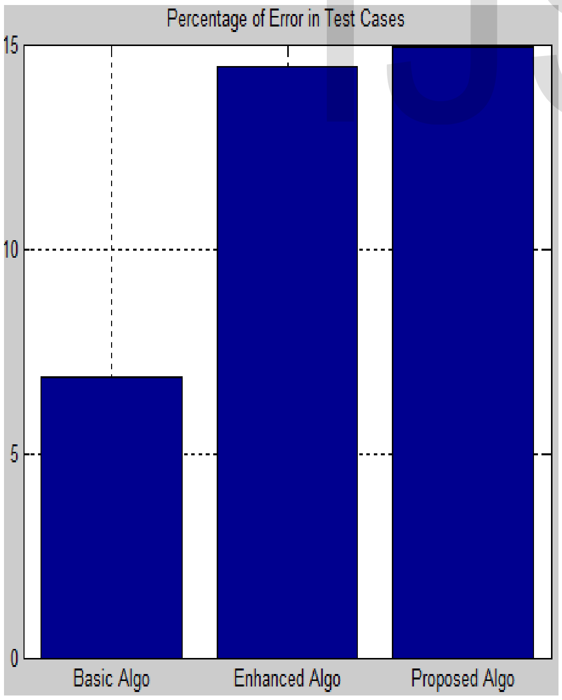

Figure 4.2: Comparison

As shown in figure 4.2 the bar graph shows the comparison of genetic and improved genetic algorithm. After selecting two features two test cases are generated automatically and Error detection rate of improved algorithm is greater than enhanced algorithm.

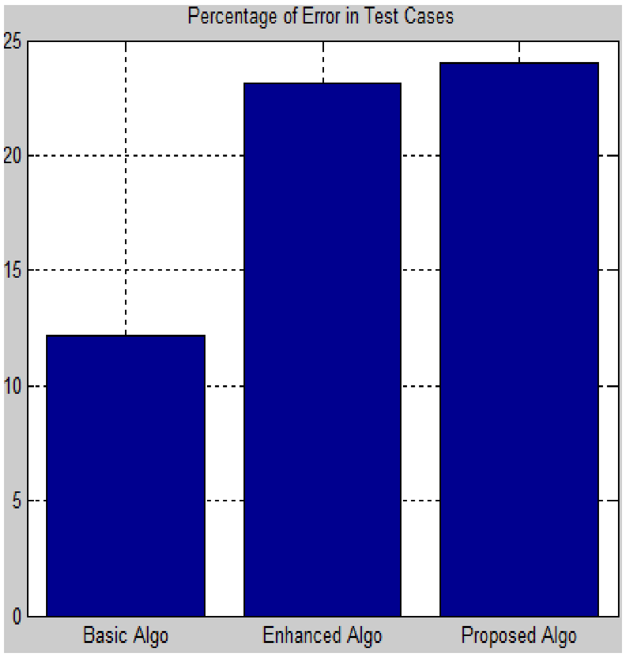

Figure 4.3: Comparison

As shown in figure 4.3, the bar graph shows the comparison of genetic and improved genetic algorithm. After selecting three features three test cases are generated automatically and Error detection rate of improved algorithm is greater than enhanced algorithm.

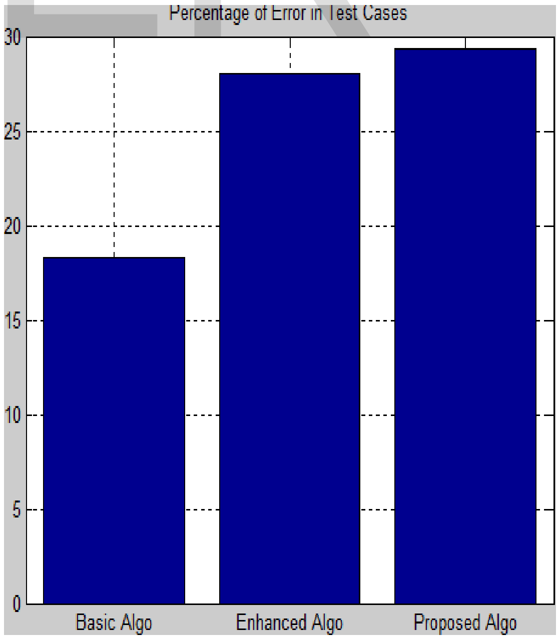

Figure 4.4: Comparison

As shown in figure 4.4, the bar graph shows the comparison of genetic and improved genetic algorithm. After selecting four features four test cases are generated automatically and

Error detection rate of improved algorithm is greater than enhanced algorithm.

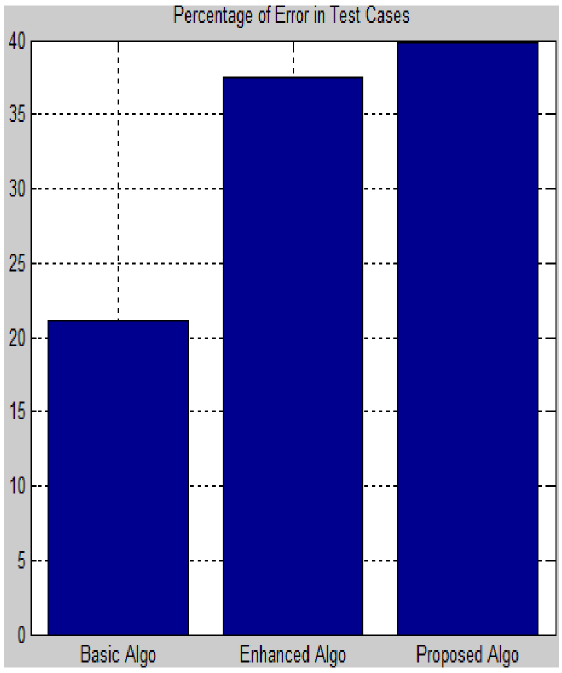

Figure 4.5: Comparison

As shown in figure 4.5, the bar graph shows the comparison of genetic and improved genetic algorithm. After selecting five features five test cases are generated automatically and Error detection rate of improved algorithm is greater than enhanced algorithm.

6. Conclusion and Future Scope

Models used to representing desire behavior of System under Test (SUT) and also represent test environment. MBT is mainly used to generate test cases automatically. Model based testing is basically used to describe the behavior of system under test. They focus upon model based testing and the simple process of how model based testing describes the fundamental behavior of system under test. Model based testing is very popular because it supports automated test case generation. In the existing techniques faults occurs in the generated test cases. In proposed technique, enhancement in Genetic algorithm has been done using unsupervised learning of neural technique to get better results of testing and increase error detection rate. In future, we can get better results by using back propagation algorithm and we can also enhance this algorithm by detecting more faults.

References

[1] Yanchun Sun, "*The Challenge and Practice of Creating Software Engineering Curriculum*", School of Electronics Engineering & Computer Science, Peking University, pp 497-501, IEEE 2011.

[2] Gaurav, Kestina Rai "*Software Testing Techniques for Test Case Generation*" International journal of Advanced Research in Computer Science and Software Engineering 2013 pp 261-265.

[3] S. Beydeda and V. Gruhn, "*An integrated testing technique for component-based software,*" ACS/IEEE International Conference on Computer Systems and Applications, June 2001, pp 328 – 334.

[4] Ibrahim K. El-Far and James A. Whittaker, "*Model-based Software Testing*", Florida Institute of Technology pp.1-22, 2001.

[5] OlliPekkaPuolitaival," *Model Based Testing Tools*", VTT Business from Technology.

[6] S. C. P. F. Fabbri, J. C. Maldonado, P. C. Masiero, M. E. Delamaro, and E. Wong, "*Mutation testing applied to validate specifications based on petri nets,*" in Proceedings of the IFIP TC6 Eighth International Conference on Formal Description Techniques VIII. London, UK, UK: Chapman & Hall, Ltd., 1996, pp. 329–337.

[7] K. Pohl, G. B¨ockle, and F. J. van der Linden, "*Software Product Line Engineering: Foundations, Principles and Techniques*", Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[8] P. Clements and L. Northrop, "*Software Product Lines: Practices and Patterns*", Addison Wesley, Reading, MA, USA, 2001.

[9] Denny Hermawanto ," *Genetic Algorithm for Solving Simple Mathematical Equality Problem*", Indonesian Institute of Sciences (LIPI), Indonesia, pp 1-10, 1997.

[10] V.Mary Sumalatha," *An Model Based Test Case Generation Technique Using Genetic Algorithms*", Gitam University, Visakhapatnam, Andhra Pradesh, India, pp 46-57,TIJCSE 2012.

[11] F. ipate, R. lefticaru," *Genetic Model based Testing:a Framework and a Case Study*", Department of Computer Science and Mathematics , University of Pite»sti, Romania, pp 209-227, 2008.

[12] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Carnegie-Mellon University Software Engineering Institute, Tech. Rep., Nov. 1990.

[13] T. Berger, S. She, R. Lotufo, A. Wasowski, and K. Czarnecki, "*Variability modeling in the real: a perspective from the operating systems domain*" in ASE, 2010, pp. 73–82.

[14] R. A. DeMillo and A. J. Offutt, "*Constraint-based automatic test data generation*" IEEE Trans. Softw. Eng., vol. 17, no. 9, pp. 900–910, Sep. 1991.

[15] M. Papadakis and N. Malevris, " *Mutation based test case generation via a path selection strategy*" Inf. Softw. Technol., vol. 54, no. 9, pp. 915–932, Sep. 2012