International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 21 (2017) pp. 11252-11260

A Novel Approach for Testing an Aspect Oriented Software System Using Prioritized-Genetic Algorithm (P-GA)

Dr. Sandeep Dalal
Assistant Professor, Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak-124001, Haryana, India.

Orcid Id: 0000-0002-2924-3474

Susheela Hooda
Research Scholar, Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak-124001, Haryana, India.

Orcid Id: 0000-0003-0911-9772

Abstract

Nowadays, Unified Modeling Language (UML) has been extensively used in the area of software testing. Aspect Oriented Programming (AOP) paradigm is relatively a new field of software development. Testing an AOP is still in its infancy stage. In addition, dealing with aspects in AOP is an intractable problem. Furthermore, there is no work based on optimization for test data generation. In this paper, aspect integrated UML activity diagram has been used for generation of optimal and minimal test case scenarios by applying genetic algorithm (GA) in order to obtain full aspectual coverage criteria. Moreover, this paper also discusses the comparison of P-GA (proposed approach) with other contemporary testing techniques that demonstrate the effectiveness of the proposed approach.

Keywords:-Aspect-Oriented Program (AOP), Genetic Algorithm(GA), Aspect-Oriented Software Testing, Unified Modeling Language(UML),UML Activity diagram, Aspectual branch coverage criteria.

INTRODUCTION

Aspect-Oriented Programming (AOP) is relatively a new programming paradigm in which user's problem is divided into two concerns namely primary concern and cross cutting concern. The primary concern is the main problem for which we develop the software whereas crosscutting concerns are not directly related to primary concern but they are needed to the proper execution of the software.

Testing an AOP is still in its infancy stage. This is the well-thought area of research. Many of the researchers have been working in this area. But still, one needs to concentrate on optimizing the testing strategies for AOP. For designing the quality software, testing strategies play an important role because good test paths lead to the quality software. Testing could be performed manually or automatically. Due to time, cost and effort manual testing is not feasible. Therefore, one needs to focus on automated testing process. But there is a misconception among professionals that automated testing is a replacement of manual testing. But entirely, it is not true, automated testing tools are just merely a part of the solution rather than provide a complete solution [1], [2], [3],[4].

These days, software testing using UML diagrams is gaining more popularity than other available testing techniques such as specification based testing and code based testing [5]. Model based testing is performed at an early stage of the software development life cycle (SDLC). Therefore, errors can be detected earlier and not let to allow carrying in later stages of software development process. In addition, use of code to test a software is a complex and tedious task. Various UML diagrams such as state model, activity diagram, collaboration diagram and sequence/use case diagram are available to depict the static and dynamic behavior of the software [6][7]. UML activity diagram depicts the external behavior of the software without providing the internals details. Moreover, UML Activity diagram helps to better understand the system behavior and find test information easily as compared to code based testing [8][9][10].

In spite of upswing the curiosity in Aspect-Oriented Programming (AOP) paradigm, still, there is a lack of work in AOP testing. Furthermore, there is no work based on optimizing the testing process in AOP. In this paper, an approach has been introduced to minimize/reduce the test sequences using Genetic Algorithm (GA) in order to obtain the aspectual branch coverage criteria. An aspectual branch coverage criteria cover all aspects within the AOP source code [11]. Aspectual branch coverage metric [11] has also been used to measure the coverage of aspectual behavior of the program.

GA is one such general purpose search algorithm which is based on the principle of natural evolution and used for best-fittest individual selection [12]. GA is a smart and an effective method which has been applied in large, complex and real life problems from the last few years [13].GA solves optimization problems by manipulating the initial population.GA has been

used in model based testing and regression testing for object oriented programming [15].GA has also been used in integration testing for aspect-oriented programming [16].

In recent past years, software testers have been shown more interest in metaheuristic techniques for optimizing the testing procedure in procedural and object oriented programming paradigms [5,8,9]. Not a bit work has been brought to bear in AOP paradigm.

Due to lack of enough work leaves many unanswered questions such as how perfectly these techniques can be suitable and what differences can be perceived while bringing into play these techniques on AOP. The proposed approach (P-GA) has been addressing these questions and caters the first optimized approach to generate the test case scenarios derived from aspect integrated UML activity diagram.

The contributions of this paper are:

1. This paper presents the first application of metaheuristic testing technique to optimize the number of test case sequences based upon Unified Modeling Language (UML) and Genetic algorithm for AOP programming paradigm.
2. This paper presents the results that yield proofs to support the claim that metaheuristic testing technique for AOP is cost effective.
3. This paper deals with efficient reduction of a number of test cases.
4. This paper also focuses on aspectual branch coverage (i.e. covered branches inside aspect code) criteria rather than all branches. The obtained results provide evidence that we can reduce test effort for achieving full or partial aspectual branch coverage criteria.

Although the proposed approach can be generally applied to test object-oriented programs [9] by focusing on selective elements, it is the first approach that is to be practiced to test aspect oriented programs by focusing on aspectual branches in a software system. This paper is divided into 6 sections. Section 2 describes the basic structure of GA. Section 3 describes the proposed approach and methodology and section 4 describes the experimental evaluation results of the proposed approach. Section 5 presents the comparative results of P-GA with random test data generation and manual test data generation (without GA) as a sanity check (to check the validity of applicability) and provide evidence of the effectiveness of P-GA(proposed approach). Section 6 concludes the paper and provides the direction of the future plan.

OVERVIEW OF GENETIC ALGORITHM

GA is an optimization and machine learning algorithm and has been effectively used to find the optimal solutions in software testing. GA solves optimization problems by using the principles of natural evolution [9].

GA operates in the following manner:-

1. Solution domain of optimization problems is represented by chromosome (genome).
2. Evaluate the solution domain by formulating the fitness function.
3. Apply the operations (selection, crossover, and mutation) to generate the new population.

The pseudo code of a basic algorithm for Genetic Algorithm is as follows [8] [9]:-

```
Initial (Population)
Evaluate (population)
While (Stopping condition not meet)
{
        Selection (population)
        Crossover (population)
        Mutation (population)
        Evaluate (population)
}
```

Three most important operations that are used to produce an efficient solution for given problem.

1) Selection

2) Crossover

3) Mutation

Selection:-GA used various selection techniques to select population such as tournament selection, rank selection, Boltzman selection etc.

Crossover: - This operation is used to generate next generation population from the earlier generated population by following procedure [9]:-

(1) Randomly select two individuals from the parent generation.

(2) Select a position of the gene as a crossover point that divided individual into two parts.

(3) Exchange first part of both genes corresponding to the selected individuals.

(4) Add the two resulted individuals to the next generation.

Mutation:-Mutation changes chromosome bit to bring new peculiarity in the population.

PROPOSED APPROACH AND METHODOLOGY

This section describes a P-GA approach for prioritizing the test case scenarios which are derived from aspect-integrated UML activity diagram. The purpose of the P-GA is trying to cover the aspectual branch at least once. The main constructs used in aspect integrated UML activity diagram are shown in Fig.1. UML (Unified Modeling Language) is used to visualize high-level view of the system which helps for the nontechnical person to understand the workflow of the system. UML diagram is used to describe structural, behavior and implementation view of an aspect oriented system [10][11].UML diagram consists of many constructs such as start, decision, activity, aspect, concurrent and stop node.

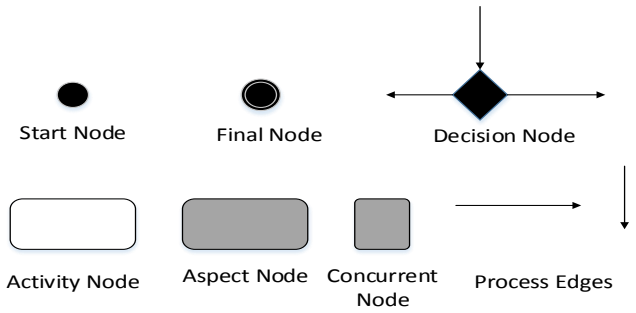

Figure 1: Basic Constructs of Aspect Integrated UML Activity Diagram

In above Fig.1, there are a different kind of nodes such as (Start, Final, Decision, Activity, Concurrent and Aspect node) in which aspect nodes are used to represent the crosscutting concerns and other nodes are used to represent the primary concern in AOP. Therefore, Aspect nodes are more important in AOP as they impact more in the behavior of the source code of AOP rather than other nodes [12] In this work, UML activity diagram is used to present the work flow for an AOP application of transfer scenario of a banking system.

P-GA (proposed approach) is a meta-heuristic technique based upon a Genetic algorithm and UML activity diagram. Unique Test case selection criteria are being used in P-GA (proposed approach) described in pseudo code which has been shown in fig.2 and fig.3. In P-GA (proposed approach), aspectual branch coverage (the number of covered aspectual branches divided by a total number of aspectual branches) criteria have been used to measure how well aspects are tested in an aspect-oriented program [13].

Proposed Methodology

This subsection describes the steps to automate the test data generation using P-GA (proposed approach). The following steps have to be taken:-

1. Transform an aspect integrated UML activity diagram in to Control Flow Graph (CFG) where each node represents activities (initial node, decision node, aspect node, concurrent node and merge node) and edges depict the flow of activities.

```
//algorithm for calculating IF and Weight assignment to each node
Algorithm Calculate IF and Weight _of _each _node ()
  {
    For ( i=0;i<n; i ++)
       IFi=fan-in*fan-out;
    {
      if (node type=”Normal Node”)
     weight =1;
     Elseif (node type=”Decision Node”)
      Weight=2;
    elseif (node type=”Concurrent Node”)
      Weight=3;
     Else
         (node type=”Aspect Node”)
          Weight=4;
   }}
}
```

Figure 2: Algorithm of Weight assignment and IF

Algorithm P- GA ()
//Proposed algorithm for test case generation from aspect integrated UML activity diagram.

Step 1: Convert the activity diagram into Control Flow Graph.

Step 2: Store the information into adjacency matrix.

Step 3: Display network of all nodes and neighbor.

Step 4: Generate randomly population (1-10) by ranking the fitness of chromosome.

Step 5: Call Calculate IF and weight of each_ node ()

Step 6: Calculate the prioritized fitness value (PFV):-

$$PFV_i = \sum_{i=1}^{n}(IF_i + W_i)$$ //prioritized fitness value

Step 7: Apply crossover and mutation// crossover=0. 40 , mutation rate=0.05

Step 8: Recalculate PFV of each test path.

Step 9: Repeat step 4 to 8 until an optimal solution is found.

Step 10: Perform Sorting on generated test path according to fitness value and select test path having highest fitness value.

Figure 3: Algorithm for prioritized test case scenario

2. Generate independent paths from CFG using cyclomatic complexity metric. (cyclomatic complexity metric is used to identify independent paths from graph).These paths will be used as the testing path. Each independent path contains at least one new node from other.
3. Decision nodes in CFG are used to represent the independent test paths in the form of 0 or 1.Each decision node has two branches. Therefore, we assign 1 or 0 to these branches (1 means true and 0 means false).Test case scenario's bit representation will consist of a number of bits equal to decision nodes in CFG.
4. Apply algorithm of Weight assignment and IF as shown in Fig.2 to calculate the fitness value of each node.
5. Then, apply the Prioritized test case scenario algorithm (P-GA) as shown in Fig.3 on generated independent test case scenarios to calculate the prioritized fitness value of each test case scenario.

Test cases are prioritized according to fitness value of each test case scenario. Higher the fitness value means the most prioritized test case scenario. The prioritized test case scenario will fulfill the aspectual branch coverage criteria (covered aspectual branches inside aspect code).

EXPERIMENTAL EVALUATION OF P-GA TESTING TECHNIQUE

The proposed P-GA approach has been applied on Transfer Scenario of Banking System [17]. Three parameters have been calculated for experimental evaluation:

1. Average Prioritized Fitness Value(APFV)
2. Effectiveness of the test suit minimization
3. Aspectual Branch Coverage metric

The notion for APFV calculation is:

$$APFV = \frac{\text{Sum of PFV of all selected test cases}}{\text{Total number of selected test cases}} \quad \text{--------- (1)}$$

Average Prioritized Fitness Value (APFV) has been calculated using above equation 1. A comparatively higher APFV means a better prioritization technique.

S. Yoo et. al.[18] developed a metric to measure the effectiveness of the test suit minimization.

The notion for calculating effectiveness is:

$$\left(1 - \frac{\text{No.of test case in the reduced test suit}}{No.of\ test\ cases\ in\ the\ original\ test\ suit}\right) * 100\% \quad \text{------ (2)}$$

M. Harman [13] used an aspectual branch coverage criteria which measured the percentage of covered aspectual branches among all aspectual branches.

Notion for calculating number of test case scenarios which overs aspectual branches:-

$$\left(\frac{\text{No. of test case scenarios which covered full aspectual branches}}{Total\ no.of\ test\ case\ scenarios}\right) * 100\% \quad \text{--------- (3)}$$

This paper presents the results by using above three parameters that yield proof to support the claim that P-GA testing technique for AOP is effective.

Transfer Scenario of a Banking System

The proposed approach has been applied on well- known problem of transfer scenario in a banking system [17]. Fig. 4 shows an Aspect integrated UML activity model of Transfer scenario of a banking system. In Fig.4, there are three aspect nodes: Authorization, Logging, and Authentication nodes are accompanied by primary features of Transfer scenario. Aspectual nodes are represented by a dark gray rounded rectangle, concurrent nodes are represented by the light gray rounded rectangle and therefore, they can be easily distinguished from Primary concerns. Priority is assigned to each node of UML activity diagram of transfer scenario of banking system according to weight assignment algorithm as shown in above Fig.2. The aspect integrated UML Activity diagram of transfer scenario of the banking system (Fig. 4) is converted into control flow graph (CFG) as shown in Fig.5. Node numbers are assigned to each node of CFG corresponding to Fig.4 UML activity diagram of transfer scenario of a banking system. In Fig.5, aspect nodes are represented by the dark gray circle and concurrent nodes have also been taken on account of concurrency between nodes and are also represented by the light gray circle.

The Steps are described below:

- First Construct the Aspect Integrated UML Activity Model of Transfer scenario of the banking system as shown in below Fig. 4 and converts it into Control flow graph (CFG) as represented in Fig.5.

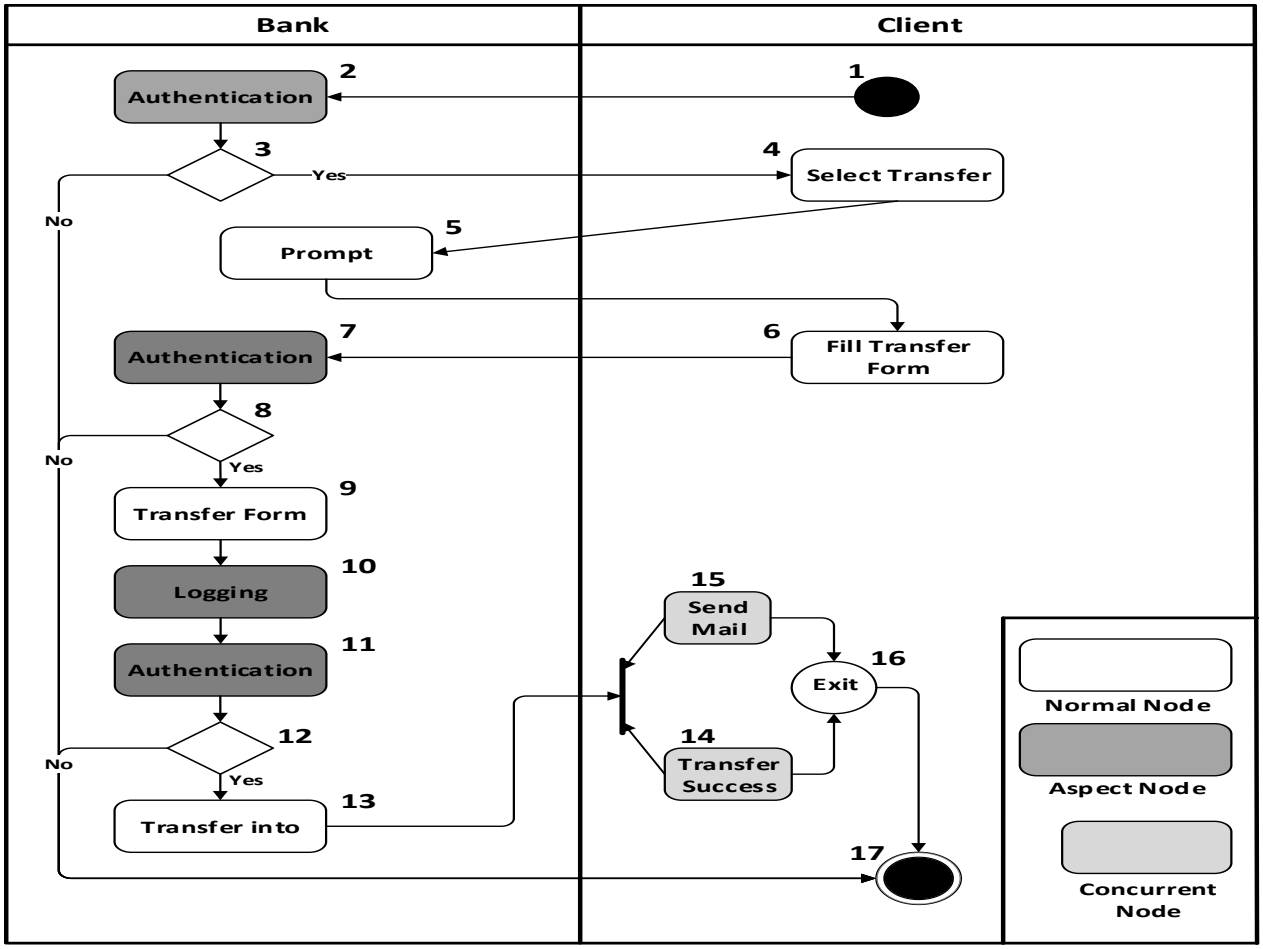

Figure 4: Aspect integrated UML Activity model of Transfer Scenario of Banking System

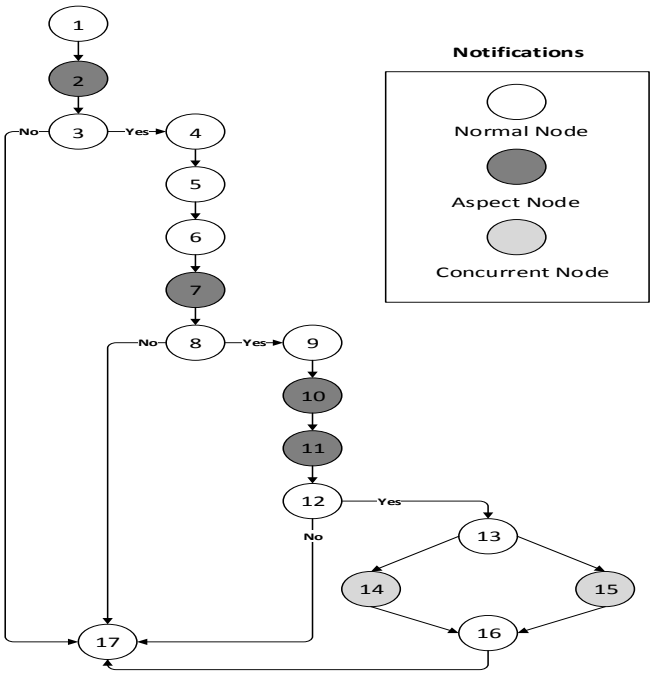

Figure 5: CFG of Aspect Integrated Activity Model of Transfer Scenario of Banking System

- Calculate the Cyclomatic complexity of given CFG as shown in Fig.5 using below equation to identify the independent paths from CFG.

Cyclomatic Complexity=E-N+2 ---------- (5)

Where E= Number of Edges

N=Number of Nodes

Therefore, Cyclomatic Complexity of given CFG in Fig. 5 will be 5 by using equation 5. These five independent paths will be used as the testing path. Each independent path must contain at least one new node from other independent paths

Hence, five independent paths which are generated from Fig. 5 with non-GA as shown below:-

1->2->3->17

1->2->3->4->5->6->7->8->17

1->2->3->4->5->6->7->8->9->10->11->12->17

1->2->3->4->5->6->7->8->9->10->11->12->13->14->16->17

1->2->3->4->5->6->7->8->9->10->11->12->13->15->16->17

- There are three decision nodes (3, 8 and 12) in Fig.5. Each decision node has two branches i.e. true and false whereas true is represented by 1 bit and false is represented by 0 bit. Therefore, three bits (three decision nodes: 3, 8, 12) have been used to represent the test case scenarios in bit form. Let us consider test path is as 1->2->3->17, it contains only one decision node i.e. 3.When we traverse above test path, it includes the only false branch of node 3 and other decision nodes are not covered during traversing. Therefore, above test path will be represented by 000.Bit representation of each test path has been shown below in Table 1.

Table 1: Bit representation of each test path

TCSs	TCSs	Chromosome
TCS1	**1->2->3->17**	**000**
TCS2	**1->2->3->4->5->6->7->8->17**	**100**
TCS3	**1->2->3->4->5->6->7->8->9->10->11->12->17**	**110**
TCS4	**1->2->3->4->5->6->7->8->9->10->11->12->13->14->16->17**	**111**
TCS5	**1->2->3->4->5->6->7->8->9->10->11->12->13->15->16->17**	**111**

- Calculate the IF, weight and fitness value of each node in Table 2 using Fig.2.

Table 2: Computation of Fitness value of each node

Node Number	Information Flow (IF)	Weight of each node	Fitness value of each node
1	0	1	1
2	1	4	5
3	2	2	4
4	1	1	2
5	1	1	2
6	1	1	2
7	1	4	5
8	2	2	4
9	1	1	2
10	1	4	5
11	1	4	5
12	2	2	4
13	2	1	2
14	1	3	4
15	1	3	4
16	2	1	3
17	0	1	1

- Calculate the prioritized fitness value (PFV) of each test case scenario of Table 1 using of Fig.3. Prioritized fitness value (PFV) of each test case scenario has been shown in Table 3.

TCS1= (0+1) + (1+4) + (2+2) + (0+1) =11

TCS2=(0+1)+(1+4)+(2+2)+(1+1)+(1+1)+(1+1)+(1+4)+(2+2) + (0+1) =26

TCS3=(0+1)+(1+4)+(2+2)+(1+1)+(1+1)+(1+1)+(1+4)+(2+2) +(1+1)+(1+4)+(1+4)+(2+2)+(0+1)=42

TCS4= (0+1)+(1+4)+(2+2)+(1+1)+(1+1)+(1+1)+(1+4)+(2+2)

(1+1)+(1+4)+(1+4)+(2+2)+(2+3)+(1+1)+(2+1)+(0+1)=52

TCS5= (0+1)+(1+4)+(2+2)+(1+1)+(1+1)+(1+1)+(1+4)+(2+2) +(1+1)+(1+4)+(1+4)+(2+2)+(2+3)+(1+1)+(2+1)+(0+1)=52

Table 3: Independent paths with their fitness value

TCSs	TCSs	PFVi
TCS1	**1->2->3->17**	**11**
TCS2	**1->2->3->4->5->6->7->8->17**	**26**
TCS3	**1->2->3->4->5->6->7->8->9->10->11->12->17**	**42**
TCS4	**1->2->3->4->5->6->7->8->9->10->11->12->13->14->16->17**	**52**
TCS5	**1->2->3->4->5->6->7->8->9->10->11->12->13->15->16->17**	**52**

- Now generate the test case scenarios randomly by executing P-GA (proposed algorithm) algorithm of Fig. 3 and calculate prioritized fitness value for each randomly generated path. Table 4 shows the three randomly generated test case scenarios (TCSs), their bit representation and PFVs correspondingly.

Table 4: Fitness of initial generated population

TCSs	TCSs	chromosome	PFV_i
TCS1	1->2->3->17	000	11
TCS2	1->2->3->4->5->6->7->8->17	100	26
TCS3	1->2->3->4->5->6->7->8->9->10->11->12->17	110	42

Apply crossover and mutation operators to generate a new population. Table 5 shows the newly generated population after iteration1.

Table 5: Iteration 1

TCSs	TCSs	chromosome	PFVi
TCS2	1->2->3->4->5->6->7->8->17	100	26
TCS3	1->2->3->4->5->6->7->8->9->10->11->12->17	110	42
TCS4	1->2->3->4->5->6->7->8->9->10->11->12->13->14->16->17	111	52

Table 6 and 7 show the newly generated populations by applying the same above procedure (apply crossover and mutation).

Table 6: Iteration 2

TCSs	TCSs	chromosome	PFVi
TCS3	1->2->3->4->5->6->7->8->9->10->11->12->17	110	42
TCS4	1->2->3->4->5->6->7->8->9->10->11->12->13->14->16->17	111	52
TCS5	1->2->3->4->5->6->7->8->9->10->11->12->13->15->16->17	111	52

Table 7: Iteration 3

TCSs	TCSs	chromosome	PFVi
TCS 4	1->2->3->4->5->6->7->8->9->10->11->12->13->14->16->17	111	52
TCS 5	1->2->3->4->5->6->7->8->9->10->11->12->13->15->16->17	111	52
TCS 3	1->2->3->4->5->6->7->8->9->10->11->12->17	110	42

After 3rd iteration as shown in Table 7, the test data 111 has the PFV is 52. So the highest prioritized test case scenario corresponding to chromosome 111 is 1->2->3->4->5->6->7->8->9->10->11->12->13->14->16->17 and 1->2->3->4->5->6->7->8->9->10->11->12->13->15->16->17 and next highest prioritized test case scenario corresponding to chromosome 110 is 1->2->3->4->5->6->->7->8->9->10->11->12->17.

Above three test case scenarios as shown in Table 7, should be tested in order to get full aspectual branch coverage criteria.

COMPARATIVE EVALUATION OF P-GA WITH OTHER TESTING TECHNIQUES

This section describes the comparative evaluation of the obtained results from P-GA (proposed approach) with an un-prioritized and randomly prioritized sequence of transfer scenario of the banking system with three parameters discussed in section 4.

Five simple test case scenarios (un-prioritized sequence) are {TCS 1, TCS 2, TCS 3, TCS 4 and TCS 5} .When these test case scenarios were executed using the P-GA (proposed approach) implemented in JAVA BEANS, we get three test case scenarios {TCS 3, TCS 4 and TCS 5} as shown in Table 7 with APFV (using equation 1 described in above section 4) value is 47.66.

$$APFV = \frac{(42+52+52)}{3}$$

APFV = 47.66

APFV value for un-prioritized test case scenarios {TCS 1, TCS 2, TCS 3, TCS 4 and TCS 5} is 35.66. APFV value for randomly order sequence {TCS 1, TCS 2 and TCS 3} of Table 4 is 28.66 as shown in Fig.6. Therefore, it is evident that proposed P-GA technique performs better than un-prioritized and randomly prioritized test case scenarios.

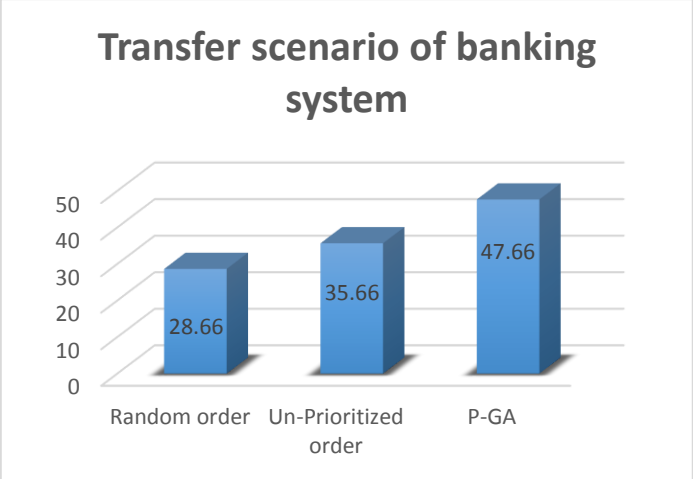

Figure 6: Comparison of APFV value of Transfer Scenario of Banking System

Now, calculate the effectiveness of each testing technique(p-GA, un-prioritized and random) in terms of test case minimization using equation 2.Initially, we have five test case scenarios from TCS1 to TCS5 in case of un-prioritized technique and get three test case scenarios after executing P-GA (proposed approach).Therefore, reduction in a number of test case scenarios is 40%.

$$= 1 - \left(\frac{3}{5}\right) * 100\ \%$$

$$= 40\%$$

Effectiveness of test suit minimization for a random testing technique is also 40% whereas un-prioritized testing has no reduction in a number of test case scenarios. Table 8 shows the effectiveness of each testing technique. It has proved that P-GA (proposed approach) is more effective than other techniques

Table 8: Comparative effectiveness of different testing techniques in terms of reduction a countable number of test case scenarios (%)

Name of Problem	Un-Prioritized order	Random order	P-GA
Transfer scenario of banking system	0	40	40

Now, Aspectual branch coverage criteria (number of test case scenarios which covered full aspectual nodes) for different testing techniques will be measured using equation 3. As shown above in Fig.5, there are four aspect nodes (2, 7, 10, 11).In case of P-GA (proposed approach), we get three test case scenarios as shown in Table 7.Each test case scenario has covered all four aspectual nodes {2, 7, 10, 11}. Therefore, we can calculate aspectual branch coverage criteria for p-GA (proposed approach) using equation 3 as shown below:

$$= \left(\frac{3}{3}\right) * 100$$

$$=100\%$$

In case of un-prioritized testing, there are five test case scenarios as shown in Table 2, in which only three test case scenarios {TCS3, TCS4 and TCS5} has covered all four aspectual nodes{2,7,10,11}.Therefore, Un-prioritized testing technique achieved 60% aspectual branch coverage criteria. Whereas, in case of random testing, only one test case {TCS3} as shown in Table 3 covered all four aspectual nodes. Therefore, we achieved only 33.33% aspectual branch coverage criteria using a random testing technique.

Table 9 shows the achieved aspectual branch coverage criteria by each testing technique and also provide evidence that P-GA performs better than other techniques. P-GA has achieved 100 % aspectual branch coverage criteria.

Table 9: Comparative effectiveness of different testing techniques in terms of aspectual coverage criteria (%)

Name of Problem	Un-Prioritized order	Random order	P-GA
Transfer scenario of banking system	60	33.33	100

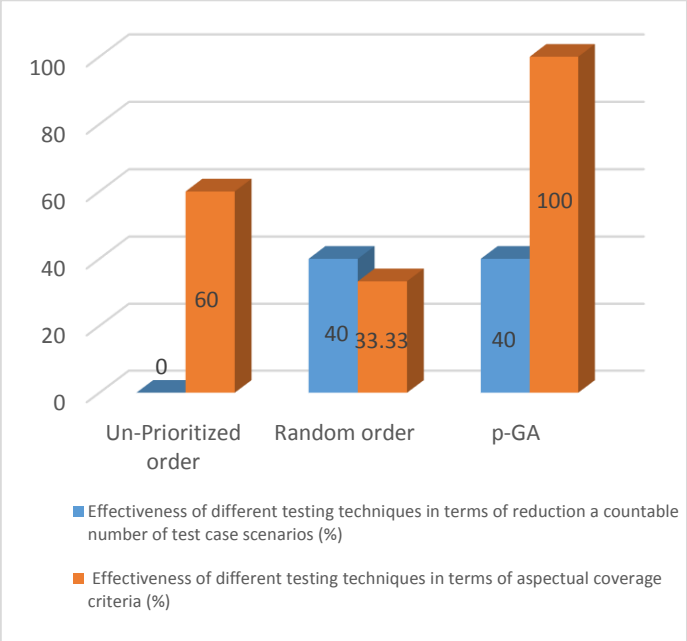

Figure 7: Comparative effectiveness of P-GA with un-prioritized and random method for transfer scenario of banking system

Fig. 7 shows the effectiveness of each testing technique in terms of reduction a countable number of test case sequences and covered aspectual branch coverage. It is evident that P-GA (proposed approach) performs quite well as compare to other testing technique by achieving 100 % aspectual coverage criteria and 40% reduction in number of test case.

CONCLUSION AND FUTURE WORK

An aspect integrated UML activity diagram and a Genetic algorithm is being used to generate prioritized test case scenario. This approach can be used to minimize the testing cost by executing test case scenario which has a high probability of detection of faults. The proposed approach has been applied to various aspect-oriented programs.

The proposed approach has been focused on aspectual branch coverage and applied on small AO Problems. In future work, try to apply proposed approach for comparatively large AO problems, plan to extend the approach for achieving other types of coverage in AOP systems and try to make more efficient using metaheuristic techniques.

REFERENCES

[1] D. E. Goldberg, "Genetic Algorithm in search optimization and Machine Learning", Addison _ Wesley, Reading, MA, 1989.

[2] A. Singhal, A. Bansal and A. Kumar, "A Critical Review of Various Testing Techniques in Aspect-Oriented Software Systems", ACM SIGSOFT Software Engineering Notes, Jul 2013.

[3] A. Ghani and R. Parizi, "Aspect-Oriented Program Testing: An Annonated Bibliography", Journal of Software, Vol.8, June 2013.

[4] S. Dalal and S. Hooda, "Automated Test Sequence Generation of Aspect-Oriented Programs based upon UML Activity Diagram", International Journal of Engineering and Technology [IJET], Vol.9, No.2, Apr-May 2017.

[5] V. Mary Sumanlatha and G. S. V. P. Raju, "A Model Based Test Case Generation Technique Using Genetic Algorithm", in International Journal of Computer Science and Applications, Vol. 1, No. 9, Nov. 2012.

[6] S. Hooda, S. Dalal, and K. Solanki, "A Systematic Review for Model Based Testing an Aspect-Oriented Program", Proceeding of 3rd IEEE International Conference on Computing for Sustainable Global Development, INDIACOM 2016; Available at IEEE-Xplore Digital Library.

[7] S. Dalal and S. Hooda ," A Novel Technique for Testing an Aspect-Oriented Software System using Genetic and Fuzzy Clustering Algorithm", Proceeding of International Conference on Computer and Applications, ICCA 2017;Available at IEEE-Xplore Digital Library.

[8] S. Sabharwal, R. Sibal and C. Sharma, "Prioritization of Test Case Scenario Derived from UML Activity Diagram using Genetic Algorithm", ICCCT, pp.481-485, IEEE(2011).

[9] S. Sabharwal, R. Sibal, and C. Sharma, "Applying Genetic Algorithm for Prioritization of Test Case Scenario Derived from UML Diagram", IJCST, vol.8, May 2011.

[10] C.Kaur and S.Garg , "Testing Aspect-Oriented Software Using UML Activity Diagram ,"International Journal of Engieering and Technolgy,Vol,1,No.3,May 2012.

[11] S.Madadpour,S.Hosseinabadi, C.Kaur and S.Garg , "Testing Aspect-Oriented Software With UML Activity Diagram ,"International Journal of Computer Applications,Vol,33,No.8,Nov. 2011.

[12] M. Badri, L. Badri and M. B. Fortin, "Automated state based Unit Testing for Aspect-Oriented Programs: A Supporting Framework", Journal of Object Technology, Vol. 8, No.3, pp.121-146, June 2009.

[13] M. Harman, F. Islam, T. Xie, and S. Wappler, "Automated Test Data Generation for Aspect-Oriented Program", International Conference of AOSD, 2009.

[14] Y. Suresh and S. k. Rath, "Genetic Algorithm Based Approach for Test Data Generation in Basis Path Testing", in Proceeding of Journal of Software Computing and Software Engineering [IJCSE], Vol.3, No.3, 2013.

[15] Wasiur. R, Taskeen Z. and Vipin S., "Use of Genetic Approach for Test Case Prioritization from UML Activity Diagram", International Journal of Computer Applications, Vol. 115-No.4, Apr.2015.

[16] R. Delmare and N. A. Kraft, "A Genetic Algorithm for Computing Class Integration Test Orders for Aspect-Oriented Systems", 2012.

[17] Z. Cui, L. Wang and Xu. Li, "Modeling and integrating Aspects with UML Activity Diagrams", ACM Symposium on Applied Computing, 2009.

[18] S. Yoo and M. Harman, "Regression Testing Minimization, Selection and Prioritization: A Survey", Software Testing, verification and Reliability, Wiley. Inter Science, vol.22, pp.67-120, March 2012.