DOI: http://dx.doi.org/10.26483/ijarcs.v9i3.6046

ISSN No. 0976-5697

Volume 9, No. 3, May-June 2018

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

IMPROVEMENT IN MUTATION TESTING USING BOLTZMANN LEARNING FOR FAULT PREDICTION IN TEST CASES

Akanksha Tiwari
M. Tech Student (CS-SE)
Babu Banarasi DasUniversity
Lucknow, India

Abhinav Singh
Assistant Professor CSE Dept.
Babu Banarasi Das University
Lucknow,India

Abstract: The software engineering is the technology to process the software and perform various operations on that software. The testing the important application of software engineering in which test cases are applied to detect faults from the software . In the recent times, it is been analyzed that faults may also arise in the test cases which are used for the fault detection. In this work, mutation algorithm is applied for the detection of faults from the software. To improve performance of mutation algorithm in terms of fault detection rate the technique of back propagation is applied which learn from the precious experience and drive new values. The system is tested on 10 test cases and simulation is performed in MATLAB. The simulation results show that the fault detection rate is increased and execution time is reduced.

Keywords: structurally complex,mutation,faults.

INTRODUCTION

Software is the program or set of programs. It is different from the program in many ways. As in software many things are includes: as it consists of the programs, the complete documentation of that program, the procedure that is use to set up the software and the various operation of the software system. Any program is the subset of the software. As on the other hand, program is the combination of source code and object code. Software engineering is a systematic, disciplined and quantifiable approach to design, development and maintenance of software within cost, time and other constraints [1]. Systematic and disciplined means the developer apply well understood techniques in an organized and discipline way. Cost, time and other constraint means software developer must ensure that software must be produced within limited budget and time. Other constraint means software developer must have great knowledge about what is required to produce a system and how each activity should take place in systematic way during development process. As software requirement is increases day by day. So it is necessary to maintain the good quality software. To develop good quality software, software engineering is required. For this, the developer's needs to adopt the software engineering concepts, strategies, and practices to avoid the conflicts that are occur during the development process. Software engineering is an approach to develop, maintain and operate the software. The software development plays a crucial role in software engineering. Many specific techniques are required to develop software. The most common thing in development process is the requirement gathering and customer needs [2]. If a developer fails to complete the needs of the customer than he or she may fails to develop good quality software. Software can be said to of good quality, if it is able to fulfill the needs of the customer. The customer can be satisfied in terms of quality, cost and design of the system software. Many developers adopt the techniques like systematic and organized approach to develop software. A test case is set of procedure use to test the software. Test case is a set of condition under which under which a software tester determine whether the application or software system is working correctly or not [3]. To design a test case for particular software the designer must design positive or negative test case for the software. Positive test cases are design to check software under normal condition and negative test case are design to check software at extreme condition. The order of test case execution affects the time, at which goal of testing are fulfilled. If the goal is fault detection then an improper execution order might reveal most of fault late which leads to delay in bug fixing activity and the delivery of software.Software testing is a procedure of testing or comparing the actual outcome with the expected outcome. Testing of the software is done in order to check the correct functionality of the system or project.If the testing will not be performed then system may lead to catastrophic or improper results in the field. So it's better to check or test the system earlier, so that the excellent results can be produced. Software Defects Prediction (SDP) includes software metrics, their attributes like line of code etc. The main goal of software defects prediction model includes ordering new software modules based on their defect-proneness and classifying them whether it is new software or not. The SDP process consists of two parts: Data Collection and Model Construction [4].

Mutation testing is a kind of testing in which, the application is tested for the code that was modified after fixing a particular bug/defect. It also helps in finding out which code and which strategy of coding can help in developing the functionality effectively. Mutation testing is by definition a criterion of testing: the tests are either generated by accident

until the enough numbers of mutations are killed, or the tests are designed to purposely kill surviving mutations. In the latter case, testing of mutations can be categorized as a code based technique [5]. For this technique to be effective, a great number of mutants must be produced in a systematic way. If the testing is successful in determining a difference between a program and a mutant, another one is killed.Mutation testing is an accepted technique for improving the quality of developed software. It is a white box testing technique that is applicable during unit testing of program and involves creation and execution of different versions known as mutants of a program. In case of white-box testing, this can be done by making systematic changes in the source code of the component [6]. For black-box testing, the specification of the component has to be systematically manipulated to generate test cases. The work is focused on specification-oriented mutant generation to validate the behavioral specification of the component.

LITERATURE REVIEW

Xiaoxing Yang et.al (2015) explained in this paper that construction of previous work, and further study whether the idea of directly optimizing the model performance measure can benefit software defect prediction model construction. The work includes two aspects: one is a novel application of the learning-to-rank approach to real-world data sets for software defect prediction, and the other is a comprehensive evaluation and comparison of the learning-to-rank method against other algorithms that have been used for predicting the order of software modules according to the predicted number of defects. The empirical studies demonstrate the effectiveness of directly optimizing the model performance measure for the learning-to-rank approach to construct defect prediction models for the ranking task [7].

Shaik Nafeez Umar et.al (2013) described in this paper that statistical model, how it will accurately predict the defects for upcoming software releases or projects. They have used 20 past release data points of software project, 5 parameters and build a model by applying descriptive statistics, correlation and multiple linear regression models with 95% confidence intervals (CI). In this appropriate multiple linear regression model the R-square value was 0.91 and its Standard Error is 5.90%. The Software testing defect prediction model is now being used to predict defects at various testing projects and operational releases. We have found 90.76%, % precision between actual and predicted defects [8].

Muhammad Dhiauddin et.al (2012) described in this paper that an initial effort of building a prediction model for defects in system testing carried out by an independent testing team. The motivation to have such defect prediction model is to serve as early quality indicator of the software entering system testing and assist the testing team to manage and control test execution activities. Metrics collected from prior phases to system testing are identified and analyzed to determine the potential predictors for building the model. The selected metrics are then put into regression analysis to generate several mathematical equations. Mathematical equation that has p-value of less than 0.05 with Rsquared and R-squared (adjusted) more than 90% is selected as the desired prediction model for system testing defects. This model is verified using new projects to confirm that it is fit for actual implementation [9].

Mrinal Singh Rawat, Sanjay Kumar Dubey (2012) proposed in this paper that software defects may lead to degradation of the quality which might be the underlying cause of failure. In today's cutting edge competition it is necessary to make conscious efforts to control and minimize defects in software engineering. However, these efforts cost money, time and resources. This paper identifies causative factors which in turn suggest the remedies to improve software quality and productivity. The paper also showcases on how the various defect prediction models are implemented resulting in reduced magnitude of defects [10].

Christopher Henard, (2013) explained in this paper that mass customization and economics force to design software product line. It reuses its assets by adding more features in it. There are many constraints which are represented by the feature model and allow tailored software products. It contains thousands and billions of software products. As a result, due to large size of products, product line is a challenging. In this paper, existing technique based on the feature model of the product line by selecting limited set of products. In this paper test suites are used to detect such errors. In particular, propose two mutation operators to derive erroneous feature models (mutants) from an original feature model and assess the capability of the generated original test suite to kill the mutants. Experimental results demonstrate that dissimilar tests suites have higher mutant detection ability than similar ones, thus validating the relevance of similarity-driven product line testing [11].

Jan Peleska, (2013) explained in this paper that model based testing is one of the leading technologies. The key factors are essential for industrial scale application of MBT. Both are identified from the feature extraction. With former view they had described techniques for automated test cases, test data and test procedure generation for concurrent reactive real times system which enables for MBT. Their experience introduced MBT approaches in MBT for testing teams. There are many scientific problems to improve the acceptance and effectiveness of MBT [12].

RESEARCH METHODOLOGY

The model based mutation analysis is the technique which generates complex and faulty test data to perform automated testing. In the presence of faulty data, it is very difficult to perform efficient testing and analyze system robustness. In this work, the faults from the generated test cases will be detected using the Boltzmann learning. The Boltzmann learning will be the improvement in the model based mutation analysis to remove faults from the generated test cases which will lead to increase system's testing efficiency. The systems that include units connected symmetrically amongst each other are known as Boltzmann machines. The stochastic decisions that include decisions related to being on or off are made by these systems. The complex distributions within the observed data are discovered with the help simple learning algorithms provided by this mechanism. For various scientific tasks, imperative learning or inference of Boltzmann machines is provided. The settlement of weights on connections and thresholds for inference issues is provided through this approach. A cost function is represented with the help of these weights. In

order to solve some advanced problems, the inference in Boltzmann machines is used more frequently. There is a need to provide expectations in one unit and correlations amongst two units for performing learning in Boltzmann machines. It is important to provide precise estimation of correlations. The percentage of faults within the test cases can be predicted using fault prediction mechanism. A learn-to-rank algorithm is used in order to detect faults from test cases within this work. There are three important steps included within the learn-to-rank algorithm. The first step is selection of population. The second step is calculation of mutation value. The last step is calculation of fitness value. The calculation of fitness value depends upon the initial population value which is selected randomly. In this work, Back Propagation technique is applied in which system learns from the experience values and derives new values. The selection of population value is not random. It depends upon the system condition which is derived using back propagation algorithm.

PROPOSED ALGORITHM

```
 Init population P (t)
            evaluate P (t);
            t := 0;
Network  ConstructNetworkLayers()

 InitializeWeights(Network, test cases)

For ( i=0;i=testcases;i++)

    SelectInputPattern(Input fault values)

    ForwardPropagate(p)

   BackwardPropagateError(P)

   UpdateWeights(P )

End

Return (P)
            while not done do
                        t := t + 1;
                P' := test case P (t);
 recombine P' (t);
mutate P' (t);
                evaluate P' (t);
                P := survive P,P' (t);
    end
```

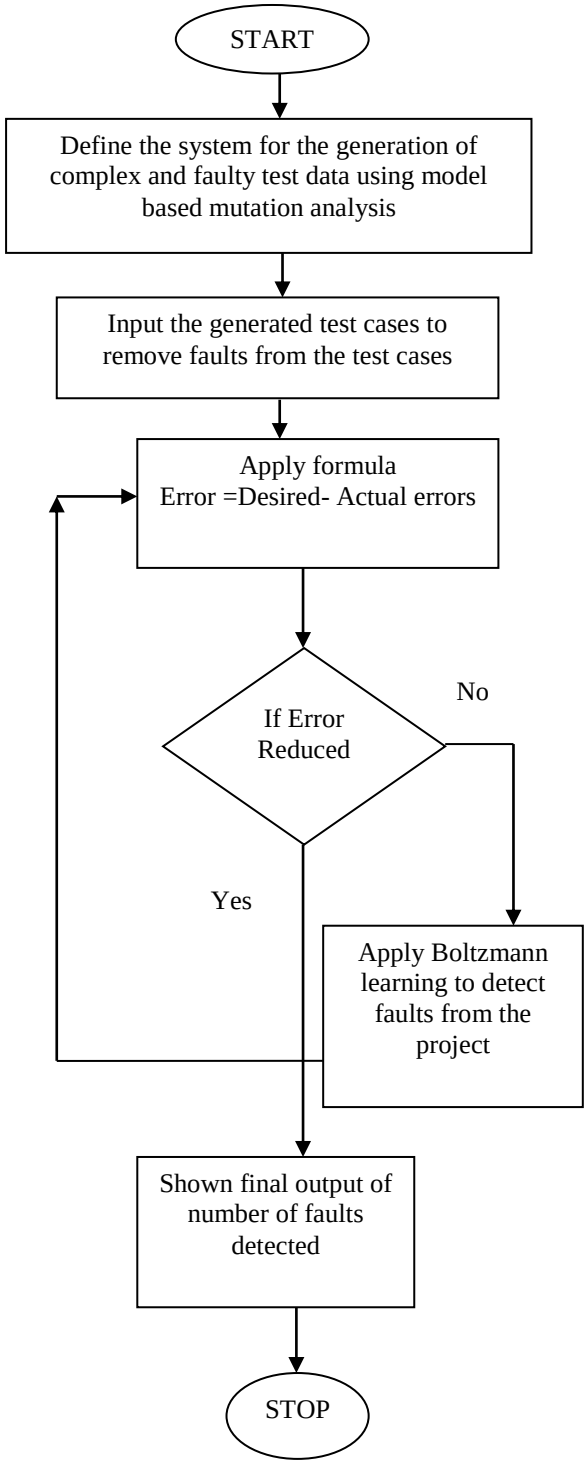

Fig: 1 Flowchart of Proposed Work

EXPERIMENTAL RESULTS

The proposed work has been implemented in MATLAB and the results have been analyzed by conducting various experiments.

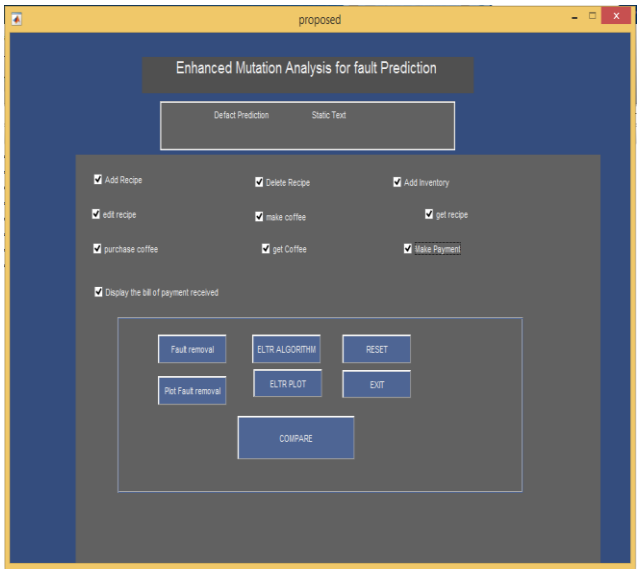

Fig 2: Selection of test cases

As shown in figure 2, the proposed algorithm is applied in which Boltzmann learning algorithm is used with the mutation analysis. The Boltzmann learning algorithm will learn from the previous experiences and drive new values

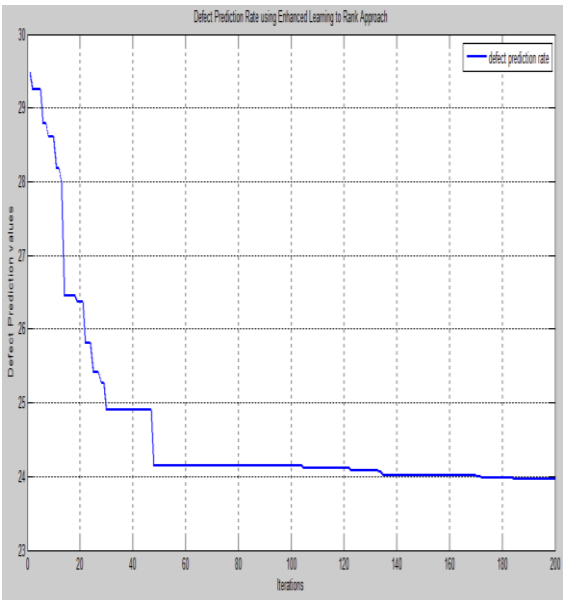

Fig 3: Fault Prediction

In figure 3, the defect prediction values are shown along with number of iterations through this execution graph.

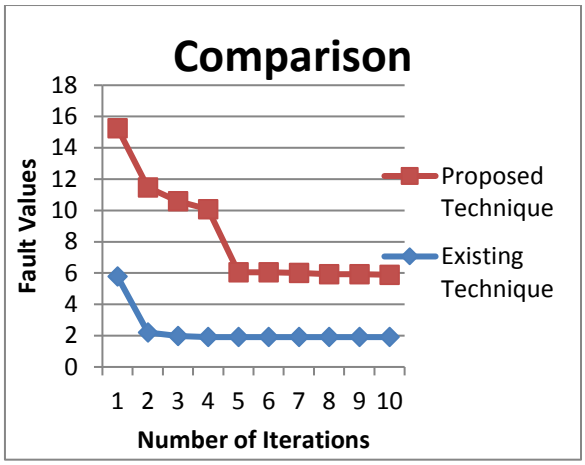

Fig 4: Comparison Graph

As shown in Figure 4, the proposed and existing algorithms are compared in terms of number of faults detected. It has been analyzed that number of fault detection is increased in proposed algorithm as compared to existing algorithm.

CONCLUSION

The fault detection is the testing technique which is applied to detect faults from the software or from the input test cases. The Mutation is the algorithm which is applied for the faults in the software. This algorithm selects population randomly which reduce fault detection rate. In this work, technique of back propagation is applied in which system learns from the previous experiences and drive new values. This leads to improve fault detection rate and reduce execution time. In future technique will be proposed which is based on bio-inspired techniques for the fault detection rate.

REFERENCES

[1] M. Mendonca, A. Wasowski, and K. Czarnecki, “Sat-based analysis of feature models is easy” in Proceedings of the 13th International Software Product Line Conference, ser. SPLC ’09. Pittsburgh, PA, USA: Carnegie Mellon University, 2009, pp. 231–240.

[2] G. Perrouin, S. Sen, J. Klein, B. Baudry, and Y. L. Traon, “Automated and scalable t wise test case generation strategies for software product line” in ICST. IEEE Computer Society, 2010, pp. 459–468.

[3] Mangal, B. S, “Analyzing Test Case Selection using Proposed Hybrid Technique based on BCO and Genetic Algorithm and a Comparison with ACO”, IJCA, 2012.

[4] Suman and Seema, “A Genetic Algorithm for Regression Test Sequence Optimization”, International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 7, September 2012

[5] Bohringer, Christoph, Rutherford, Thomas F, “Integrating Bottom-Up into Top-Down: A Mixed Complementary Approach”, 2003

[6] Christian Murphy, Gail Kaiser, Ian Vo, Matt Chu “Quality Assurance of Software Applications Using the In Vivo Testing Approach”, 2004.

[7] Ghiduk, A. S, “A New Software Data-Flow Testing Approach via Ant Colony Algorithms” Universal Journal of Computer Science and Engineering Technology, 1, 64-72, 2011.

[8] Xiaoxing Yang, Ke Tang, Senior Member, IEEE, and Xin Yao, “A Learning-to-Rank Approach to Software Defect

Prediction", IEEE TRANSACTIONS ON RELIABILITY, VOL. 64, NO. 1, MARCH 2015

[9] Shaik Nafeez Umar, "Software Testing Defect Prediction Model- A Practical Approach", IJRET: International Journal of Research in Engineering and Technology, Volume: 02 Issue: 05 | May-2013

[10] Muhammad Dhiauddin, Mohamed Suffian, Suhaimi Ibrahim, "A Prediction Model for System Testing Defects using Regression Analysis", International Journal of Soft Computing And Software Engineering (JSCSE) e-ISSN: 2251-7545 Vol.2,o.7, 2012

[11] Mrinal Singh Rawat, Sanjay Kumar Dubey , "Software Defect Prediction Models for Quality Improvement: A Literature Study", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 2, September 2012 ISSN (Online): 1694-0814

[12] Christopher Henard, Mike Papadakis∗, Gilles Perrouin, Jacques Klein, and Yves Le Traon "Assessing Software Product Line Testing via Model-based Mutation: An Application to Similarity Testing", 2013