Automatic Test Suite Generation for Regression Testing

[1]S.Poornima, [2]Selvakumarsamy
[1]M.TECH Student, [2]Assistant Professor
[1,2]Dept of Software Engineering, SRM University, Kattangulathur, Chennai, India-603203
[1]poornimasit@gmail.com, [2],selvakumarasamy.s@ktr.srmuniv.ac.in

Abstract- **Software testing is the most important technique used in industries. In testing process, the targeted application is tested by the tester in order to verify whether the actual result is same as the expected result as per the requirements. Some of the crucial challenges confronted by software testers during regression testing are the lack of test cases, test data for the modified version of the application. To overcome this problem, we propose a method to generate the optimized Test suite based on the comparison of the initial and modified versions of the software. This paper has several advantages like maximum code coverage and number of iterations.**

Index Terms - **Software testing, test data, test case, Optimized Test Suite.**

I. INTRODUCTION

Maintaining the quality of the software is one of the most challenging tasks in industries. There are various development processes carried out in the industry in order to maintain the quality of the software. In such case, Software testing plays an important role in order to maintain the quality of the software. Software Testers test the application based on the test conditions and test cases to check whether the actual result is same as the expected result of the targeted software. Regression testing is performed for the modified version of the targeted application. This process is carried out by testing the application based on the test cases of the initial version. The purpose of the regression testing is to find out the errors and to check whether any changes have affected the working of the application.

In this paper, we propose an automatic generation of optimized test suite based on the initial and the modified version of the application. In proposed method, the initial test cases are generated randomly. Then the generated test cases are optimized using genetic algorithm until they provide the evaluation result.

II. RELATED WORK

The purpose of this paper is to design and develop the optimized test suite for regression testing that can help to reduce the testing effort of the tester and to minimize the testing time. Various techniques [3] is used to evaluate the maximum code coverage and the number of iterations to be covered. Various approaches used in this paper achieve a high level of branch Coverage which generates the optimized Test Suite.

a. Coverage Measures

The adequacy of testing is evaluated by the coverage measure [3] that describes the degree to which a program's source code has been tested. The adequacy can be achieved by covering branch, condition coverage and statement coverage. For each coverage there is corresponding results. There is another coverage measure called boundary value coverage. In the boundary value coverage measure, test data are designed to run on the boundary values of the true and false results of the condition statement

b. Mutation Analysis

In mutation analysis, which evaluates the quality of the test case set, program errors are intentionally injected into a program [5]. The error-injection in the program is called a mutant. Since there are several possibilities in error injection, one original code will generate some mutants.

c. SBST

Search Based Technique [2] has been applied to test the software in order to cover all the objects. In object oriented software it is infeasible to cover some branches than others while generating the optimized test suite.

d. Mutation Testing

Mutation testing [3] is often considered a worthwhile test goal and has been used in a search-based test generation environment [1]. When test cases are generated it is important that it should cover all the criteria. Otherwise, it has been proven that random testing would fare better under some scalability models.

III. PROPOSED METHODOLOGY

Fig 1 shows the overall structure of the proposed method. The initial version of the application is given as input to the tool based on the given file the input parameters are created. The input parameters are generate based on the functions such as code coverage

and number of iterations to be covered for example the no of iterations to be covered is 10 iterations and maximum code coverage should be above 80 percent. Based on the generated input parameter Random test cases are generated.
Then the randomly generated test cases is applied to the application in order to verify the testing time, coverage ratio and no of faults. The above result is provided to each and every randomly generated test case.
In order to obtain the Optimized test suite for regression testing input the modified version of the application then apply the genetic algorithm. Genetic algorithm is one the optimization method used.

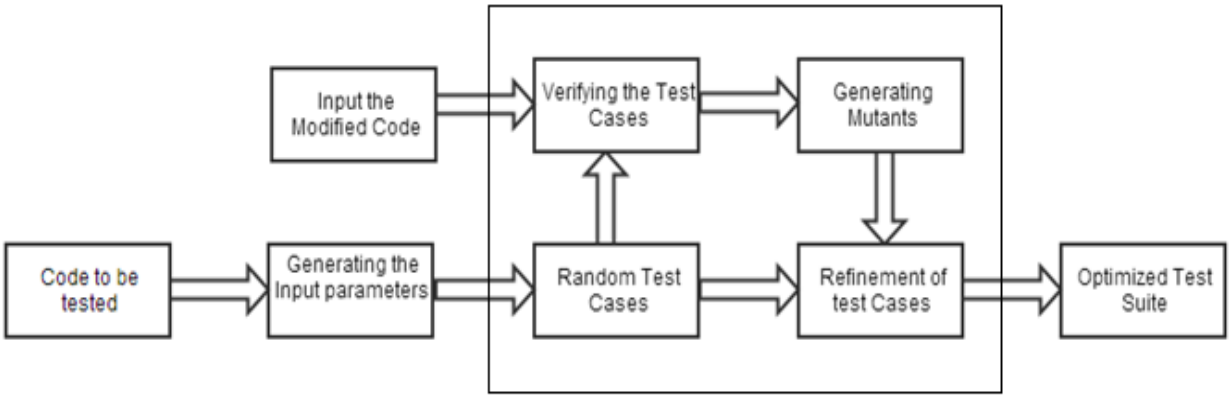


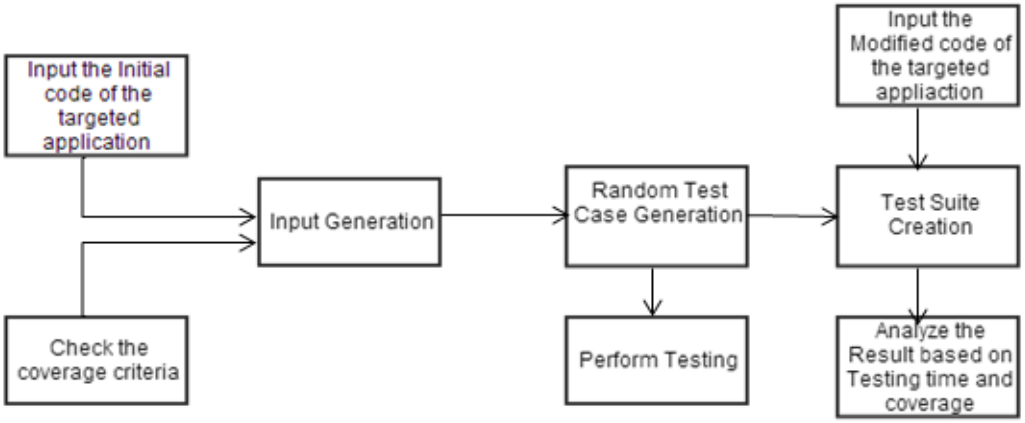
Fig.1.Structure of Proposed method

To apply the Genetic algorithm first we should satisfy the following two requirements.
Representation of the solution
Evaluate the fitness solution(i.e. Should satisfy the Coverage Criteria)
If the above requirements are satisfied then the operations of the genetic algorithm can be applied to randomly generated test cases.
1. Choose the initial population (i.e. randomly generated test cases)
2. Check whether the coverage criteria are met (maximum code coverage and iterations)
3. If the Coverage Criteria is not met then apply the mutation operation of genetic algorithm.
   a. Choose the best fit parents for the reproduction based upon the testing time.
   b. Apply the Cross over and mutation process in order to generate the new off springs.
   c. Verify the newly generated offspring with the coverage criteria.

Thus optimized test suite is generated by using the genetic algorithm.

3. EXPERIMENTAL SETUP

The initial document of the application is given as input to the tool. The random test cases are generated with the help of



input parameters. The Random test cases undergo testing in order find out the testing time, coverage criteria and no of faults for each and every randomly generated test case. Then the newly modified application is added to the tool based upon the randomly generated the test cases test suite is generated. For the creation of test suite initially the randomly generated test cases should be verified to check whether the coverage criteria is met if not the process of genetic algorithm is applied.

IV. EXPERIMENTAL RESULT

The Evaluation result are represented by testing time, coverage criteria and no of faults. The testing time represents the starting and ending time taken by the each test case to test the application. The Coverage criteria represents the no of functions covered in a particular branch.

V. CONCLUSION AND FUTURE WORK

In this paper, we presented automatic generation of test cases by focusing on branch coverage. We randomly generated the test cases for the initial application and then refined them using genetic algorithm for the modified version of the application.
The quality is analyzed by various measures such as testing time, coverage measures and no of faults. Our Proposed can only deal with branch condition coverage and few mutant operators it does not deal with various other coverages.

VI. REFERENCE

[1] G. Fraser, A. Arcuri, "EVOLUTIONARY GENERATION OF WHOLE TEST SUITES," Proc. 11th Int'l Conf. Quality Software, pp. 31-40, 2011.
[2] S. Ali, L. Briand, H. Hemmati, and R. Panesar-Walawege, "A SYSTEMATIC REVIEW OF THE APPLICATION AND EMPIRICAL INVESTIGATION OF SEARCH-BASED TEST-CASE GENERATION," IEEE Trans. Software Eng., vol. 36, no. 6, pp. 742-762, Nov./Dec. 2010.
[3] Gordon Fraser, Andrea Arcuri, " WHOLE TEST SUITE GENERATION," IEEE Transactions On Software Engineering, Vol. 39, No. 2, February 2013
[4] B. Baudry, F. Fleurey, J.-M. Je´ze´quel, and Y. Le Traon, "AUTOMATIC TEST CASES OPTIMIZATION: A BACTERIOLOGIC ALGORITHM," IEEE Software, vol. 22, no. 2, pp. 76-82, Mar./Apr. 2005.
[5] Hirohide Haga, Akihisa Suehiro, "AUTOMATIC TEST CASE GENERATION BASED ON GENETIC ALGORITHM AND MUTATION ANALYSIS", 2012 IEEE International Conference on Control System, Computing and Engineering, 23 - 25 Nov. 2012, Penang, Malaysia
[6] B. Selic, and J. Rumbaugh, "UML FOR MODELING COMPLEX REAL TIME SYSTEMS", Available Online Via www.rational.com/Products/Whitepapers/100230.Jsp.