Regression Testing: Advanced Tool for Testing the Software's Bugs

Ritu
Teacher-IT/ITeS,
GGSSS, NIT-5
Faridabad

Anuradha Pillai
Asisstant Professor,
YMCAUST

Charu Virmani
Associate Professor
MRIIRS

Deepika Punj
Vocational
Asisstant Professor
YMCAUST

ABSTRACT
In present scenario**,** advancement in the area of information technology results high demand for new software or modifications in the existing software. Software testing is the process of testing the quality and reliability of the software through execution of the program and eliminates the bugs or errors, which are present in Software. For achieving correctness, reliability, good quality of software and its usability, it is essential that the software measure according to capability, maintainability, reusability and testability.

Keywords
Testing, Software, Bugg, Regression.

1. INTRODUCTION
With the rapid development of modern information technology, software products find their ways into all areas of society. The quality of software products naturally become the focus of common concern as well as the key problem in software engineering. Researchers in the field of software testing pay more attention to testing the software and increasing its reliability. There still exist many difficulties and challenges to test the software more accurately. Almost in all the software, the testing task becomes a challenging task. Its purpose is to evaluate the software to demonstrate that it meets the requirements.

Regression testing is commonly used to verify the quality of software in the process of software development. It's quite important and expensive but, still providing a powerful mechanism to maintain and manage the test cases and test process to reduce the cost of regression testing, and to improve the efficiency of regression testing are the emphases and difficulties in the whole software testing. The genetic algorithm is a search algorithm based on the mechanics of natural selection and natural genetics. [6] Genetic algorithms are the heuristic search algorithms that are used to solve a variety of optimization problems. Genetic algorithms mimic the process of natural biological evolution and the Darwin's principal of the survival of the fittest. The genetic algorithms cause a population of individuals to evolve from one generation to another, each time allowing the best characteristics of one generation to pass to the next generation.

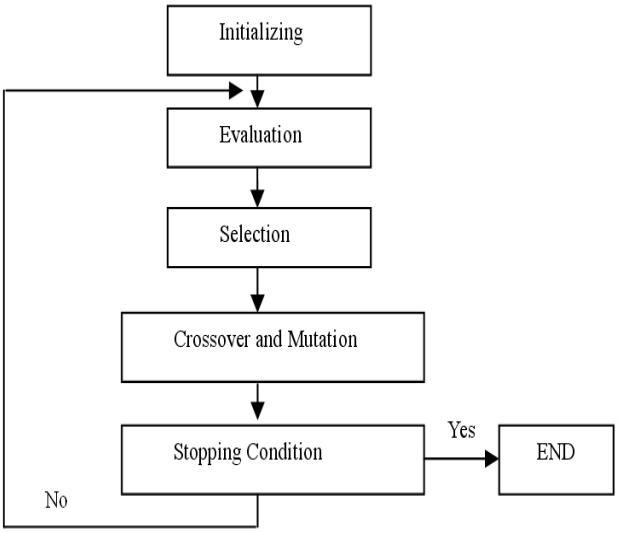

Fig. 1 Block Diagram of Genetic Algorithm [13]

2. LITERATURE REVIEW
Software testing is an important aspect in software development process. The goal and need of software testing is "affirm the quality of software systems by systematically exercising the software in carefully controlled circumstances". Testing of software is an important and critical part of the software development process, on which the quality and reliability of the delivered software strictly depend. [4] The empirical research on test case prioritization technique reveals that there were three steps considered as most important for software testing i.e. a) Test case selection, b) Test case prioritization, and c) Test suite minimization. [14] The test case prioritization is very important task while testing the software quality. Earlier, tree model was proposed for the prioritization purpose as well as for test suite minimization. The prioritization of test cases has been done accordingly to the number of call-tree paths covered by each test case. [15] In the past two decades, the various methods for prioritization was proposed and developed that are a) Based on integrates coverage [10], b) The greedy-based prioritization approach [8], c) Greedy-based prioritization approach with regression test selection [16], Meta-heuristics for test case prioritization [9], Code-coverage TCP techniques [7], Regression test suite prioritization algorithm [11], Component Based Software Development (CBSD) [1] etc. There are several aspects of the test case prioritization problem as follows[5]:

- To increase the rate of fault detection of a test suite that is, the likelihood of revealing faults earlier in a run of regression tests using that test suite.
- To increase the coverage of coverable code in the system under test at a faster rate, thus allowing a

code coverage criterion to be met earlier in the test process.

- To increase their confidence in the reliability of the system under test at a faster rate.
- To increase the rate at which high-risk faults are detected by a test suite, thus locating such faults earlier in the testing process.
- To increase the likelihood of revealing faults related to specific code changes earlier in the regression testing process.

Genetic Algorithm approach is very useful to prioritize the test suite with the aim of maximizing the number of detected faults. The algorithm calculates average faults found per minute. The results are illustrated to determine the effectiveness of prioritized and non-prioritized case with the help of APFD metric. [3] GA considers an optimization problem as the environment where feasible solutions are the individuals living in that environment. The degree of adaptation of an individual to its environment is the counterpart of the fitness function evaluated on a solution. Similarly, a set of feasible solutions takes the place of a population of organisms. An individual is a string of binary digits or some other set of symbols drawn from a finite set. Each encoded individual in the population may be viewed as a representation of a particular solution to a problem. [17] While prioritizing the test cases the two factors are considered for system level prioritization. The first factor is the rate of fault detection, which is the average number of faults per minute by a test case. The second factor is the fault impact: Testing efficiency can be improved by focusing on the test case that is likely to contain high number of severe faults. So, for each fault severity value was assigned based on the impact of the fault on the product. [12] The 21 different techniques for code based test case prioritization and classified into three different groups i.e. comparator group, statement level group and function level group were explained earlier by researchers. [2,5]

Steps in Genetic algorithm:

- Population Generation
- Selection
    a) Elitist Selection
    b) Roulette Wheel Selection
    c) Tournament Selection
- Reproduction Operators
- Crossover
    a. One point crossover
    b. Two point Crossover
    c. Uniform Crossover
    d. Cut and Splice Crossover
- Mutation
- Termination Condition
- Fitness Function Construction

3. DESIGN

3.1 Problem Formulation

In present era, software testing is essential to verify and confirm the availability of good quality software programmes due to high demand. A good testing process is one that has a high probability of finding an as-yet undiscovered error. The software testing should aim to suggest changes or modifications if required, thus adding value to the entire process. The objective of this paper is to design tests that systematically uncover different classes of errors and do so with a minimum amount of time and effort. Performance requirements are required as it specified in specification document to ensure the software quality and reliability based on the data collected during testing.

4. METHODOLOGY

In regression testing, the software testing is done when changes are made in the software. It may be noted that not all the changes are of equal importance. The changes in the conditional statements or decision nodes are more important as compared to the other changes. The number of nodes changed may determine the fitness in this case. It may also be noted that in case of insertion or deletion of nodes the change is considered more important as compared to other cases. The work dwells upon these changes to calculate the fitness value of the mapped chromosomes. The following steps depict the technique:

1. Calculate coverage as decision node. T is the coverage may remain same or its cost in terms of number of nodes may increase or decrease; where T_i is the test case and T is the test case suite.
2. Calculate the change in coverage for test case in test case suite. More the number of changes or more is the number of effected variable more is the value of deltas, the parameter which will determine the fitness value.
3. The change in coverage fitness value α, as in regression testing changes are more important and those which are affected by changes are to be considered first.
4. All the test cases will be represented by binary chromosomes. A chromosome will give the set of test cases which will determine the set of test cases which forms a test suite.
5. The fitness of those chromosomes evaluated by Step 3. More the number of changes in conditional statements, more is the value of deltas and hence less is $e^{\Delta,}$ and hence more is $1/(1+e^{\Delta})$.
6. New chromosome will be generated by crossover and mutation operations.
7. Reproduction is done by rollet wheel selection.
8. The Test case suites are selected by applying GA.

5. RESULT AND CONCLUSION

The basic premise of this proposed work was that the conditional changes were more important. From the literature study, it was found the gap in existing technique because a little work has been done on conditional nodes. So-as to fill the gap, a novel technique is proposed, and establishing the fact that the fault finding capability improved by applying the proposed technique in the previous section. The verification of proposed technique was done on a benchmark program and

test data was taken from a professional thereby instilling the confidence in the technique. Finally, it is concluded that the new techniques always need the improvements and their constraints validation.

6. REFRENCES

[1] A. Acharya, D. P. Mohapatra and N. Panda, "Model based test case prioritization for testing component dependency in CBSD using UML sequence diagram*", IJACSA*, Vol. 1 (3), pp. 108-113, December, 2010.

[2] Alexey G. Malishevsky, Joseph R. Ruthruff, Gregg Rothermel, Sebastian Elbaum. "Cost cognizant Test Case Prioritization", *published in the proceedings of the IEEE International Conference on Software Maintenance*, 2006.

[3] Arvinder Kaur and Shubhra Goyal. "A Genetic Algorithm For Regression Test Case Prioritization Using Code Coverage", *International Journal of Advanced Science and Technology*, Vol. 29, April 2011.

[4] E. F. Miller, "Introduction to software testing technology", *Software Testing & Validation Techniques*, pp. 4 – 16, 1981.

[5] G. Rothermel, R. H. Untch, C. Chu, and M.J. Harrold, "Test Case Prioritization: An Empirical Study", *published in the proceedings of the Software Maintenance*, Oxford, UK, pp. 179-188, Sept. 1999.

[6] Goldberg, D. E. "Genetic Algorithms in Search, Optimization, and Machine Learning", Boston: Addison-Wesley.

[7] H. Srikanth, L. Williams and J. Osborne. "System Test Case Prioritization of New and Regression Test Cases", IEEE, 2005, pp.64-73.

[8] Jones J.A. and Harrold M.J. "Test-suite reduction and prioritization for modified condition/decision coverage", *published in the proceedings of International Conference on Software Maintenance (ICSM 2001),* IEEE Computer Society Press, pp. 92–101, 2001.

[9] Li Z., Harman M. and Hierons R.M. "Search Algorithms for Regression Test Case Prioritization", *IEEE Transactions on Software Engineering*, Vol. 33(4), pp. 225–237, 2007.

[10] Leon D. and Podgurski A. "A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases", *published in the proceedings of the IEEE International Symposium on Software Reliability Engineering (ISSRE 2003)*, IEEE Computer Society Press, pp. 442–456, 2003.

[11] Praveen Ranjan Srivastava. "Test Case Prioritization", *Journal of Theoretical and Applied Information Technology*, pp. 178-181, 2008.

[12] R. Kavitha, Dr. N. Sureshkumar. "Test Case Prioritization for Regression Testing based on Severity of Fault", *International Journal on Computer Science and Engineering, 2*010.

[13] Ruchika Malhotra and Mohit Garg. "An Adequacy Based Test Data Generation Technique Using Genetic Algorithms", *Journal of Information Processing Systems*, Vol.7, No.2, pp. 363-384, June 2011

[14] S. K. Swain. "Test Case Prioritization Based on UML Sequence and Activity Diagrams", *PhD thesis submitted at KIIT University*, 2010.

[15] Siavash Mirarab and Ladan Tahvildari. "An empirical study on bayesian network-based approach for test case prioritization", *published in the proceedings of the International Conference on Software Testing, Verification, and Validation*, IEEE Computer Society, pp. 278–287, Washington, DC, USA, 2008.

[16] Smith A., Geiger J., Kapfhammer G.M. and Soffa M.L. "Test suite reduction and prioritization with call trees", *published in the proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE)*, ACM Press, 2007.

[17] Tomassini, M. "Parallel and Distributed Evolutionary Algorithms: A Review", Evolutionary Algorithms in Engineering and Computer Science (pp. 113 - 133). Chichester: J. Wiley and Sons.