Regression Testing based on Genetic Algorithms

Esha Khanna
Assistant Professor
IT Department
D. A. V Institute of Management,
Faridabad, India

ABSTRACT
Regression testing re-executes the test cases to validate that changes made in software does not affects the correct functionality inherited from previous version. Due to limited time and resources, test cases are prioritized based on some criteria such that important test cases are executed within testing period. The work proposes a genetic algorithm based prioritization technique, which intelligently reorders the test cases on maximum fault detection rate. The work paves the way of genetic algorithms in regression testing.

General Terms
Algorithms, Reliability, Testing.

Keywords
Regression Testing, Test Case Prioritization, Genetic Algorithms.

1. INTRODUCTION
Efficient and thorough testing is required to build quality software. During maintenance phase, software may go through number of changes. A bug may get fixed or new functionality may be added. Regression testing re-executes the test cases to validate that changed modules have not interrupted the proper working of other modules [1]. It ensures that the inherited functionality from previous version has not been affected by the modification. Further, new test cases that focus on enhanced software functionality are added to the test suite. As a result, size of regression test suite grows and test cases become colossal.

Re-executing all the regression test cases is a costly process. Due to limited time, cost and resources only a part of test suite is executed. Selection of such test cases has to be based on some criterion such that quality of testing is not deteriorated. One such technique is test case prioritization. In test case prioritization, test cases are ordered in such a way that most important test cases are executed before others [2].

The work proposes genetic algorithm based intelligent technique to prioritize test cases for regression testing. Genetic algorithms are adaptive heuristic search algorithms that are based on Charles Darwin theory of the survival of the fittest [3]. Genetic Algorithms are used to solve optimization problem. The work proposes a technique that prioritizes subsequence of regression test suite in such a way, that its execution has a high rate of fault coverage when compared to rates of randomly prioritized test suite. The work reviews various techniques of test case prioritization.

The paper has been organized as follows. Section two discusses regression testing. Section three reviews various techniques of test case prioritization. Section four explains genetic algorithms. Section five presents the proposed framework and section six concludes.

2. REGRESSION TESTING
Software undergoes many changes during its maintenance phase. New features reflecting new customer requirements are added and some existing features are updated. Changes in the version of software, requires its retesting to confirm that new features have not affected the functioning of derived features. This is known as regression testing. Regression testing is selective retesting of the system or components to verify that modifications have not caused an unintended effects and the system or component still complies with its specified requirements [4]. Regression testing is required in the following scenarios [5].

- Software code is modified in accordance to changed customer requirements.
- Functionality of software is enhanced.
- Fixing of defects in previous versions.
- Removal of obsolete functionality from previous version.

Test cases of regression test suite are classified in three categories [6].

- Test cases that tests all software functionality.
- Test cases that focuses on the functionality that may be affected by the new changes.
- Test cases that are added to the test suite to test enhanced software functionality.

As new test cases are added the size of regression test suite grows rapidly. Regression is the most expensive phase of software testing and is required whenever the software functionality is updated. Regression testing may be done manually or can be automated with help of playback tools [6].

Re-executing all the test cases of regression test suite is a costly process. Due to restricted time and resources, subset of test cases is executed. Techniques of regression testing are selection, minimization and prioritization [7]. Test case selection technique selects and re-executes only a subset of test cases that focuses on changed module [7]. Test case minimization technique reduces the test case suite based on some criterion [7]. Test case prioritization technique reorders the test cases based on some criterion in such a way that most important test cases are executed within time and resources [8].

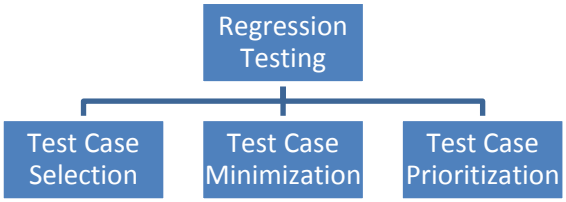

Fig 1: Techniques of Regression Testing

Test case selection and minimization techniques reduce the number of test cases of regression test suite, while test case prioritization does not alter the number of test cases. Test cases are assigned high, medium and low priorities on the basis of given criterion. Prioritization criteria may be based on test case history [9], coverage [10], fault severity [11], customer requirements [12] or costs [13]. The work proposes a technique which prioritizes the test cases based on maximum fault coverage by using genetic algorithms.

3. LITERATURE REVIEW

An extensive review has been carried out in order to find the gap in the existing literature. The review has been carried out on the guidelines of Kitchenham [14]. Some of the works has been summarized in this section.

Zing Li, et. al. presented five search algorithms for test case prioritization of regression suite [15]. Two metaheuristic search techniques and three greedy algorithm techniques were studied. Techniques included hill climbing, Genetic Algorithm, greedy, additional greedy and 2-optimal greedy algorithms. The work carried out an empirical study to compare 5 algorithms on 6 programs ranging from 374-11148 lines of code. Results showed that additional greedy and 2-optimal algorithm have best overall results. In one of the works, cuckoo search was used for test case selection and prioritization [16]. Cuckoo search is a single parameter optimization problem which is inspired by the obligate brood parasitism of cuckoo species. The work prioritized test cases on criteria of number of faults covered in minimum time.

Elbaum et. al. [17] proposed various test case prioritization techniques to improve the rate of fault detection.14 test casse prioritization techniques were classified into three groups i.e control techniques, statement level techniques and function level techniques. The work resulted that fine granuality techniques are better than course granularity techniques. Cagatay Catal discussed various critical issues for regression testing [18]. The work introduced ten best practices for test case prioritization and their role in successful software testing. Yoo and Harman classified test case prioritization techniques into 9 groups [7]. They were coverage based, distribution based human based, probabilistic approach, history based, requirement based, model based, cost aware approach and others.

Genetic algorithms were used by Konsaard and Ramingwong to carry out the task of test case prioritization [19]. The work reordered test cases according to criteria of maximum code coverage. The work compared performance of proposed technique with five other approaches on the basis of average percentage of condition covered and execution time. In the work by Harsh Bhasin and Manoj, a Genetic Algorithm based prioritization technique was proposed [3]. The technique used coupling number calculator to assign fitness value to test cases. Test cases prioritized using this technique resulted in high fault detection rate in comparison to others.

4. GENETIC ALGORITHM

Genetic algorithm (GA) is heuristic search algorithm inspired by Charles Darwin theory of natural evolution. GA is an intelligent search technique and is used to solve optimization problems [20]. It performs random searches through a pool of solution and aims to find best alternative according to some given criterion. Basic steps of a GA are as follows.

Step 1: Selection of initial population- The first step is to select a finite set of individuals (chromosomes) for a given problem. These individuals represent feasible solution. This leads to the generation of initial population.

Step 2: Evaluation of each individual- Each member of the population is evaluated according to some objective function. The objective function (also known as fitness function) assigns a fitness value to each individual member of population based on some given criterion.

Step 3: Generate new population- A new population of solution is generated by applying GA operators. These are reproduction, crossover and mutation [21].

- Reproduction operator- Chromosomes are selected from initial population and are entered to mating process by using reproduction operator. The selection methods are Roulette wheel selection, Rank selection, Tournament selection and Boltzmann selection [21].
- Crossover operator- crossover operator takes selected chromosomes as inputs and generates new offspring. It mimics the biological recombination of chromosomes. For two given chromosomes X={x1, x2…… xn} and Y={y1, y2……yn} and a crossover point say k, crossover operator generates two new chromosomes X1={x1……xk ,yk+1…….yn} and Y1={y1……yk , xk+1…….xn}. Crossover operator is classified as simple crossover, double crossover and N-point crossover [21].
- Mutation operator- Mutation operator changes the genetic makeup of offspring. It randomly replaces some of the bits of child chromosome. Different techniques of mutation operator are flipping, interchanging and reversing [21].

Step 4: Stopping criteria- Duplicates are removed from the new generation of chromosomes. If optimized solution is obtained stop, else go to step 2. Stopping conditions for GA are maximum generations, Elapsed time and unchanged fitness [21].

5. PROPOSED WORK

The work proposes use of genetic algorithm in regression testing. Test cases in regression test case suit become colossal. Re-executing all the test cases is not possible within limited time and resources. In such scenarios, test cases need to be prioritized. Test cases of regression test suite are intelligently reordered so that only important test cases are re-executed without affecting the overall quality of testing. This can be useful if the testing has to be stopped prematurely due to restricted cost and resources. The work prioritizes test cases using genetic algorithm on the basis of rate of fault detection. Fault detection depicts how quickly a test case can detect maximum number of faults within testing process assuming that all the faults are of equal severity. This helps in providing

a feedback in an early stage thereby assisting the developers to fix them on time. Steps of proposed work are as follows.

Test cases for each module are generated. The task can be automated using test generation tools. This forms the initial population of solution. Each test case is then assigned a fitness value on the basis of number of fault coverage criteria. The test case which covers more faults is assigned a greater fitness value than the test case which covers less number of faults. On the basis of fitness value, best test cases are selected to form a test suite in such a way that execution of suite leads to total fault coverage in a module. In order to create new generation of prioritized test cases genetic operators are applied. Crossover operator applied to test suits generates randomness in solution. In order to add variation in new generation, mutation operator is applied. Duplicate test cases are removed and prioritized test cases are obtained. The process is summarized in figure.

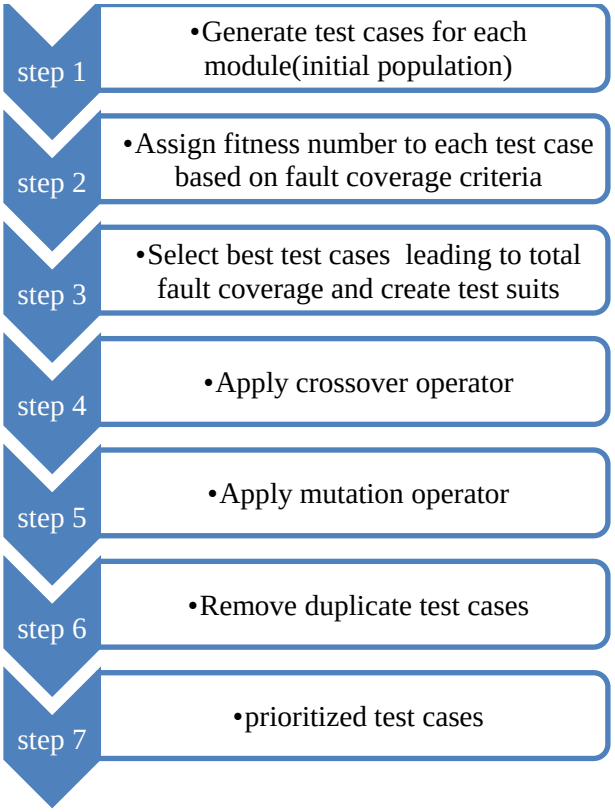

Fig 2: Steps of proposed work

6. CONCLUSION

Modification of software requires re-execution of all the test cases to validate that changes in one module have not modified correct functionality of others. Due to limited resources, all the test cases cannot be re-executed within the development period. Prioritization of test cases is required in order to maintain quality of regression testing. Test cases need to be prioritized in such a way that important test cases are executed before completion of software. This results in cost effective regression testing within limited time and resources. The work prioritizes test cases on the basis of high fault coverage rate. The paper presents use of genetic algorithm that optimizes the task of regression test case prioritization. The technique would be helpful to both researchers and practitioners if the testing has to be stopped prematurely due to lack of resources.

7. REFERENCES

[1] Chauhan, N. 2010. Software Testing principles and practices. Oxford University Press.

[2] Rothermel, G., et. al. 2001. Prioritizing Test Cases for Regression Testing. IEEE Transactions on Software Engineering. vol. 27, no. 10, pp. 929-948, DOI 10.1109/32.962562.

[3] Bhasin, H., Manoj. 2012. Regression Testing Using Coupling and Genetic Algorithms. International Journal of Computer Science and Information Technologies. Vol. 3(1).

[4] IEEE std. definition of Regression Testing.

[5] Rajal, J. S., Sharma, S. 2015. A Review on Various Techniques for Regression Testing and Test Case Prioritization. International Journal of Computer Applications. Volume 116, No. 16.

[6] Pressman, R.S. 2010. Software engineering: a practitioner's approach. McGraw-Hill Higher Education.

[7] Yoo, S., Harman, M. 2012. Regression testing minimization, selection and prioritization: a survey. Software Testing, Verification & Reliability. John Wiley and Sons Ltd. Volume 22 Issue 2, Pp. 67-120.

[8] Srivastava, P., R. 2008. Test Case Prioritization. Journal of Theoretical and Applied Information Technology.pp 178-181

[9] Engström, E., et. al. 2011. Improving Regression Testing Transparency and Efficiency with History-Based Prioritization -- An Industrial Case Study. Software Testing, Verification and Validation (ICST). IEEE Fourth International Conference on 21-25 March 2011 Page(s):367 - 376 Berlin,IEEE, DOI:10.1109/ICST.2011.27.

[10] David Leon and Andy Podgurski. 2003. A Comparison of Coverage-Based and Distribution-Based Techniques for Filtering and Prioritizing Test Cases. In Proceedings of the 14th International Symposium on Software Reliability Engineering (ISSRE '03). IEEE Computer Society, Washington, DC, USA, 442

[11] Sejun Kim and Jongmoon Baik. 2010. An Effective Fault Aware Test Case Prioritization by Incorporating a Fault Localization Technique. In Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '10). ACM, New York, NY, USA, Article 5, 10 pages

[12] R. Kavitha, V.R. Kavitha, N. Suresh Kumar. 2010. Requirement Based Test Case Prioritization. In Proceedings of International Conference on Communication Control and Computing Technologies, 826–829.

[13] Ramasamy, K., Mary, S.A., 2008.Incorporating varying requirement priorities and costs in test case prioritization for new and regression testing IEEE, Computing, Communication and Networking.

[14] Kitchenham, B.A. et. al. 2010. Systematic literature reviews in software engineering .A tertiary study, Information & Software Technology .INFSOF , vol. 52, no. 8, pp. 792-805, 2010

[15] Li. Z., et. al. 2007. Search Algorithms for Regression Test Case Prioritization. Software Engineering, IEEE transactions vol 33 issue 4.

[16] Nagar, R., et. al. 2015. Test case selection and prioritization using cuckoos search algorithm. International conference on Furistic Trends on Computational Analysis and Knowledge Management. IEEE.

[17] Elbaum, S., et. al. 2000. Prioritizing test cases for Regression testing. Presented in international symposium of software testing and analysis. 102-112.

[18] Catal, C. 2012. The Ten Best Practices for Test Case Prioritization. ICIST. Springer. pp 452-459.

[19] Konsaard, P., Ramingwong, L. 2015. Total coverage based regression test case prioritization using genetic algorithm. 12th International conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology. IEEE.

[20] Klir, G. J., Yaun, B. 1995. Fuzzy sets and Fuzzy logic theory and applications. Appendix B- Genetic Algorithms- An overview. Pearson Education Inc.

[21] Shivanandam, S. N., Deepa, S. N. 2012. Principles of Soft Computing. Second edition. Wiley India.