SOFTWARE TESTING STRATEGY APPROACH ON SOURCE CODE APPLYING CONDITIONAL COVERAGE METHOD

Jaya Srivastaval[1] and Twinkle Dwivedi[2]

[1]Department of Computer Science & Engineering, Shri Ramswaroop Memorial University,India
[2]Department of Computer Science & Engineering, Shri Ramswaroop Memorial University,India

ABSTRACT

Software testing is an important activity of the software development process. Software testing is most efforts consuming phase in software development. One would like to minimize the effort and maximize the number of faults detected and automated test case generation contributes to reduce cost and time effort. Hence test case generation may be treated as an optimization problem In this paper we have used genetic algorithm to optimize the test case that are generated applying conditional coverage on source code. Test case data is generated automatically using genetic algorithm are optimized and outperforms the test cases generated by random testing.

KEYWORDS

Source Code, Testing.

1. INTRODUCTION

Software testing is the procedure of executing a program or system with the intent of finding faults. Testing is a process of confirming that product is working according to the specification and satisfying the customer needs. Software testing provides a means to reduce errors, cut maintenance and overall software costs. Numerous software testing methodologies, tools, and techniques have emerged over the last few decades promising to enhance software quality. Software testing is important part in the software development life cycle. Two common approaches are white box testing and black box testing. There are different coverage measure for testability to the source code such as statement coverage, branch coverage and condition coverage. In the branch coverage we make sure that we execute every branch at least once For conditional branches, this means that, we execute the TRUE branch at least once and the FALSE branch at least once conditions for conditional branches can be compound boolean expressions a compound boolean expression consists of a combination of boolean terms combined with logical connectives AND, OR, and NOT Condition coverage. In this paper we propose a model for automatic and optimized test case generation. In our proposed method the initial test case generated using conditional coverage to cover all the paths then genetic algorithm is used for optimizing the test cases. This is an efficient approach of optimizing test case by using both genetic algorithm and conditional coverage.

1.1 Testing strategies

A test strategy is an outline that describes the testing approach of the software development cycle. A strategy for software testing integrates the design of software test cases into a well planned series of steps that result in successful development of the software[6][13]. Testing begins at component level and work outward towards the integration of entire computer based system. There are different types of testing strategies which are shown in Figure 1:

1. Unit testing- it concentrate on each component function of the software as implemented in the source code. Components are then assembled and integrated
2. Integration testing- it focus on the design and construction of the software system. It also focuses on input and output and how well the component fit together and works together.
3. Validation testing- requirements are validated against the constructed software. It provides final assurance that the software meets all functional and performance requirement.

4. System testing- In this the software and other system elements are testes as a whole that overall system function and performance is achieved. In this paper we used unit testing we take source code and focus on the each component function of the source code and internal structure of the program's source code.

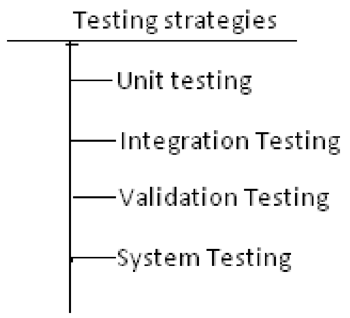

Figure1: The Testing Strategy

1.2 Coverage Criterion

The adequacy of testing is evaluated by the coverage measure that describes the degree to which a program's source code has been tested. Coverage is the extent that a structure has been exercised as a percentage of items being covered [15][18]. If coverage is not completed then more tests may be designed to test those items that were missed and therefore increase coverage. There are different types of coverage used in testing they are as follows

Statement coverage- all statements in the programs should be executed at least once.
Branch coverage- all branches in the program should be executed at least once.
Condition coverage- it executes the true or false outcome of each condition. It is closely related to the decision coverage but has better sensitivity to the control flow.
In this paper we apply conditional coverage by making the control flow graph of the source code so that we can detect the faults as much as possible.

2. PROBLEM IDENTIFICATION

The process of testing any software system is an enormous task which is time consuming and costly. Software testing is laborious and time-consuming work; it spends almost 50% of software system development resources . Generally, the goal of software testing is to design a set of minimal number of test cases such that it reveals as many faults as possible.

An automated software testing can significantly reduce the cost of developing software. Other benefits include: the test preparation can be done in advance, the test runs would be considerably fast, and the confidence of the testing result can be increased. Automated test case generation using genetic algorithm reduce the cost and effort and applying conditional coverage on source code reveals as many faults as possible by covering all the paths of the source code.

3. PROPOSED WORK

In this paper my motive is to optimize the test cases by making the control flow graph from the source code by covering all the paths taking conditional coverage and then generate the test cases randomly. Then genetic algorithm apply on the test cases of the source code and genetic algorithm optimizes the test cases. Original test cases are generated using random testing applying conditional coverage on source code after that test cases are refined using genetic algorithm for automatically test case generation that detect faults as much as possible. In this paper the model shows how the genetic algorithm optimizes the test cases that are generated randomly and detect the faults of the source code as much as possible.

We make a model that shows the overall structure of our proposed work. Figure 2 shows the proposed model.

In our model we start testing process by taking a source code. For testing process we have to generate test cases. In our methodology we make control flow graph for the source code and then apply the condition coverage on the control flow graph and test cases are generated randomly. These test cases are refined using genetic algorithm for producing a set of optimize test case that can detect all possible faults. The condition coverage executes each true and false outcome of each condition and the testing adequacy is evaluated by coverage criterion that how much percentage of source code is exercised by the coverage and condition coverage cover all the paths of the source code.

For producing the optimize test case we use genetic algorithm. Genetic algorithm starts with guesses and attempts to improve the guesses by evolution. A GA will typically have five parts: (1) a representation of a guess called a chromosome, (2) an initial pool of chromosomes, (3) a fitness function, (4) a selection function and (5) a crossover operator and a mutation operator. A chromosome can be a binary string or a more elaborate data structure. The initial pool of chromosomes can be randomly produced or manually created. The fitness function measures the suitability of a chromosome to meet a specified objective: for coverage based automatic test case generation, a chromosome is fitter if it corresponds to greater coverage. The selection function decides which chromosomes will participate in the evolution stage of the genetic algorithm made up by the crossover and mutation operators. The crossover operator exchanges genes from two chromosomes and creates two new chromosomes. The mutation operator changes a gene in a chromosome and creates one new chromosome.

A basic algorithm for a GA is as follows
The pseudo code for GA is:

```
Initialize (population)
Evaluate (population)
While (stopping condition not satisfied) do
{
Selection (population)
Crossover (population)
Mutate (population)
Evaluate (population)
}
```

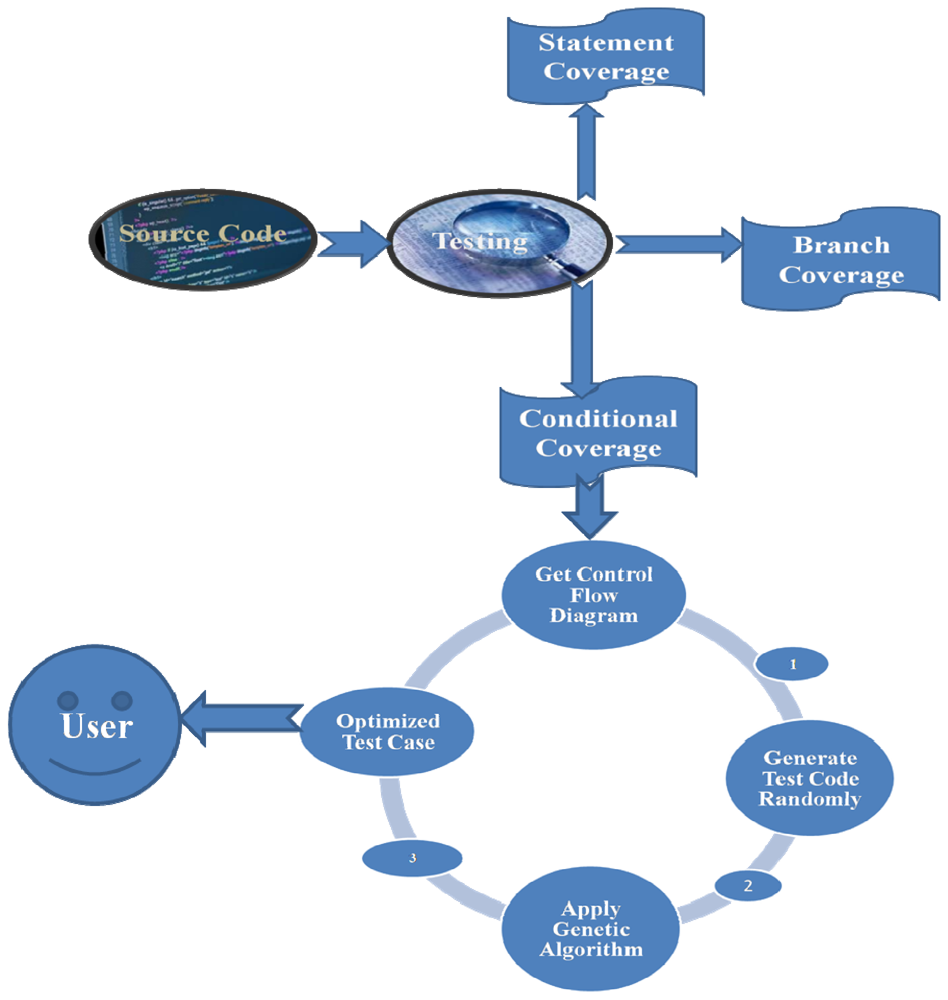

Figure 2: Overall Structure of Proposed Work

4. METHODOLOGY OR MODEL

The adequacy of testing is evaluated by the coverage measure that describes the degree to which a program's source code has been tested. The adequacy criteria include statement coverage, branch coverage and condition coverage. Condition coverage is apply on the control flow graph of source code. In control flow graph the nodes are statement and the edges represent the flow of control between statements.

In the branch coverage we make sure that we execute every branch at least once. For conditional branches, this means that, we execute the TRUE branch at least once and the FALSE branch at least once conditions for conditional branches can be compound boolean expression.

In our model we make control flow graph for the source code then we apply conditional coverage to cover all the paths of the control flow graph because condition coverage execute every edge of the graph at least once. We generate test case randomly using conditional coverage. Test cases are refined by the genetic algorithm for optimizing the test cases. New test cases are generated using genetic algorithm. Steps to execute genetic algorithm are as follows.

1. Genetic representation of the random test cases.
2. Fitness function to evaluate the test cases.
3. Select the best fit individuals for reproduction.
4. Breed new test cases through crossover and mutation operation.
5. Repeat this generation until the termination condition is satisfied.

Our algorithm works on control flow graph (CFG). CFG is a simple notation for the representation of control flow. An independent path is any path through the program that introduces at least one new set of processing statements or a new condition. When stated in terms of a flow graph an independent path must move along at least edge that has not been traversed before the path is defined. Condition coverage is applied on the control flow graph for producing test case randomly. Condition coverage covers all the paths of the source code so that it can detect maximum faults.

New test cases are generated automatically using genetic algorithm. The mutation and crossover operation produce new test cases. These test cases are optimized test cases because they detect all possible faults of the source code because test cases are generated covers all the path of program's source code so these test cases detect fault as much as possible. The steps to execute proposed method is as follows

4.1 Proposed Algorithm

1. Make the control flow graph of the source code.
2. Apply conditional coverage on the control flow graph
3. Generate initial test case randomly.
4. Generate a set of optimize test case using a genetic algorithm.

So in our methodology for producing optimize test case for testing process we make control flow graph for source code and generate test case randomly by applying conditional coverage on the control flow graph and then for producing optimize test case that can detect all the faults as much as possible we refined test case using genetic algorithm. Genetic algorithm produced a set of optimize test case that is used by the user. Here the user has authority to testing the source code by using test cases. Genetic algorithm has well defined steps such as initialization, selection, crossover and mutation. The crossover and mutation process produce new test cases that can detect maximum faults for the source code. So by using this methodology we can generate test case automatically that save effort, time and cost for testing process. The algorithm described above enhances the software testing strategies by saving effort and cost by generating test cases automatically.

5. RESULT & CONCLUSION

The model shows that test cases are generated randomly by applying conditional coverage and genetic algorithm produce a set of optimize test cases automatically that detect all the possible faults from the source code. Automatically test case generation of my model reduce the effort and cost of the testing process. In this paper we proposed a methodology for automatic test case generation of software test cases by focusing on condition coverage of source code. First we randomly generated initial test cases and refined them using a genetic algorithm. Genetic algorithm has well defined steps and mutation and crossover operator produce new optimize test cases.A set of optimized test case detect faults as much as possible from the source code. So automatically test case generation enhance the software testing strategies and also improve the software quality.

The overall result shows that this methodology is a promising approach for fully automatic test case generation for the testing technique that use condition coverage of the source code. To increase the efficiency and effectiveness, and thus to reduce the overall development cost for software based systems a systematic and automatic test case generator is required. Genetic algorithms search for relevant test cases in the input domain of the system under test and our methodology produce test case automatically using genetic algorithm. The application scope of genetic algorithm can go further that every technique of testing can be implemented using genetic algorithm.

ACKNOWLEDGMENT

We are thankful to the faculty of Computer Science and Engineering, Shri Ramswaroop Memorial University for the motivation and continuous support time to time.

REFERENCES

[1] D.J.Berndt, and A.Watkins , "Investigating the Performance of Genetic Algorithm-Based Software Test Case generation",In Proceedings of the Eighth IEEE International Symposium on High Assurance System Engineering (HASE'04), University of South Florida, pp. 261-262 March 25-26, 2004.

[2] D. P. Mohapatra," Automated Test Case Generation and Its Optimization for Path Testing Using Genetic Algorithm and Sampling ", 2009 IEEE.

[3] H. Haga," Automatic Test Case Generation based on Genetic Algorithm and Mutation Analysis", IEEE International Conference on Control System,Computing and Engineering,2012.

[4] R.Kumar," Automatic Test Suit generation with Genetic Algorithm", IJETCAS 13-178; 2013.

[5] I.Somerville, "Soft ware engineering," 7th Ed. Addison-Wesley.

[6] A. P. mathur,"Foundation of Software Testing", 1st edition Pearson Education 2008.

[7] N.Mansour, M. Salame," Data Generation for Path Testing", Software Quality Journal, ,Kluwer Academic Publishers.12, 121–136, 2004.

[8] P. R. Srivastava , "Generation of test data using Meta heuristic approach" IEEE TENCON , India available in IEEEXPLORE, 19-21 NOV 2008.

[9] J.Wegener, A.Baresel,, and H.Sthamer, "Suitability of Evolutionary Algorithms for Evolutionary Testing," In Proceedings of the 26th Annual International Computer Software and Applications Conference, Oxford, England, August 26-29, 2002.

[10] P. R.Srivastava1 and Tai-hoon Kim2, "Applicationof Genetic Algorithm in Software Testing", InternationalJournal of Software Engineering and ItsApplications Vol. 3,No.4, October 2009.

[11] C.Korel. "Automated software test data generation". IEEE Transactions on Software Engineering, 16(8),August 1990.

[12] B.F. Jones, H.-H. Sthamer and D.E. Eyres." Automatic structural testing using genetic algorithms", Software Engineering Journal, pages 299- 306, September, 1996.

[13] D.E.Goldberg, "Genetic Algorithms: in Search, Optimization & Machine Learning", Addison Wesley, MA. 1989.
[14] J.Horgan, ,S. London., and M.Lyu, "Achieving Software Quality with Testing Coverage Measures", IEEE Computer, Vol. 27 No.9 pp. 60-69, 1994.
[15] D.J.Berndt, J.Fisher, L.Johnson, J.Pinglikar, and A.Watkins,"Breeding Software Test Cases with Genetic Algorithms," In Proceedings of the Thirty-Sixth Hawaii International Conference on System Sciences (HICSS-36), Hawaii, January 2003.
[16] M.Last, S. Eyal1, and A. Kandel, "Effective Black-Box Testing with Genetic Algorithms," IBM conference.
[17] J.C.Lin, and P.L.Yeh, "Using Genetic Algorithms for Test Case Generation in Path Testing," In Proceedings of the 9th Asian Test Symposium (ATS'00). Taipei, Taiwan, December 4-6, 2000.
[18] A. baresel, H. sthamer and M. schmidt, "fitness function design to improve evolutionary structural testing," proceedings of the genetic and evolutionary computation conference, 2002.
[20] Dr V. Rajappa, A. Biradar, and S. Panda, "Efficient Software Test Case Generation Using Genetic Algorithm Based Graph Theory," First International Conference on Emerging Trends in Engineering and Technology, ICETET '08, pp.298-303, 2008.
[21] N.K. Gupta and Dr. M. K. Rohil,"Automatic Test Case GenerationUsing genetic Algorithm For Unit Testing of Object Oriented Software", First International conference on Emerging Trends in Engineering and Technology 2008.
[22] B. T.de Abreu, E.Martins, Fabiano, and L.de Sousa "Automatic test Data generation for path testing using a genetic Algorithm",new stochastic algorithm".2004.

AUTHORS

I Jaya Srivastava pursuing M.Tech in Computer Science & Engineering from Shri Ramswaroop University and done B.Tech in computer science & Engineering from United Institute Of Technology, Allahabad.

I Twinkle dwivedi pursuing M.Tech in Computer Science & Engineering from Shri Ramswaroop University and done B.Tech in information technology from Goel institution of technology and management lucknow.