International Journal of Computer Applications (0975 – 8887)
Volume 167 – No.4, June 2017

Use of Firefly Algorithm in Optimization and Prioritization of Test Paths Generated from UML Sequence Diagram

Gufran Ahmad Ansari, PhD
Department of Information Technology
College of Computer, Qassim University
Al-Qassim Saudi Arabia (KSA)

ABSTRACT

Software testing is the primary activity to produce reliable software. Reliability of software is very much dependent on the way of testing performed. Software testing, which is usually last activity of the software development cycle is performed under the pressure. Quality and reliability of software are much dependent on test paths which are executed by test cases. Generation of optimized test paths is a challenging part of the software testing process. In this paper, an important effort is made to propose a new technique to obtain the optimized test paths from UML sequence diagram. A tailored algorithm called as Firefly Algorithm is used to get the critical paths. Firefly algorithm is metaheuristic and inspired from flashing behavior of fireflies. A case study of Patient registration system are is used as to explain the proposed approach. Information Flow Metric and their cyclomatic complexity are used for prioritization of test paths. Results indicated that optimized paths from sequence diagram have no redundancy and produced the better results.

General Terms

Optimized Test Path from UML

Keywords

UML, Software Testing, Sequence Diagram, Quality Software, Optimized test paths

1. INTRODUCTION

Software testing is an important process that frequently used to validate the quality of the software. The correct testing can increase software product quality. With the rising demand for reliable software, software testing can add up to 50% of the total software cost. Software testing has not only evolved for look errors or bugs in the software but it becomes a discipline for evaluating the quality software [1]. According to IEEE testing is, "The process of exercising or evaluating system or system components by manual or automated means to verify that it satisfies specified requirements" [2]. It can be performed manually or automatically. Automated software testing is found to be better than manual testing. Software testing process needs more effort with a human interface. In this research paper author mainly generating prioritization and optimization based test paths from UML sequence diagram using Firefly Algorithm. Software testing generally used two methods which are black box testing and white box testing. White box testing is known as structural testing) is to test systematically the internals of the particular program module, black box testing focuses only the output of the software testing is known as functional testing which using functional criteria [3, 4]. Mostly software tester engineer uses test cases to find out whether software system fulfills the predefined requirements or not. Now a day's many models are used by the tester that can generate test cases automatically. With the big demand, complexity and size software systems require orderly, scalable and automated testing approach. Sequences of conditions that fulfill certain coverage criteria are called test cases. Test Case Generation by means of Unified Modeling Language, UML Sequence Diagrams, and Labeled Transition Systems introduced UML Sequence diagram in the field of test case generation [5]. Test cases generated from these models help to find uncertainty and inconsistencies in the requirement and design of the system. Generated test cases should apply in such a way that it can give maximum throughput by the uncovering fault. Due to inherent complexity, big systems are very difficult to test the system because it's needed big numbers of test cases that are required to test these types of the systems. Generating test cases is a challenging task in software testing. Prioritization and optimization frequently applied to run the test cases in order which may disclose faults earlier in the process of testing. Test case prioritization is a proficient technique to ensure trustworthy and good quality software. For good quality software product must be optimized and delivered on time. A test case is the triplet [I, S, O], where I is the initial state of the system, S is the input test data and O is the expected output of the system [6]. Data flow information and Control flow are generally getting from software source code. Data flow information and Control flow have an important impact on test case generation process. Selecting right sequence is a difficult part in software testing [7]. Generally, Test cases are generated from the source code of the program. After the analysis and designed the software code can be generated. So testing the software in beginning or early stage is very difficult. For avoiding wasting time, effort and cost in testing process its need that the test cases should be designed at designed level and for doing this reliability of the software will be increased [8, 9]. Good test cases are directly linked to the accuracy of the model and the system which will be used for generation of test cases. Test cases are generated from the system model which effectively gathers attribute of the implementation under test will make available high probability of finding bugs or defects. It is necessary that Test cases should be updated with the change in the design of the system which is not all the time feasible physically. Therefore manually creation and execution of test cases are error prone also costly. Automated test case generation can enhance the software reliability with the increase in coverage and decrease the development cost of the software testing. There is an urgent need to plan test cases in such way that it can get better fault detection and coverage rate [10].

Through UML Object-oriented systems can be easily modeled. Sequence diagrams are used for describing the behavior by modeling the flow of messages in a system. Sequence diagram illustrates how an object, or group of objects, work together within a system. The sequence diagram shows how the messages are exchanged between the objects. Sequence diagrams are well appropriate for objected oriented software. Sequence diagram demonstrates how the objects interact each other and messages exchanged between the objects. Communication in a sequence diagram is a sequence of messages between the objects to perform a specific task [11, 12]. In the early stage of software development cycle, researchers are used UML based testing for many years. At the same time, Code based prioritization techniques are investigated by most researchers and prioritization of test cases from UML diagrams has not been given much attention by researchers so far [13]. The creation of test sequence has been an issue under consideration for many years to help such solutions to the testing problems which are proficient and helpful in nature. In 1992, some formal methods were reviewed by H. Ural. He reviewed some Finite State Machine (FSM) based specifications for test sequence generation [14]. Li et al. [15] generated test sequences using Euler circuit algorithm. Although generated test paths are minimized with some redundant transitions. For the generation of the optimized test sequence, Srivastava et al used Cuckoo search [16]. Rhmann and Saxena [30] generated test cases from UML sequence diagram using Extenics theory. Authors converted the Message Flow Graph (MFG) of Sequence diagram into it's the dual graph and then generated minimized test cases from the dual graph. XML is drawn from Sequence diagram and from XML MFG is created. A case study of aircraft departure activity is used to validate the proposed approach.

The proposed approach generates prioritized and optimized test sequences from sequence diagram. This approach employ in the Firefly algorithm which is inspired by flashing behavior of firefly and developed by Yang [17]. Firefly algorithm is successfully applied in the different field of research like image processing [18], energy conservation [19] in wireless sensor network [20] and structural optimization [21]. Information Flow Metric [22] is applied to the component of the system design. Authors considered nodes of MFG as components. For each node IF value is calculated. For each node IF value is calculated from the given equation (1)

$$IF(A) = [FANIN(A) \times FANOUT(A)]^2 \qquad (1)$$

Where FANIN (A) is a number of nodes that call to node A and FANTOUT (A) is a number of nodes called by node A.

Software complexity can be measured by Cyclomatic complexity. Software modules which have the higher value of cyclomatic complexity have a higher probability of containing errors.

UML is a standardized modeling language and it based on a standard of Object Management Group (OMG). The UML is heavily used in software engineering as visual modeling language which is used to make and document the artifacts of software [23-24].The Unified Modeling Language developed by G. Booch which provides the graphical tool for modeling and designing software and hardware problems. OMG has defined several UML specifications and standard representation of UML [25-26]. Ansari, G.A**,** Rhmann and Saxena V**.,** proposed, a novel technique of test cases prioritization from UML state diagram by taking account risk. State machine diagram is transformed into WEFSM (Weighted Extended Finite State Machine) and a case study of ATM system is used to evaluate the proposed approach [31]**.** Authors provided a survey on the UML state machine diagram that has been measured for generating test cases. They generated test data for concurrent state and events from UML state machine [27].

This paper includes introduction work in section (1) introduction, (2) Background,(3) Proposed methodology for generation of test paths and optimization, (4) Sequence Diagram (5) Adjacency Matrix (6) Conclusion.

2. BACKGROUND

2.1 UML Sequence Diagram

The sequence diagram is made up of objects and messages. In the sequence diagram, objects are shown as rectangular boxes on the top of the diagram. A sequence diagram shows object interactions set in time sequence. The communication objects show by an arrow and the message on that arrow. The sequence diagram passes the message from top to bottom. Sequence diagrams are also called event diagrams.

2.2 Firefly Algorithm

Firefly algorithm is used to solve optimization problems. It is inspired from flashing behavior of firefly. Three main rules are used in firefly algorithm based optimization [17]: Each firefly can be attracted to other firefly and attraction of fireflies is determined by the brightness value of firefly. An objective function is a use for calculation of brightness value of firefly. Firefly with less brightness value will move to higher brightness value firefly. The Pseudo code of firefly is summarized in Fig. 1. The attractiveness of firefly and variation of light intensity are two main factors used in firefly algorithm. As source move away attractiveness decreases. Intensity of light can be defined as

$$I(r_{ij}) = I_0 e^{-\gamma r_{ij} ..2} \qquad (2)$$

Where γ is light absorption coefficient, r_{ij} is the distance between fireflies i and j are for x_i and x_j respectively.

Update formula for Firefly i being attracted to another more attractive Firefly j is calculated by equation 5.

$$\Delta X_i = \beta e^{-\gamma r2}\, ij\left(X_j^t - X_i^t\right) + \alpha e_i, Xi^{t+1} = X_i^{\,t} + \Delta X_i \qquad (3)$$

Where t is generation number, e_i is random vector and a is randomization parameter. Firefly algorithm [24] is given below in pseudo code form:

```
Firefly_ algorithm()
        objective function f(x), where $x=(x_1,\ldots, x_d)^T$
        initial population of firefly $x_i$   (i=1,2……..,n)
        the brightness of firefly $x_i$  is $I_i$ determined by $f(x_i)$
        light absorption coefficient v is defined
                while( t<MaxGeneration)
                        for i=1:n all n fireflies
                        for j=1:n all n fireflies
                        if($I_i$<$I_j$), move firefly i towards j;
                end if
                        vary attractiveness with
                distance r via exp[-γr]
                        evaluate new solutions and
```

```
                    update light intensity
                            end for j
                            end for i
                            rank the firefly and current
                    global best g is find
                    end while
result and visualization
```

3. PROPOSED METHODOLOGY

The following is the procedure for optimized test cases generation from UML Sequence diagram:

1. Generate UML Sequence diagram from requirement specification of the given project;
2. Draw Message Flow Graph(CFG) from the Sequence diagram;

   Message Flow Graph (MFG) from Sequence diagram is designed where each node represents a message and flows of messages are represented by edges connecting the nodes [25];
3. Convert the Message Flow Graph(MFG) into Adjacency Matrix;
4. Information flow metric and cyclomatic complexity of each node is calculated from the Adjacency Matrix of MFG;

   Cyclomatic Complexity of directed graph G is calculated using the following formula:

   $$v(G) = 1 + \left(\sum_{i=1}^{n} ReducedOut\deg ee(i) \right); \qquad (4)$$

   Where for each node reduced out degree is one less than the out degree of that node [26]. Cyclomatic complexity for each node of MFG is calculated from adjacency matrix of the CFG. Authors counted reduced out degree of nodes above the node in adjacency matrix for which cyclomatic complexity is being calculated and added 1;
5. Optimized test paths from sequence diagram are generated using Firefly algorithm. A new matrix decision matrix [28] is introduced which contains decision factor for each edge of MFG.

   $$DF_i = \frac{1}{\left[10 \times \left\{CC_i \times (N - i) - 0.10\right)\right\}\right]} \qquad (5)$$

   Where, N is a total number of nodes, CC_i is the cyclomatic complexity of node i. Decision factor determine the brightness value of the nodes;
6. Test paths are prioritized by applying three fire flies at each node of the MFG of the Sequence diagram. The brightness value of each firefly is determined by the formula given by equation 8:

   $$B_i = \frac{B_0}{1 + \gamma d} \qquad (6)$$

Where brightness value of firefly at node 1 is B_0 and

$$\gamma = IF_i + CC_i$$

Information Flow Metric and cyclomatic complexity at node I are defined as IF_i and CC_i

For a node d is random distance calculated from start node to node of MFG at which fireflies are deployed.

Mean of brightness of firefly at each node is used to determine the priority of test path. For each test path mean of brightness is calculated. Higher mean brightness value paths are assigned higher priority.

4. SEQUENCE DIAGRAM

Now, let us consider a case study of Patient Registration System whose sequence diagram is designed and represented in given Fig1.

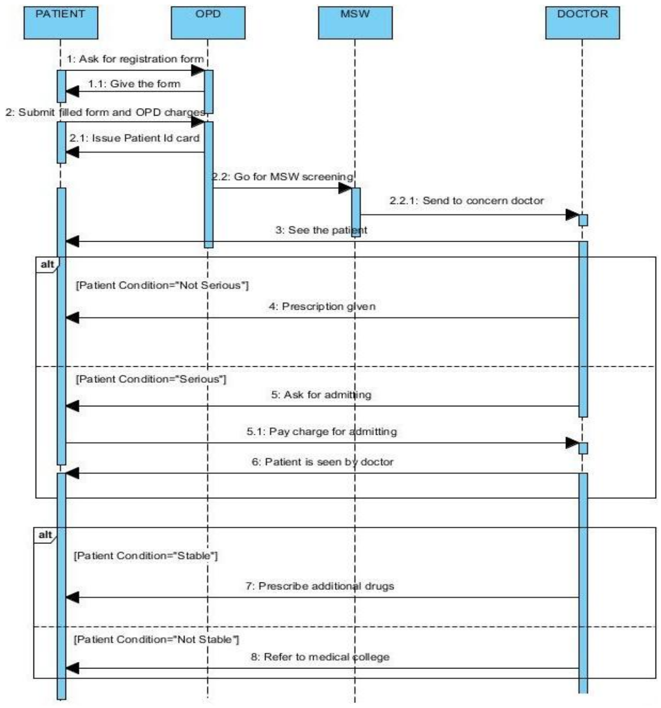
PATIENT
OPD
MSW
DOCTOR
1: Ask for registration form
1.1: Give the form
2: Submit filled form and OPD charges
2.1: Issue Patient Id card
2.2: Go for MSW screening
2.2.1: Send to concern doctor
3: See the patient
alt
[Patient Condition="Not Serious"]
4: Prescription given
[Patient Condition="Serious"]
5: Ask for admitting
5.1: Pay charge for admitting
6: Patient is seen by doctor
alt
[Patient Condition="Stable"]
7: Prescribe additional drugs
[Patient Condition="Not Stable"]
8: Refer to medical college

Fig 1: Sequence Diagram for Patient Registration System

In Fig: 1, different activities of the patient registration process are shown. The first patient goes Out Door Patient and asks for the registration form. OPD counter person gives the form to the patient and patient fill the form and along with charges submit the form to the OPD. After that OPD issues an ID card to the patient. Patients take the ID card and go to MSW department for screening, later than MSW department send patient to concern doctor. Concern doctor sees the patients and if the patient condition, is not serious give prescription to the patient and if patient condition is serious then ask for the admission. In the serious condition patient pay charges to the bill section and bill section provide the payment record slip to the patient. The patient is seen by the doctors and if patient condition is stable then doctors prescribed additional drugs to the patient and if patient condition is non-stable then refer to the patient to medical college. MFG for Patient Registration System shown below in Fig: 2.

5. ADJACENCY MATRIX

In adjacency matrix, each node represents corresponding message node 7 and 11 have two edges coming out as there all fragments.

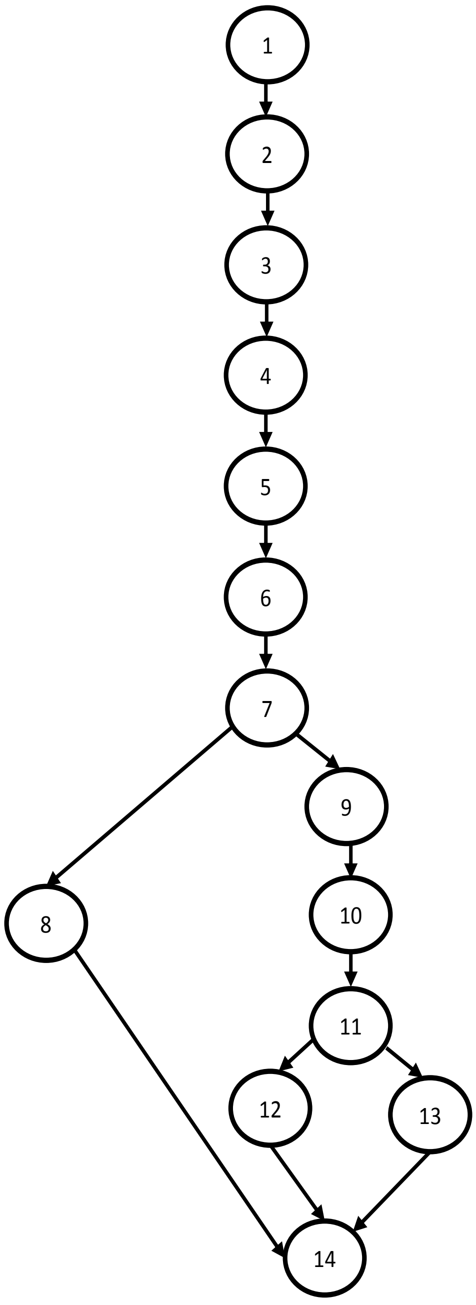

Fig 2: Message Flow Graph (MFG) for Patient Registration System

Cyclomatic Complexities are computed from adjacency matrix using equation 4 and are given below:

Cyclomatic Complexity

(CC at node 1 to 6 = 3)

(CC at node 7 to 10 = 2)

(CC at node 11 to 13 = 1)

The adjacency matrix is used to calculate out degree of each node. Each node's out-degree is calculated from the sum of a number of 1's in each row. This is recorded in Table 1.

Table 1. Adjacency Matrix for Patient Registration System

Nodes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Out Degree
1		1													1
2			1												1
3				1											1
4					1										1
5						1									1
6							1								1
7							1	1							2
8														1	1
9										1					1
10											1				1
11												1	1		2
12														1	1
13														1	1
14															0

Information flows metric values for each node calculated from MFG using equation 1.

IF $_1$ = 0, IF $_2$ = to IfF6 = 1, IF $_7$ = 4, IF $_8$ = to IF $_{10}$ = 1, IF $_{11}$ = 4, IF$_{12}$ = 1, IF $_{13}$ = 1, IF $_{14}$ = 0

Decision factor for each node is calculated using equation 5 and given below:

DF$_1$=1/ [10× {CC$_i$× (N-i)-0.1)}] = DF1=1/ [10× {3× (14-1)-0.1)}] = 0.00239

DF$_2$ = 1/ [10× {CC$_i$× (N-i)-0.1)}] = 1/ [10× {3× (14-2)-0.1)}] = 1/ 10×3×11.9 = 1/357 = 0.0028011

DF$_3$=1/327 = 0.00305, DF$_4$=1/297 = 0.00336, DF$_5$=1/267 = 0.00374, DF$_6$=1/237 = 0.00421, DF$_7$=1/207 = 0.00483,

DF$_8$=1/118 = 0.00847, DF$_9$=1/98 = 0.0102, DF$_{10}$=1/78 = 0.0128, DF$_{11}$=1/58 = 0.0172, DF$_{12}$=1/19 = 0.0526, DF$_{13}$=1/9= 0.0111

DF$_{14}$=1000

Algorithm for test path generation traverse edge from adjacency matrix if there are two edges coming-out from a node then select node with higher decision value and print that node.

Remove all selected nodes from adjacency matrix while traversing the matrix and replace 1 with 0 in matrix. Repeat this process till adjacency matrix becomes null matrix.

Cyclomatic complexity, Information flow metric and decision factor of each node are calculated using Equations 1, 6 and 7 respectively. For each node the values of the cyclomatic complexity, information flow metric and decision factor are computed and recorded in Table 1:

Table 2. Decision Table for message Flow metric and Decision Factor

Nodes	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0.0024	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0.00280	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0.0031	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0.0034	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0.00374		0	0	0	0	0	0	0
6	0	0	0	0	0	0.00421		0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0.0085	0.0102	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	1000
9	0	0	0	0	0	0	0	0	0	0.013	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0.17	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0.53	0.111	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	1000
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1000
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Test paths are generated by traversing the MFG. For selection of node in test path in case of decision node, node with the higher decision factor are selected. Generated test paths are given below in Table 3.

Table 3. Test paths generated from UML sequence diagram

Test Path 1:	1- 2- 3- 4 -5- 6 - 7 - 9- 10 - 11 -13 -14
Test Path 2:	11-12-14
Test Path 3:	7-8-14

Generate three firefly at each node:

Take a look at node 1, value of A at node 1 is taken as 100 and distances of node 2, 3, ---- are taken from node 1.

Table 4. Different nodes and brightness of fireflies at each node of MFG

Node Number	d_i	γ	$B_i = \frac{B_0}{1+\gamma d}$	**Node Number**	**d_i**	γ	$B_i = \frac{B_0}{1+\gamma d}$
2	1.1	4	18.518	8	7.1	3	4.484
	1.2		17.241		7.2		4.424
	1.3		16.129		7.3		4.366
3	2.1	4	10.204	9	8.1	3	3.952
	2.2		10.638		8.2		3.906
	2.3		9.803		8.3		3.861
4	3.1	4	7.462	10	9.1	3	3.533
	3.2		7.246		9.2		3.496
	3.3		7.042		9.3		3.460
5	4.1	4	5.747	11	10.1	5	1.941
	4.2		5.617		10.2		1.923
	4.3		5.499		10.3		1.904
6	5.1	4	4.672	12	11.1	2	4.310
	5.2		4.587		11.2		4.273
	5.3		4.504		11.3		4.237
7	6.1	6	2.659	13	12.1	2	3.968
	6.2		2.617		12.2		3.937
	6.3		2.577		12.3		3.906

Table 5 shows test paths generated from Firefly algorithm and test paths prioritization values are calculated from the mean of the brightness values on each node of MFG.

Table 5. Test paths with their prioritized values

TS1	1-2 -3- 4 -5- 6 -7 - 9- 10 - 11 -13 -14 **13.403**
TS 2	11-12-14 **2.065**
TS3	7-8-14 **2.347**

Table 6 shows the paths generated from proposed approach on case study taken by [29]. Authors also generated test paths from Genetic algorithm based approach. Test paths generated from Genetic Algorithm have redundant edges and nodes.

Table 6. Test paths generated by proposed approach and by Genetic algorithm [29]

Test paths by Firefly Algorithm		**Test paths by GA**	
1	1-2-6	1	1,2,3,4,5,2,3,4,2,3,4,2,6
2	2-4-5	2	1,2,4,5,2,4,5,4,2,4,2,6
3	2-3-5-4	3	1,2,4,5,4,5,2,3,4,5,2,6
4	3-4-2	4	1,2,6,4,5,4,2,3,4,2,6
5	5-2	5	1,2,4,5,2,4,5,4,2,6
		6	1,2,6

Fig3. Shows the comparison of our Firefly Algorithm based approach with Genetic Algorithm based approach [29]. It is observed that Firefly Algorithm based test paths have no redundant nodes or edges. Such elimination of redundancy can save huge cost associated with testing.

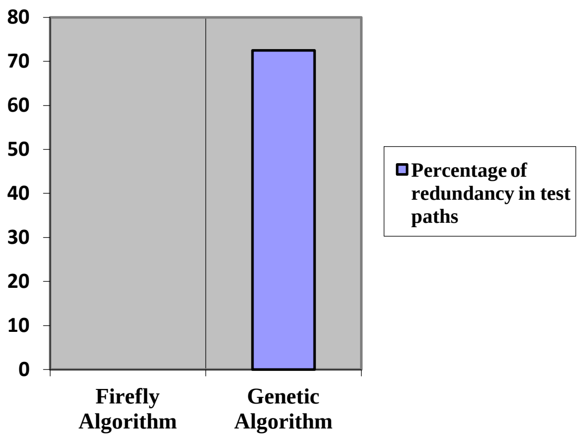

Fig 3: Comparison of our approach with Genetic Algorithm based approach [29]

6. CONCLUSIONS

This paper presents firefly algorithm based test paths generation from UML sequence diagram. UML sequence diagram can be very helpful in cluster level testing of the software. Test paths are usually used by software engineers to check the functionality of the software. There may be an infinite number of test paths for even small size software. Testing all paths in not feasible as testing is usually performed in pressure. Optimized test paths generated from sequence diagram can be very helpful in identification of faults in interactions of objects. Identification of faults earlier in testing can save resources and time. Results indicated that our approach is better in the removal of redundancy in test paths. Redundancy in test paths takes the extra cost to check the functionality. We compared our approach with Genetic Algorithm and found that Firefly Algorithm is better in comparison of Genetic Algorithm for test path prioritization.

7. REFERENCES

[1] R. A. Khan and R.K Choudhary, “Software Testing Process: A Perspective Framework” ACM SIGSOFT Software Engineering Notes” Volume 36, Number 4, pp 1-5, July 2011.

[2] Rajvir Singh, “Test Case Generation for Object-Oriented Systems: A Review” IEEE, Fourth International Conference on Communication Systems and Network Technologies, 2014

[3] R. S. Pressman, Software Engineering: A Practitioner’s Approach, 7th Edition, McGraw-Hill, 2010.

[4] Soma Sekhara Babu Lam et al. “Automated Generation of Independent Paths and Test Suite Optimization Using Artificial Bee Colony” Procedia Engineering, Elsevier pp. 191-200, 2012.

[5] Emanuela G. Cartaxo, Francisco G. O. Neto, and Patr´ıcia D. L. Machado, "Test Case Generation by means of UML Sequence Diagrams and Labeled Transition Systems", IEEE 2007.

[6] Vikas Panthi and D.P. Mohapatra, “ Test Scenarios generation Using Path Coverage”, *International Journal of Computer Science and Informatics*, pp 64-68, Volume 3, Issue 2, 2013

[7] P. R. Srivastav a, K. Baby and G. Raghurama, “An approach of optimal path generation using ant colony optimization”, *In: Proceedings of the TENCON 2009 - 2009 IEEE Region 10 Conference*, Singapore, pp.1-6, 2009,

[8] Abdurazik, A., Offutt, J., “Using UML collaboration diagrams for static checking and test generation”, *In: Proceedings of the 3rd International Conference on the UML. Lecture Notes in Computer Science*, vol. 1939, pp. 383–395. Springer, New York (2000)

[9] Ali, S., Briand, L.C., Jaffar-ur-Rehman, M., Asghar, H., Zafar, Z., Nadeem, A. “A state based approach to integration testing based on UML models”, *J. Inf. Softw. Technol*. 49(11–12), 1087–1106 (2007)

[10] S Gosh, R France, C. Braganza, N. Kawane, A Andrews and O Pilskalns, “Test adequacy assessment for UML design model testing”, *In: Proceeding of the international symposium on the software reliabilty engineering*, Denver, CO., 2003, pp. 332-343.

[11] Rumbaugh, J., Jacobson, I., Booch, G.: The UML Reference Manual. Addison-Wesley, Reading (2001)

[12] Ajay Kumar Jena et al, “Model Based Test Case Generation from UML Sequence and Interaction Overview Diagrams” Proceedings of the International Conference on “Computational Intelligence in Data Mining Springer (ICCIDM-2014)

[13] Wang Linzhang, Yuan Jiesong, Yu Xiaofeng, Hu Jun, Li Xuandong and Zheng Guoliang, “ Generating Test cases from UML Activity diagram based on Gray-box Method”, *Proceeding of the 11th APSEC’ 04, IEEE.*

[14] Hasan Ural: Formal methods for test sequence generation. *Computer Communications* 15(5): 311-325 (1992).

[15] Liping Li, Xingsen Li, Tao He and Jie Xiong, “Extenics based test case generation from UML Activity diagram”, *Information Technology and Quantitative Management*, 2013, pp. 1186-1193.

[16] Praveen Ranjan Srivastava, Chandolu Sravya, Ashima, Sai Kamisetti and Manogna Lakshmi, “Test sequence optimization: an intelligent approach via cuckoo search”, *International Journal of Bio-Inspired Computation*, Vol. 4, No. 3, 2012.

[17] X. S. Yang, “Firefly algorithms for multimodal optimization, in Stochastic Algorithm: Foundations and Applications”, *SAGA, Lecture Notes in Computer Science*, 2009, 169-178.

[18] Ming Huwi Horng, “Vector quantization using the firefly algorithm for image compression”, *Expert Systems and Applications*, Vol. 39, 2012, pp. 1078-1091.

[19] Lendro Das Santos Coelho and Viviana Coco Mariani, “Imroved firefly algorithm approach applied to chiller loading for energy conservation”, *Energy and Buildings*, Vol. 59, 2013, pp. 273-278.

[20] Philipp Sommer and Roger Wattenhofer,“Gradient Clock Synchronization in wireless sensor networks”, *In Proceeding of IEEE International Conference on Information processing in sensor networks, USA,* 2009, pp. 37-48.

[21] Amir Hossein Gandomi, Xin She Yang and Amair Hossein Alavi, "Mixed Variable structural optimization using Firefly algorithm", *Computer and Structures*, Vol. 89, 2011, pp. 2325-2336.

[22] Pankaj Jalote, An Integrated Approach to Software Engineering, 3rd edition, Springer, 2005.

[23] Ali Al-Khalifah and Ansari, G.A., "Modeling of E-procurement System through UML using Data Mining Technique for Supplier Performance", IEEE *International Conference on Software Networking (ICSN*), South Korea 2016.

[24] G. Booch, Object Oriented Analysis and Design with Applications, 2nd edition, Addison Wesley, 1994.

[25] OMG, Unified Modeling Language Specification, available online via http://www.omg.org.,2011

[26] OMG, OMG XML Metadata Interchange (XMI) Specification, available online via http://omg.org.

[27] Aggarwal M, Sabharwal S. Test case generation from UML state machine diagram: A survey. IEEE 3rd International Conference on Computer and Communication Technology (ICCCT); pp. 133–40.Nov 23–25, Allahabad 2012.

[28] Wasiur Rhmann and Vipin Saxena, Optimized and Prioritized Test Paths Generation using Firefly Algorithm from UML Activity Diagram, *International Journal of Computer Application*, Vol. 145, No. 6, pp. 16-22, 2016.

[29] Hoseini and Saeed Jalis, "Automatic Test path generation from sequence diagram using Genetic Algorithm", *International Symposium on telecommunication, IEEE*, 2014, pp. 106-111.

[30] Wasiur Rhmann and Vipin Saxena "Generation of Test Cases from UML Sequence Diagram using Extenics Theory", *British Journal of Mathematics and Computer Science,* Vol. 16, No. 1 , 2016, pp. 1-16.

[31] Ansari, G.A, Rhmann and Saxena V., "Risk based Test Case Prioritization using UML State Machine Diagram" International Journal of Applied Information Systems (IJAIS) Vol. 11, No.-7, pp. 15-21, December 2016.