International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 04 | Apr -2017 www.irjet.net p-ISSN: 2395-0072

Optimizing Software Testing using fuzzy logic in Aspect oriented programming

JYOTI[1], Susheela Hooda[2]

[1]*Student, Department of Computer Science, B.S.A.I.T.M, Faridabad, Haryana*
[2] *Assistant Professor, Department of Computer Science, B.S.A.I.T.M, Faridabad, Haryana*

Abstract - *Software testing is one of the most important activities in product development. For performing the task of software testing, test cases are to be designed. Thus, efficiency of software testing depends on designing proper test cases. Considering all possible combinations of values in a program is practically not possible due to various kinds of limitations. Thus exhaustive testing is impractical. If test cases are defined accurately on the basis of conditions, then testing becomes efficient and inexpensive. However, many a times redundant and useless test cases are developed. Redundant and useless test cases increase the effort and cost. Thus, software testing can be optimized by reducing the number of test cases without reducing the coverage. Various techniques for this have been proposed for this. This includes various Meta heuristic techniques as well. However, very less has been done in this field regarding Aspect Oriented Programming. Thus, this paper covers using fuzzy technique for reducing the number of test cases in aspect oriented programming. A difference in using this technique for AOP is presented. Also, the technique proposed not only significantly reduces the number of test cases using a special type of fuzzy clustering in AOP, it even tells the efficiency by which this is successful.*

Key Words: **Software quality assurance; fuzzy logic; meta-heuristic approach; Cyclomatic complexity; Aspect-oriented programming (AOP), AOP testing; Fuzzy c-means clustering; Test case redundancy.**

1. INTRODUCTION

A program consists of supporting functions and the main program's business logic. To isolate supporting functions from the main program's business logic, Aspect-oriented Programming (AOP) is used. To increase modularity, is the main aim of AOP. Each cross cutting concern must encapsulate at one place. AOP helps in achieving this as all implementations have same cross cutting expressions. This helps in achieving minimal or no code scattering.

Main aim is to provide sufficiently good solution to an optimization problem, for which a heuristic (partial search algorithm) is chosen that provides good solution for this. Metaheuristics is higher level heuristics. Finding by trial and error is called heuristics. And meta means higher level. Thus, metaheuristics are better than simple heuristics. Metaheuristics are useful for a variety of problems. Testing techniques are divided into manual and automation testing techniques. Test cases can be found manually or can be prioritized using meta-heuristic techniques like genetic algorithm and fuzzy logic. Till date various researches have been applied on object oriented programming. Not much has been covered in context of aspect-oriented programming.

The motivation of this paper is to propose an algorithm using fuzzy logic which helps to reduce the number of test cases by clustering the similar together. Also, after performing the clustering of the various test cases, it then finds out the best from a given cluster. It even calculates the efficiency of the same. Thus, this paper proposes this algorithm which is very beneficial for increasing the overall efficiency of the software lifecycle.

This paper is divided into 6 different sections. Section 1 introduces the need of AOP and testing and research area to be covered. Section 2 presents the literature survey . Section 3 presents the proposed methodology for reducing the test cases. Section 4 shows the implementation of the proposed algorithm using examples of a small as well as larger programs. Section 5 analyses the results and finds the comparison by calculating the efficiency. At last, conclusion and future prospects of this algorithm are covered in section 6.

2. LITERATURE SURVEY

To have a proper understanding of aspect oriented, current scenario of research in this field is to be identified. In order to analyse various existing testing techniques available for aspect-oriented software systems, available literature was extensively studied. Applied various search techniques from sources like digital libraries of IEEE, ACM, Springer Link etc. During the study, various journal, technical reports and conference papers were referred from these sources.

Following criteria are used for comparison: explaining meta-heuristic search algorithms including fuzzy logic, genetic algorithm. Work done in this direction even in OOP has been discussed.

For optimizing the test cases and reducing the effort, Meta heuristic techniques have to be considered. Heuristics means to find or to discover by trial and error. And Meta means higher level and metaheuristics generally perform better than simple heuristics. Meta-heuristic algorithms have the ability to obtain the optimal solution in a very large search space of candidate solutions. Meta-heuristics consists of applying Artificial intelligence like Genetic algorithm, Fuzzy logic to a

problem. Thus, now papers introducing Meta heuristic techniques are referred.

Meta-heuristic techniques are very much used in software testing. *Pandey et al* [2] proposed a technique called search based software testing. This paper gives an insight into the recent trends of the applications of search based techniques to generate Applications of meta-heuristic techniques in the field of software testing phases. Meta-heuristic techniques can also be applied in generation of test data [3]. The aim is to generate the optimum set of test data. Next meta-heuristic techniques are needed to generate search–based test data for branch coverage [4]. This paper introduces multi–objective branch. While studying meta-heuristics, some evolutionary test environment has been identified for automatic structural testing [5]. The environment has been developed that performs fully automatic test data generation for most structural test methods.

After getting knowledge of meta-heuristics, a basis of object oriented was reviewed to find the applications of meta-heuristics in object oriented programming. Nothing much has been identified in the field of meta-heuristics in aspect oriented. So, a general comparison of AOP vs OOP is also being referred in order to apply meta-heuristics in aspect oriented.

Dr. Suresh [6] presented a method of Software quality assurance for object-oriented systems using meta-heuristic search techniques. This paper tells how much capable meta-heuristic search techniques are in software fault classification for Apache Integration Framework. Also, regression testing of object oriented systems [7] has been reviewed. A hybrid technique is formed which identifies the changes that are not visible in design models. Finally, comparison of AOP with OOP is done and identifying whether AOP is harder or easier than OOP [1]. An incremental testing process is proposed in this paper, which tests crosscutting functionalities as aspects in successive steps. Model based testing approach for object oriented systems is being considered by *Arilo et al* [15]. By identifying four hundred and six papers, this paper narrowed down it to seventy-eight papers for reference.

On identifying the meta-heuristic techniques and their usage in object-oriented programs, the problem was identified that not much has been done for implementation of meta-heuristics in aspect oriented programming. A proper review regarding the same has also never been done. So, next thing is identifying the papers in this field. The strategy goes by finding papers written for genetic algorithm, fuzzy algorithm or their combination. Fifth, let's identify the work done in the field of using genetic algorithm and mutation testing in aspect oriented programs.

Genetic algorithm has been applied to increase the efficiency of a Data Flow-based Test Data Generation Approach [8]. How effective it is to use a genetic algorithm for the same has been identified here. Another major application of genetic programming is in effort estimation [9]. *Ferrucci et al* [9] found the impact different fitness functions can have on effort estimation. To consider the indirect impact of aspects, *Delamare et al* [14], proposed an approach. When many methods are indirectly impacted by aspects, this approach can reduce the testing efforts by relying on genetic algorithm.

Fuzzy logic has been used in aspect oriented for proposing a model on coupling measures [10]. In object oriented and module oriented systems, quality metrics are used for quality features but very less has been done for aspect-oriented systems. Various factors like Number of dependencies, responsibility and Instability have been related with coupling for making the comparison. Another application of fuzzy logic approach has been identified in measuring the complexity of a generic aspect-oriented system [11]. A generalized framework for finding the complexity of AO systems has been defined. That it takes into account three AOP languages, which are AspectJ, CaesarJ and Hyper/J.

Fuzzy clustering is very much useful in various applications. One of them is using it in test-case prioritization. *Nida et al* [12] identified an approach to rank the test cases as per their preference degrees. Software testing optimization can also be achieved by fuzzy clustering [13]. This is very much useful in reducing the time spent in testing.

3. PROPOSED METHODOLOGY

On analysing the various works done, it has been identified that very less has been done using meta-heuristics with AOP in software testing. Thus, this paper proposes a fuzzy technique which helps to reduce the number of test cases for better efficiency and optimizations in the testing process. The algorithm defined as:

1. Construct activity diagram.
2. Conversion of activity diagram to Control flow graph.
3. Calculation of Cyclomatic Complexity.
4. Calculation of independent paths.
5. Calculate number of nodes of each path. This will act as cost of each path.
6. Apply fuzzy -c algorithm to form clusters.
7. Select best test case from given cluster. This is done by choosing a test case which is most closely related to other test cases i.e. which has the least sum of difference with other test cases.
8. Calculate efficiency of two things.
    a. First, calculate efficiency of the whole algorithm by comparing the number of original test cases vs new test cases.
    b. Second, how efficient and fault free this algorithm is , by computing the efficiency with which given output will cover all the cases. This is computed by calculating difference of the chosen test cases with other given in the cluster. Less the difference, more efficient it will be.

Covering each and every step in detail with the help of an example is further described in this paper. An example with 5 test cases has been discussed which properly explains the use case of each step.

3. EXAMPLE OF PROPOSED ALGORITHM

EXAMPLE 1: BANKING EXAMPLE

Problem statement: Consider a banking system which considers security as an important feature. Thus, authentication, authorization etc. to be considered as aspects. Withdrawal system is to be considered. Draw activity diagram for the same and find test cases manually, and then optimize it using fuzzy approach.

Solution: Let's apply the above algorithm step by step to find out how it works.

Step 1: Construct activity diagram. Activity diagram for banking withdrawal system is given which shows the aspect nodes, decision nodes and aspect nodes.

Sequential nodes: 1, 4, 5, 6,8,9,17,18

Decision nodes: 3,10,12,14

Aspect nodes: 2,7,11,13,15,16

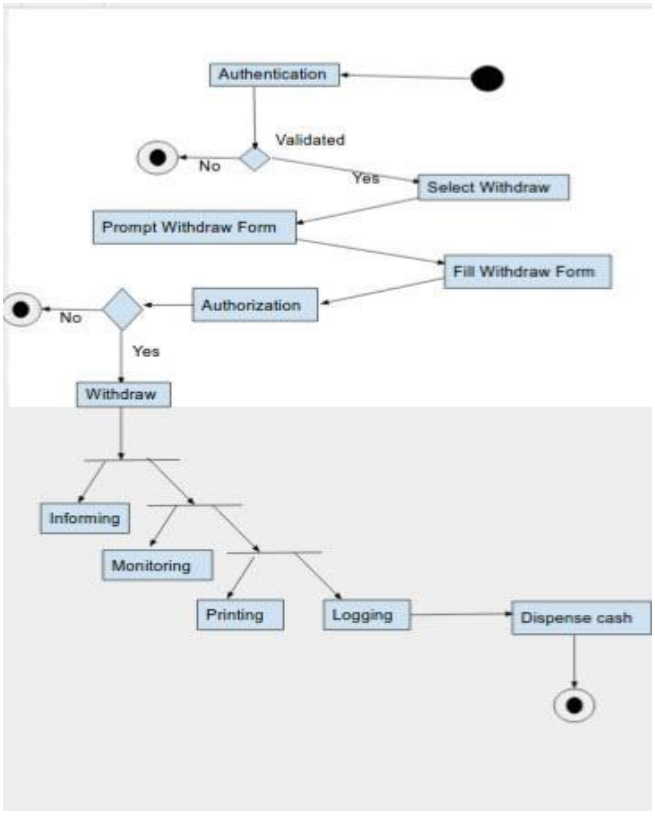

Figure 1: Banking activity diagram

Step 2: Conversion of activity diagram to Control flow graph.

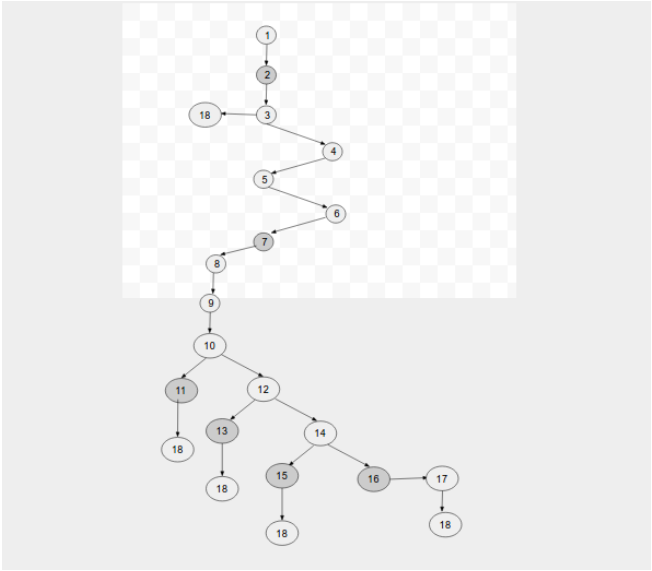

Figure 2: Banking CFG

Step 3: Calculation of Cyclomatic Complexity.
Cyclomatic complexity: Number of regions + 1
: 5 + 1 = 6.

Step 4: Calculation of independent paths.
TC 1: 1 -> 2 -> 3->18
TC 2: 1 ->2 ->3 ->4 ->5 ->6 ->7 ->8->18
TC 3: 1 ->2 ->3 ->4 ->5 ->6 ->7 ->8->9 ->10->11->18
TC 4: 1 ->2 ->3 ->4 ->5 ->6 ->7 ->8->9 ->10->12->13-> 18
TC 5: 1 ->2 ->3 ->4 ->5 ->6 ->7 ->8->9 ->10->12->14-> 15-> 18
TC 6: 1 ->2 ->3 ->4 ->5 ->6 ->7 ->8->9 ->10->12->14-> 16-> 17-> 18

Step 5: Calculate cost of each path by calculating the number of nodes in each path.
TC 1: 1 -> 2 -> 3->18 Cost : 4
TC 2: 1 ->2 ->3 ->4 ->5 ->6 ->7 ->8->18 Cost : 9
TC 3: 1 ->2 ->3 ->4 ->5 ->6 ->7 ->8->9 ->10->11->18 Cost : 12
TC 4: 1 ->2 ->3 ->4 ->5 ->6 ->7 ->8->9 ->10->12->13-> 18 Cost : 13
TC 5: 1 ->2 ->3 ->4 ->5 ->6 ->7 ->8->9 ->10->12->14-> 15-> 18 Cost : 14
TC 6: 1 ->2 ->3 ->4 ->5 ->6 ->7 ->8->9 ->10->12->14-> 16-> 17-> 18 Cost : 15

Step 6: Apply fuzzy -c algorithm to form clusters.
#Cluster 1:
4.000000 9.000000
4.000000 9.000000
12.000000 12.000000
#Cluster 2:
14.000000 15.000000
13.000000 13.000000
14.000000 15.000000

Step 7: Select best test case from given cluster.
From cluster 1: output is: 9
From cluster 2: output is: 34
Evaluating for cluster 1: values are 4,9,12.
For 4: (9-4) + (12-4) = 13
For 9: (9-4) + (12 - 9) = 8
For 12: (12-9) + (12-4) = 11
Least value is of 9.
Evaluating for cluster 2: values are 13, 14, 15
For 13: (14-13) + (15-13) = 3
For 14: (14-13) + (15 - 14) = 2
For 15: (15-14) + (15-13) = 3
Least value is of 14.
Step 8: Calculate efficiency.
First: (1- Number of reduced test cases/Number of original test cases) * 100 = (1-2/5) * 100 = 60%
Second:
Cluster 1: (1-least difference/ (sum of cost of test cases in a cluster)) * 100 = (1-8/ (4+9+12))*100 = 68%
Cluster 2: (1-least difference/ (sum of cost of test cases in a cluster)) * 100 = (1-2/ (13+14+15))*100 = 95.2%

5. CONCLUSION

In this paper, a proposed algorithm for reducing the number of test cases keeping AOP in mind has been discussed. The algorithm gives the best test case keeping everything in mind. Also, it tells with what efficiency it tells that the given test case is the best. On looking at the efficiency, we come to know to either run it or not.

Explanation through example covers each and every step of the algorithm for easy understanding. This algorithm can easily be applied on any example given activity diagram and further work would be done.

This algorithm would be one of the most important step in exploring fuzzy logic with AOP. Any researcher can gain enough information from this using this algorithm. Also, it marks the beginning of further researches in this field.

REFERENCES

[1] Mariano Ceccato, Paolo Tonella and Filippo Ricca , Is AOP code easier or harder to test than OOP code?.2015.

[2] Abhishek Pandey, Dr. Soumya Banerjee and Dr. G. Sahoo, Applications of Meta Heuristic Search Algorithms in Software Testing. 2014.

[3] P. R. Srivastava, V. Ramachandran, M. Kumar, G. Talukder, V. Tiwari, and P. Sharma, "Generation of test data using meta heuristic approach," in TENCON 2008 - 2008 IEEE Region 10 Conference, 2008.

[4] K. Lakhotia, M. Harman, and P. McMinn, "A multi-objective approach to search-based test data generation," in Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO '07, 2007.

[5] J. Wegener, A. Baresel, and H. Sthamer, "Evolutionary test environment for automatic structural testing," Information and Software Technology, vol. 43, no. 14, pp. 841–854, 2001.

[6] Y. Suresh, "Software quality assurance for object-oriented systems using meta-heuristic search techniques," in 2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2015.

[7] Pierre-Luc Vincent, Linda Badri and Mourad Badri, Regression Testing of Object-Oriented Software.2013.

[8] M. Mahajan, S. Kumar, and R. Porwal, "Applying genetic algorithm to increase the efficiency of a data flow-based test data generation approach," ACM SIGSOFT Software Engineering Notes, vol. 37, no. 5, p. 1, 2012.

[9] F. Ferrucci, C. Gravino, R. Oliveto, and F. Sarro, "Genetic Programming for Effort Estimation: An Analysis of the Impact of Different Fitness Functions," in 2nd International Symposium on Search Based Software Engineering, 2010.

[10] F. Chishti and A. Singhal, "'Proposed model on coupling measures in aspect oriented software development using fuzzy logic,'" in 2016 2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Fall), 2016.

[11] R. Kumar, P. S. Grover, and A. Kumar, "A Fuzzy Logic Approach to Measure Complexity of Generic Aspect-Oriented Systems," The Journal of Object Technology, vol. 9, no. 3, p. 59, 2010.

[12] N. Gökçe, F. BELLİ, M. EMİNLİ, and B. T. DİNÇER, "Model-based test case prioritization using cluster analysis: a soft-computing approach," TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES, vol. 23, pp. 623–640, 2015.

[13] G. Kumar and P. K. Bhatia, "Software testing optimization through test suite reduction using fuzzy clustering," CSI Transactions on ICT, vol. 1, no. 3, pp. 253–260, 2013.

[14] R. Delamare and N. A. Kraft, "A Genetic Algorithm for Computing Class Integration Test Orders for Aspect-Oriented Systems," in 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, 2012.

[15] A. C. Dias Neto, R. Subramanyan, M. Vieira, and G. H. Travassos, "A survey on model-based testing approaches," in Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007 - WEASELTech '07, 2007.