Indonesian Journal of Electrical Engineering and Computer Science
Vol. 12, No. 1, October 2018, pp. 51~60
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v12.i1.pp51-60

Comparative Analysis of Various Testing Techniques Used for Aspect-Oriented Software System

Sandeep Dalal1, Susheela Hooda, Kamna Solanki
M.D. University, Rohtak, Haryana, India

Article Info

Article history:

Received Jan 19, 2018
Revised Apr 28, 2018
Accepted Jun 14, 2018

Keywords:

AOSS testing
AOSS testing survey
Aspect-oriented programming (AOP)
Aspect-oriented software system (AOSS)
Software testing

ABSTRACT

Nowadays, Aspect-Oriented Programming (AOP) paradigm is getting more popularity in the field of software development. But testing an Aspect-Oriented Software System (AOSS) is not well matured. Therefore, many researchers have been focusing on testing an AOSS. Moreover, the literature indicates that very few papers have devoted to literature survey but still there is need to study in depth of various testing techniques used for AOSS. Therefore, in this paper, a comprehensive study of existing various testing techniques for AOSS have been conducted and present a comparative analysis result of various testing techniques based on various parameters.

Corresponding Author:

Susheela Hooda,
M.D. University,
Rohtak, Haryana, India.
Email: susheelahooda@gmail.com

1. INTRODUCTION

Aspect-Oriented Programming (AOP) is relatively the latest programming paradigm as compared to traditional programming paradigm which focuses on separation of crosscutting concern. A program consists of two major things such as 1) Business Logic and 2) Supporting functions. AOP isolates the supporting functions from the main business logic in which supporting functions are known as crosscutting concerns and business logic is known as a primary concern. Therefore, AOP increases the concept of modularity by minimizing or no code scattering. Therefore, AOP has been used to develop complex software systems.

According to Dalal et. al. [1], software testing plays a critical role to ensure the quality of the software. Soft computing techniques have proven successful in the area of software testing. Mohapatra and Prasad proposed ant-colony optimization technique for reducing the size of test suite and proved its effectiveness with other contemporary techniques [2]. Aspect-oriented software system testing is not so much matured as other programming paradigms such as object-oriented and procedural programming paradigm. Therefore, many of the researchers have worked in the field of AOP testing but still one needs to focus more in this field. Quality of the system is directly dependent on the testing. Proper testing of software leads to design and develop a quality software. Software testing can be performed manually as well as automatically [3], [4].

According to Ghani [5], automation of software testing process is performed to minimize cost and to decrease the human errors. Automation of AOP testing approaches depends on three elements: (i) automated test input generation and selection (ii) automated test oracle and (iii) automated test execution. Apart from above elements, other important characteristics are testing techniques for AOP, different testing level and coverage criteria for testing AOP.

Various researchers [3]-[7] describe aspect-oriented program testing surveys. But still, no formal survey has been found for AOP testing approaches as per our knowledge. This paper will provide the detailed analysis of AOP testing approaches by using a formal procedure [8] systematic review.

This paper does not cater to compare the existing AOP testing techniques based on their merits and demerits. The focus of this paper is to give a comparative analysis study of existing aspect-oriented software testing techniques based on some characteristics such as testing technique, testing level, automation support, automation level and test case generation criteria.Organization of this paper is as: Section 2 describes the survey planning and execution of the literature survey. Section 3 discusses detail analysis study on the available testing techniques for AOP and section 4 describes the various issues which occur usually during AOP testing. Section 5 presents the conclusions and future work.

2. SYSTEMATIC REVIEW: SURVEY PLANNING AND EXECUTION

According to B. Kitchenham [8], a systematic review is a planning which includes the methods to identify, analyze and clarify the most appropriate research about a specific research question. The following steps have to carry out the review in this literature are as follow: aim, research questions, selection of available sources, selection of search strings, classification and extract the information from each AOP testing approach.

The following steps have been taken using the approach of B. Kitchenham [8]:

Aim: To characterize the various AOP testing approaches from the available literature.

Research Question:-What are the available AOP testing techniques and their characteristics.

Sources: ACM Digital Library, IEEE X-plore, open access journals, springer, Science direct, Elsevier , various journals and conference proceedings.

Search String:- (“AOP testing techniques”)or(“Testing Aspect-oriented programs or software”)or(“unit testing for AOP”)or(“Data flow testing for AOP”)or(“automated testing for AOP”)or(“Model based AOP testing”) or(“control flow testing for AOP”)or(“Structural testing for AOP”) or(“Search based AOP testing”)or(“Regression testing for AOP”)or(“Random testing for AOP”)or(“integration testing for AOP”).

AOP testing techniques have been classified into different fields based on a review of [3]-[7] shows that in Table 1.

Table 1. Characterization of Various AOP Testing Techniques

Field	Description
Testing Technique	Define various testing techniques for AOP which are classified differently by different researchers.
Level of Testing	Define which level of testing is used to test the software behavior. Different level of testing used are: unit, integration, system, acceptance and regression testing.
Test Coverage Criteria	Define criteria to select which input sets are to be incorporated.
Tools to Support	Define the existence or not of supporting tools.
Automation Level	Automation means the reduction in cost, time and effort. Defines how automated is a proposed testing approach.
Behaviour Model	Describe which behaviour model is being used by testing technique.

3. ANALYSIS OF ASPECT-ORIENTED SOFTWARE TESTING TECHNIQUES

To have the better understand the present scenario of various testing approaches for an aspect-oriented software system. The following section discusses the various papers which have extensively covered the basic aspects of aspect-oriented software programming paradigm and their testing approaches.

3.1. Testing Techniques for AOP

There are various testing techniques for AOP which are classified differently by different researchers [3], [4] and [6]. However, a broad classification of testing techniques for AOP given shown in Figure 1 and detail description of each testing technique.

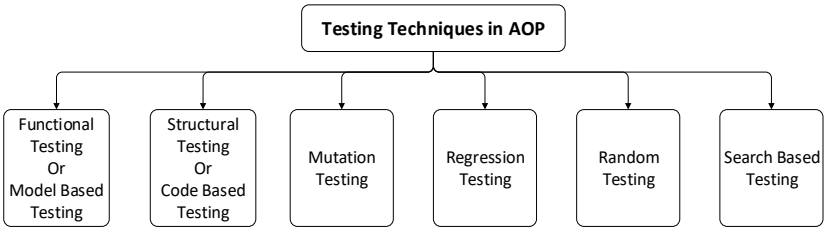

Figure 1. Testing techniques in AOP

3.1.1. Functional/Model-Based Testing

Functional testing is used to test the external behaviour of software whether software meets the customer‘s requirement or not without bothering the implementation details of the software. Model-based testing (MBT) is also considered as functional testing because like functional testing, it has also intended to test the external behaviour of the system. Moreover, UML (Unified Modeling Language) has proven successful to predict test effort in earlier stages of the software development process [9]. MBT was originally developed for system testing. However, MBT can be used in unit testing, integration testing and in other testing techniques [10]-[30].

3.1.2. Structural/Code Based Testing

Structural testing is used to test the software’s implementation in which test data is derived from the knowledge of the internal detail. (e.g. control path, data items). This knowledge can be gathered only when one has knowledge about the source code. This is the reason that structural testing is considered as code-based testing. Control flow and data flow testing have also come under structural testing. As control flow testing requires knowledge about the control flow structure of the program and data flow testing requires knowledge about the test paths of a program which can be derived from the sequence of events related to the data state [31]-[45].

3.1.3. Mutation Testing

Mutation testing is also known as fault-based testing. In order to perform mutation testing, mutants are generated from the original program by modifying the original program with the help of different mutation operators. Thereafter, mutation testing is performed to identify whether test data is able to discover the modifications which were performed in original program or not [46]-[60].

3.1.4. Regression Testing

Regression testing is performed during the maintenance phase of the software to ensure that all faults had been detected which occurred due to various changes in software while modifying the software [61]-[64].

3.1.5. Random Testing

Random testing is produced test data randomly rather than one’s choice. It is one of the simple testing technique and less costly as compared to other testing techniques. R. M. Parizi et.al. [66] proposed a framework for testing an aspect-oriented program using random testing. According to the author, this approach is the first approach for random testing for an aspect-oriented program. The author also extends his own work in [67].

3.1.6. Search-Based Testing

Search based testing is also considered as evolutionary testing based on the theory of evolution. It formulates the fitness function to select test data based on some objectives such as coverage criteria. Better fitness function helps to solve the search problem in better way [68]-[73].

Currently, 62 papers have been classified into different categories. Table 2 shows the complete list of the selected papers with their classification of different categories.

3.2. Level of Testing

Table 3 shows the different level of testing of AOP and number of approaches from Table 4 which extends the scope of application of AOP testing approaches.

\

Table 2. Various Classifications of Selected Papers

Paper Categories	#Papers
Functional Testing or Model Based Testing	21
Structural Testing or Code-based Testing	15
Mutation Testing	15
Regression Testing	4
Random Testing	2
Search Based Testing	5
Total of papers	62

Table 3. Quantitative Analysis of Testing Level

Testing Level	# Approaches
Unit testing	9
Integration testing	10
System testing	30
Regression testing	3

3.3. Test Case Coverage Criteria

Coverage criteria is a measure to determine which inputs are to be incorporated into test suit. Effective coverage criteria help to test the software effectively. As literature indicates that not all coverage criteria which have been used in traditional programming paradigm are directly useful for testing AOPs. Therefore, a broad classification of coverage criteria has been shown in Figure 2 and brief description of each coverage criteria has been discussed:

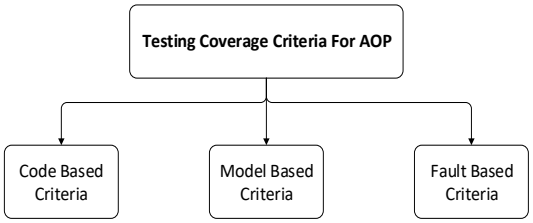

Figure 2. Different test case coverage criteria for AOP

3.3.1. Code-Based Criteria

The source code of AOP or some form of graph model are used to define code-based coverage criteria. Code–based criteria can be defined by two methods such as (1) aspect coverage criteria and (2) aspectual branch coverage criteria [5]. Aspect coverage criteria include statement coverage, joinpoint coverage, context coverage and def-use coverage [39], [41] whereas aspectual branch coverage criteria include interaction coverage, data flow coverage and data coverage [33], [36], [40], [43], [45]. Moreover, new flow graph models have proposed to handle the aspects integration with source code such as def-use graph model (AODU), PairWise Def-Use (PWDU) [39], [41] and so on.

3.3.2. Model-based coverage criteria

Control flow graph which is generated from UML model is used to define model-based criteria. Various coverage criteria have been used to define model-based criteria such as transition coverage, sequence coverage, multi aspect integration, polymorphic, branch coverage and aspectual branch coverage criteria [14]-[18], [22]-[24] has been used to generate the test data.

3.3.3. Fault based criteria

It is used to generate test data to detect faults in the mutated version of the program [46]-[60].

3.4. Tools to support

There are some approaches [12], [14], [15], [19], [23], [25], [26], [29], [36], [38], [43], [44], [46], [49], [51]-[54] and so on which provide automated tools to apply their algorithms and some approaches still support manual. For test case generation, manual approach is trivial, it takes more time that leads to a cost of the software high. Therefore, one should have to try to design tool which supports the proposed approach.

However, automated supporting tools provide rich features but it limits its usage in an organization due to a heavy license fee. Analyzing all supporting tools which have or not been applied in various AOP testing approaches is a difficult task. Therefore, Table 4 only shows the present or absent of supporting tool on the basis of the information which has been provided by authors [4]-[6]. Counting all supporting tools and analyzing them is not the intention of this paper.

Table 4. Comparative Analysis of Various Testing Techniques for Aspect-Oriented Software System

Author /Year of publication	Testing Technique/Behavior Model	Testing Level	Test Coverage Criteria	Tools to Support Approach	Automation Support to Test
J. Zhao[2003]	Structural Testing	Data flow based unit Testing	No coverage criteria defined	✓	×
R. T. Alexander et al.[2004]	Mutation Testing	System Testing	Fault based Criteria	×	×
Y. Zhou[2004]	Structural Testing	System Testing	No coverage criteria defined	✓	×
M. Mortensen et al[2004]	Mutation Testing	System Testing	Context coverage	✓	×
W.Xu et al[2004]	Model Based Testing(Aspectual flow graph based Testing)	Unit Testing	Use case, polymorphic, transition coverage criteria	×	×
G.Xu et al[2004]	Structural/Code based Testing	Unit Testing	No coverage criteria defined	×	✓
D. Xu et al[2005]	Model based testing(State based testing)	Unit Testing	Conditional branch Coverage	×	×
W. Xu et al[2005]	Model based testing(UML diagrams(class diagram, aspect diagram and sequence diagram)	System Testing	Polymorphic and branch coverage criteria	×	×
C. V. Lopes et al[2005]	Structural /Code Based Testing	Unit Testing	No Coverage criteria defined	✓	✓
M.Mortensen et al[2005] not incl	Mutation Testing	System Testing	Def-use coverage	×	×
H.Rajan et al[2005]	Structural/Code based Testing	System Testing	No Coverage criteria defined	×	×
T.Xie et al[2005]	Structural /Code based Testing	Unit Testing, Integration testing	Aspectual Integration coverage	✓	✓
M. Badri et al[2005]	Model Based Testing(State Diagram)	Unit Testing	Multi aspect integration coverage criteria	×	×
P. Massicotte et al[2005]	Model based Testing(Collaboration Diagram)	Integration Testing	Transition coverage, sequence coverage, modified sequence coverage and multi aspect integration coverage criteria	✓	✓
P.Massicotte et al.[2005]	Model Based Testing(Collaboration Diagram)	Integration Testing	Transition coverage, sequence coverage, modified sequence coverage and multi aspect integration coverage criteria	✓	×
D. Xu et al[2006]	Model Based Testing	System Testing	branch coverage criteria	×	×
T.Xie et al[2006]	Structural/Code based Testing	System Testing	Aspectual Branch, interaction Coverage Coverage	×	✓
P. Anbalagan et al[2006]	Structural/Code Based Testing	Unit Testing	No coverage	✓	✓
P.Anbalagan et al[2006]	Mutation Testing	System Testing	Fault based criteria	×	✓
O.A.L.Lemos et al[2006]	Mutation Testing	System Testing	Fault based criteria	×	×
J. Zhao et al[2006]	Regression testing	System Testing	No coverage criteria defined	×	×
G. Xu et al[2006]	Regression testing	System Testing	No criteria defined	×	×
J.S.Baekken et al[2006]	Mutation Testing	System Testing	Crosscutting node criteria	×	×

O.A.L.Lemos et al[2007]	Code based Testing	System Testing	Def-use	×	×
T. Xie et al[2007]	Structural /Code Based Testing	Unit and Integration Testing	Aspectual integration coverage	✓	×
C. Zhao et al[2007]	Mutation Testing	System Testing	Fault based criteria	×	×
P. Massicotte et al.[2007]	Model Based Testing(Collaboration Diagram)	Integration Testing	Aspect integration coverage criteria	✓	✓
G. Xu et al[2007]	Regression testing	System Testing	No coverage criteria defined	×	×
W. Xu et al[2007]	Model Based Testing	State Based Testing	No criteria defined	×	×
D. Xu et al[2007]	Model Based Testing	Aspectual use case diagram	Use case, transition and state coverage criteria	×	×
I.G.Franchin et al[2007]	Code based Testing	System Testing	Pair-wise Def-use	×	×
P.Anbalagan et al[2008]	Mutation Testing	System Testing	Fault based criteria	✓	✓
F.C.Ferrari et al[2008]	Mutation Testing	System Testing	Fault based criteria	✓	✓
D.Xu et al[2008]	Model Based Testing	UML Class, Sequence and Aspect Diagram	Use case, transition and state coverage criteria	×	×
C.H.Liu et al[2008]	Model Based testing	State Based Testing	No coverage criteria	✓	×
O.A.L.Lemos et al[2008]	Code Based Testing	System Testing	Advice point cut coverage	×	×
C.Babu et al[2009]	Model Based Testing(UML Sequence Diagram)	Integration Testing	Transition coverage criteria	×	×
M.Harman et al [2009]	Search based Testing	System Testing	Aspectual branch coverage	✓	✓
R.M.Parizi et al[2009]	Random Testing	System Testing	No coverage criteria defined	×	×
R.Delmare et al[2009]	Mutation Testing	System Testing	Fault based criteria	✓	✓
M.Badri et al[2009]	Model based Testing	UML State Diagram	State coverage	✓	✓
A.Jackson et al[2009]	Mutation Testing	System Testing	Fault based criteria	✓	✓
O.A.L.Lemos et al[2009]	Code Based Testing	Integration Testing	PairWise Def-use	×	×
F.Wedyan et al[2010]	Code Based Testing	Integration Testing	Interaction coverage criteria	×	×
F.C.Ferrari et al[2010]	Mutation Testing	System Testing	Fault based criteria	✓	✓
D.Xu et al[2010]	Model Based Testing	Extended state based Testing	State coverage	✓	✓
D.Xu et al[2010]	Model Based Testing	State Based Testing	State coverage	✓	✓
F. Wedyan et.al.[2010]	Structural Testing	System Testing	Data flow coverage criteria	✓	×
A.Delmare et al[2011]	Mutation Testing	System Testing	Fault based criteria	✓	✓
R.M.Pairzi et al [2011]	Random Testing	System Testing	No Coverage	×	×
S. Madadpour et al[2011]	Model based testing	Integration Testing	Aspect-integration coverage criteria	×	×
R.Delmare et al[2012]	Integration Testing	Search Based Approach	No coverage criteria defined	×	×
R.Delmare et al[2012]	Integration Testing	Search Based Approach	No coverage criteria defined	×	×
P.Wang et al[2012]	Structural Testing	System Testing	No coverage criteria defined	✓	✓
M. Mahajan et al[2012]	Structural Testing	System Testing	No coverage criteria defined	✓	✓
C. Kaur et al[2012]	Model Based testing	Integration Testing	Aspect Integration	×	×
A.A. Ghani [2013]	Mutation Testing	System Testing	Semantic fault based criteria	×	×
F. G. Leme et.al.[2015]	Mutation Testing	System Testing	Fault based criteria	✓	✓

F. Wedyan et.al.[2015]	Structural Testing	System Testing	Data Flow Based coverage criteria	✓	×
S. Dalal et.al.[2017]	Model Based Testing	System Testing	Aspectual branch coverage criteria	✓	×
S. Dalal et.al.[2017]	Model Based Testing	Search Based Approach	Aspectual branch Coverage Criteria	✓	✓
S. Dalal et.al.[2017]	Model Based Testing	Search Based Approach	Aspectual branch Coverage Criteria	✓	✓

3.5. Automation level

Automated support is very important for testing as it reduces cost, effort, time and also minimize the human errors. Various automation level has been described by researchers [3]-[5]. On the basis of those, a broad classification of automation level has been shown in Figure 3.

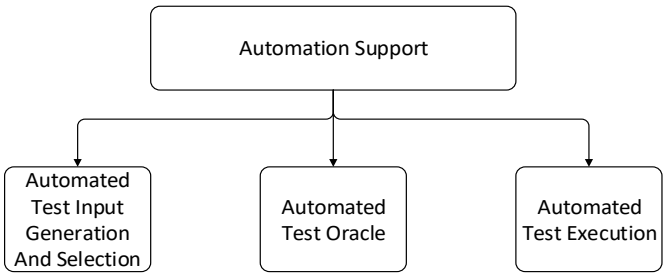

Figure 3. Classification of automation in AOP

3.5.1. Automated Test Input Generation and Selection Tools

These type of tools are used to produce test data which satisfies the particularized coverage criteria.

3.5.2. Automated Test Oracle

These are used to automatically examine the fidelity of the tests.

3.5.3. Automated Test Execution

This tool is used to execute the test and collects the test results with the help of automated test oracles.

4. ISSUES REGARDING AOP TESTING APPROACHES

There are following issues that could help and motivate the prospective researchers to do their contribution in this field.

4.1. Support of UML behavior model for Aspect-Oriented Testing Approaches

In aspect-oriented programming paradigm, software development and testing are different from other programming paradigms as it supports the concept of Separation of Concerns (SoC)[7], [8]. As observed in Table 4, only a few approaches have used the UML behavior models for testing Aspect-Oriented Programs. Efficient and effective test model can be designed by directly using the UML behavioral models. Designing specific test model for testing is a tedious and time- consuming task [8]. Usage of UML behavioral model to test AOP reduces the effort and increases reusability.

4.2. Test Case Coverage Criteria

Coverage criteria is a measure to decide which input sets are to be involved in testing. In traditional programming paradigms, coverage criteria have been proved so helpful. But in case of testing aspect-oriented programs, all coverage criteria related to traditional programming paradigms cannot be used directly as there is a concept of SoC [3]. Therefore, need to extend the coverage criteria for testing AOP [8]. Study of literature indicates that only three testing coverage criteria are defined [3].

4.3. Automation Support

Aspect-Oriented Programming is becoming mature day by day. Therefore, automation support for testing AOP is needed now. Automation does not only cut down the test effort but it also upturns the

efficiency and effectiveness of AOP testing process. Therefore, the future perspective of the researchers should be automated the AOP testing process.

4.4. Search-Based Testing

Table 4 shows that only three search-based testing approach has been proposed for AOP. So, one needs to focus on optimization approach to make the testing process effective and efficient.

4.5. Lack of Experimental Evaluation

There is a lack of experimental evaluation of empirical results on AOP testing approaches. In the 62 analyzed papers, most of the approaches have been applied only to small projects rather than the industrial environment. Those approaches have been developed for the specific purpose and do not get brought into the industry environment.

The systematic comparative analyzed partial results on AOP testing approaches has been presented in this paper due to space restriction. The detailed analysis results are available in [3], [5] and [6].If one has focused on only MBT approaches for AOP then the detailed analyzed results are available in [6], [7].

5. CONCLUSIONS AND FUTURE WORK

This paper caters a depth survey of literature on aspect-oriented software system testing approaches and also a comparative study of different AOP testing approaches. In this paper, different testing approaches for AOP have been characterized on the basis of some characteristics such as AOP testing techniques, level of testing, supporting tools, test case coverage criteria and automation level. A comparative analysis of available AOP testing techniques has been shown in Table 4. Important issues also have been discussed in this paper such as automation support, lack of optimization approach used for AOP testing, lack of AOP testing approach used in the industrial environment, support of UML behavioral model to test AOP etc. This paper helps the researchers to understand the depth of the work which has been done by other researchers and can also explore their knowledge on the subject.

REFERENCES

[1] Dalal S. and Chhiller R.S. A Novel Approach for Generating of Test Cases Based on Bee Colony Optimization and Modified Genetic Algorithm (BCO-mGA). *International Journal of Computer Applications*, Vol.68, No.19, Apr, 2013.
[2] Mohapatra S.K. and Prasad S. Test Case Reduction Using Ant Colony Optimization for Object-Oriented Program. *International Journal of Electrical and Computer Engineering (IJECE)*, Vol.5, No.6, pp.1424-1432, Dec.2015.
[3] Naqvi S.A. and Ali S. and Khan M.U. *An Evaluation of Aspect-Oriented Testing Techniques.* International Conference on Emerging technologies, Islamabad, Sept, 2005.
[4] Banik K. Investigation of Methods for Testing Aspect-Oriented Software. Master Degree Project, University of Skovde, 2013.
[5] Ghani A and Parizi R. Aspect-Oriented Program Testing: An Annotated Bibliography. *Journal of Software*, Vol.8, June 2013. Singhal, A. Bansal, Kumar A. A Critical Review of Various Testing Techniques in Aspect-Oriented Software Systems. *ACM SIGSOFT Software Engineering Notes*, Jul 2013.
[6] Hooda S, Dalal S, 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] Kitchenham B. *Procedures for performing systematic review*. Joint Technical Report Software Engineering Group, Department of Computer Science, Keele University, UK and Empirical Software Engineering, National ICT Australia Ltd.
[8] Sahoo P. and Mohanty J.R. Early Test Effort Prediction Using UML Diagrams. *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, Vol.5, No.1, pp.220-228, Jan 2017.
[9] Kande M. M, Kienzle J, Strohmeier A. *From AOP to UML: Towards an Aspect-Oriented Architectural Modeling Approach.* 8th Workshop on Aspect-Oriented modeling, Germany, 2006.
[10] Khan S.A. Nadeem A. *UML Extensions for Modeling of Aspect Oriented Software: A Survey*. National Software Engineering Conference (NSEC'10), Oct, 2010.
[11] Qaisar Z. H, Anwar K, Rehman S.U. Using UML Behavioral Model to Support Aspect Oriented Model. *International Journal of Software Engineering and Applications*, Mar, 2013.
[12] Xu W, Xu D, Goel V, Nygard K. *Aspect flow graph for testing aspect-oriented programs*. http://www.cs.ndsu.nodak.edu/wxu/research, 2004.
[13] Xu D, Xu W, Nygard K, "*A State-based approach to testing aspect-oriented programs.* Proceeding of the International Conference on Software Engineering and Knowledge Engineering, Jul.2005.
[14] Xu D, Xu W, Nygard K. *A State-based approach to testing aspect-oriented programs*. Proceeding of the 17th International Conference on Software Engineering and Knowledge Engineering, Jul.2005.

[15] Badri M, Badri L, Bourque-Fortin M. *Generating unit test sequence for aspect-oriented programs: Towards a formal approach using UML state diagrams*. Proceeding of the 3rd International Conference on Information and Communication Technology, Dec.2005.
[16] Massicotte P, Badri L, Badri M. *Generalizing aspects-classes integration testing sequences: a collaboration diagram based strategy*. Proceeding of the 3rd International Conference on Software Engineering Research, Management and Applications, Aug.2005.
[17] Massicotte P, Badri L, Badri M. *Aspects-classes integration testing strategy: an incremental approach*. International Workshop on Rapid Integration of Software Engineering Techniques, Mar.2005.
[18] Xu D, Xu W. *State based incremental testing of aspect-oriented programs*. Proceeding of the 5th International Conference on Aspect-Oriented Software Development, Mar.2006.
[19] assicotte P, Badri L, Badri M. Towards a tool supporting integration testing of aspect-oriented programs. *Journal of Object Technology*, Jan-Feb.2007.
[20] Xu W. *Testing aspect-oriented programs with state models. PhD Dissertation*, North Dakola State University of Agriculture and Applied Science, May 2007.
[21] Xu D, He X. *Generation of test requirements from aspectual use case.* Proceeding of the 3rd Workshop on Testing Aspect-Oriented Programs, Mar.2007.
[22] Xu D, Xu W, Wong W. E. Testing aspect-oriented programs with UML design models. *International Journal of Software Engineering and Knowledge Engineering* , 2008.
[23] Liu C. H, Chang C.W. A state-based testing approach for aspect-oriented programming. *Journal of Information Science and Engineering*, 2008.
[24] Babu C, Krishnan H. R. Fault model and test-case generation for the composition of aspects. ACM *SIGSOFT Software Engineering Notes*, 2009.
[25] Badri M, Badri L,Bourque-Fortin M. Automated state-based unit testing for aspect-oriented programs: A supporting framework. *Journal of Object Technology*, May-June 2009.
[26] Xu D, El-Ariss O, Xu W, Wang L. Testing aspect-oriented programs with finite machine. *Journal of Software Testing Verification and Reliability*,doi:10.1002/stvr.440,2010.
[27] Madadpour S and. Hosseinabadi S. H. M. Testing Aspect-Oriented Programs with UML Activity Diagrams. *International Journal of Computer Applications*, Nov.2011.
[28] Kaur C, Garg S. Testing Aspect-Oriented Software Using UML Activity Diagram. *International Journal of Engineering Research and Technology*, May 2012.
[29] Dalal S, Hooda S. Automated Test Sequence Generation of Aspect-Oriented Programs based upon UML Activity Diagram. *International Journal of Engineering and Technology*, Vol.9, No.2, May 2017.
[30] Zhao J. *Data-flow-based unit testing of aspect-oriented programs*. Proceeding of International Computer Software and Applications Conference, pp.188-197, Dec. 2003.
[31] Zhou Y, Ziv H ,Richardson D. *Toward a practical approach to test aspect-oriented software*. Proceeding of Workshop on Testing Component-based System, Sept.2004.
[32] Xu D, Xu W, Nygard K.*A State-based approach to testing aspect-oriented programs*. Proceeding of the 17th International Conference on Software Engineering and Knowledge Engineering, Jul.2005.
[33] Lopes C. V, Ngi T.C. *Unit _testing aspectual behavior.* Proceeding of the workshop on Testing Aspect-Oriented Programs, Mar. 2005.
[34] Rajan H,Sullivan K. *Generalizing AOP for aspect-oriented testing.* Proceeding of the 4th International Conference on Aspect-Oriented Software Development, Mar.2005.
[35] Xie T, Marinov J. Z. D ,Notkin D. *Automated test generation for AspectJ programs*. Workshop on Testing Aspect-Oriented Programs, Mar. 2005.
[36] Xie T, Zhao J. *A framework and tool supports for generating test inputs of AspectJ programs*. Proceeding of the 5th International Conference on Aspect-Oriented Software Development, Mar.2006.
[37] Anbalagan P, Xie T .*Automated pointcut testing for AspectJ programs: APTE*. Proceeding of the Workshop on Testing Aspect-Oriented Programs, Jul.2006.
[38] Lemos O. A. L, Vincenzi A. M. R J,Maldonado C ,Masiero P.C. Control and data flow structural testing criteria for aspect-oriented programs. *Journal of System and Software*, Vol. 80, 2007.
[39] Xie T, Zhao J. Per*spective on automated testing of aspect-oriented programs*. Proceeding of the 3rd Workshop on Testing Aspect-Oriented Programs, Mar.2007.
[40] Franchin I.G, Lemos O.A.L, Masiero P.C .*Pair-Wise Structural testing of object and aspect-oriented Java programs*. Proceeding of the 21st Brazilian Symposium on Software Engineering, 2007.
[41] Lemos O. A. L, Masiero P.C. *Integration testing of aspect-oriented programs: A structural pointcut-based approach*. Proceeding of the 22nd Brazilian Symposium on Software Engineering, 2008.
[42] Wedyan F,Ghosh S. *A Data Flow Testing Approach for Aspect-Oriented Programs*. IEEE International Symposium on High Assurance System Engineering, 2010.
[43] Wang P, Zhao X.*The Research of Automated Select Test Cases for Aspect-Oriented Programs.* Proceeding of the International Conference on Mechanical, Industrial and Manufacturing Engineering, 2012.
[44] Wedyan F, Ghosh S, Vijayasarthy L. R. An approach and tool for measurement of state variable based data flow test coverage for aspect-oriented programs. J*ournal of Information and Software Technology*, Vol.59, March 2015.
[45] Alexander R. T., Bieman J. M., Andrews A.A. T*owards the systematic testing of aspect-oriented programs.* Technical Report, 2004.
[46] Mortensen M, Alexander R.T. Adequate testing of aspect-oriented programs. Technical Report, Department of Computer Science, Colorado State University, USA, Dec.2004.

[47] Anbalagan P,Xie T. Au*tomated pointcut testing for AspectJ programs: APTE*. Proceeding of the Workshop on Testing Aspect-Oriented Programs, Jul.2006.
[48] Lemos O.A.L, Ferrari F.C, Maisiero P.C, Lopes C.V.*Testing aspect-oriented programming pointcut descriptors*. Proceeding of the 2nd Workshop on Testing Aspect-Oriented Programs, Jul.2006.
[49] Baekken J.S, Alexander R.T. *Towards a fault model for AspectJ programs: Step 1-pointcut faults*. Proceeding of the 2nd Workshop on Testing Aspect-Oriented Programs, Jul.2006**.**
[50] Zhao C, Alexander R.T. *Testing AspectJ programs using fault-based testing*. Proceeding of the 3rd Workshop on Testing Aspect-Oriented Programs, Mar.2007.
[51] Anbalagan P, Xie T.*Automated generation of pointcut mutants for testing pointcuts in AspectJ programs.* Proceeding of the 19th International Symposium on Software Reliability Engineering, Nov.2008.
[52] Ferrari F.C, Maldonado J.C, Rashid A. *Mutation testing for aspect-oriented programs*. Proceeding of the 1st International Conference on Software Testing, Verification and Validation, 2008.
[53] Delmare R, Baudry B, Le Traon L.*AjMutator: A tool for the mutation analysis of AspectJ pointcut descriptor.* Proceeding of the IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW 2009), Apr.2009.
[54] Jackson A,Clarke S. MuAspectJ: Mutant generation to support measuring the testability of AspectJ programs. Technical Report (TCD-CS -2009-38), ACM, Sept.2009.
[55] Ferrari F.C, Rashid A, Nakagawa E.Y, Maldonado J.C. *Automating the mutation testing of aspect-oriented Java programs*. In proceeding of the 5th Workshop on Automation of Software Test (AST'10), May 2010.
[56] Delmare A, Baurdy B, Ghosh S, Gupta S, Le Traon Y. An approach for testing pointcut descriptor in AspectJ. *Journal of Software Testing, Verifcation and Reliabilty*, 2011.
[57] Xu D, Ding J. *Prioritizing State-Based Aspect Tests.* Proceeding of International Conference on Software Testing Verification and Validation, 2010.
[58] Ghani A. A. A, Towards Semantic Mutation Testing of Aspect-Oriented Programs, *Journal of Software Engineering and Applications*, Vol.6, No.10A, Oct.2013.
[59] Leme F. G., Ferrari F. C., Maldonado J.C and Rashid A., *Multi-Level Mutation Testing of Java and AspectJ Programs Testing of Java and AspectJ Programs Supported by the Proteum/AJv2 Tool*,http://www2.dc.ufscar.br/~fabiano/pub_arquivos/Leme2015_cbsoftTools2015.pdf.
[60] Zhao J., Xie T. and Li N., *Towards a regression test selection for AspectJ Programs*, Proceeding of the 2nd Workshop on Testing Aspect-Oriented Programs, Jul.2006.
[61] Xu G.*A regression tests selection technique for aspect-oriented programs*. Proceeding of the 2nd Workshop on Testing Aspect-Oriented Programs, Jul.2006.
[62] Xu G, Rountev A. *Regression Test Selection for AspectJ Software*. Proceeding of the 29th International Conference on Software Engineering, 2007.
[63] Qamar M, Nadeem A, Aziz R. *An Approach to Test Aspect-Oriented Programs*. Proceeding of the World Congress of Engineering, London, U.K., 2007.
[64] Delmare R, Ghosh B, Ghosh S, Traon Y. *Regression Test Selection when Evolving Software with Aspects.* Proceeding of International conference of Aspect-Oriented Software Development, 2008.
[65] Parizi R. M, Ghani A.A.A, Abdullah R, Atan R. *Towards a framework for automated random testing of aspect-oriented programs.* In proceeding of the ISCA 18th International Conference on Software Engineering and Data Engineering (SEDE 2009), June 2009.
[66] Parizi R. M, Ghani A.A.A. *On the preliminary adaptive random testing of aspect-oriented programs.* Proceeding of the 6th International Conference on Software Engineering Advances (ICSEA 2011), Barcelona, Spain, and 2011.
[67] Harman M, Islam F, Xie T, Wappler. Au*tomated test data generation for aspect-oriented programs.* Proceeding of the 8th International Conference on Aspect-Oriented Software Development, 2009.
[68] Delmare R, Kraft N.A.*A Genetic Algorithm for Computing Class Integration Test Orders for Aspect-Oriented System.* Proceeding of the IEEE 5th International Conference on Software Testing, Verification and Validation, 2012.
[69] Mahajan M, Kumar S, Porwal R. Applying Genetic Algorithm to increase the Efficiency of a Data-Flow Based Test Data Generation. *ACM SIGSOFT Software Engineering Notes*, Vol.37, Sept 2012.
[70] Dalal S, Hooda S.A Novel Approach for Testing an Aspect-Oriented Software System Using Prioritized-Genetic Algorithm. *International Journal of Applied Engineering Research*, Vol.12, No. 21, 2017.
[71] Dalal S, Hooda S. *A Novel Approach for Testing an Aspect-Oriented Software System Using Genetic and Fuzzy Clustering Algorithm.* Proceeding on International Conference on Computer and Applications, ICCA 2017; Available at IEEE-Xplore Digital Library.
[72] Verma A. and Kaur S. Design and Development of an Algorithm for Prioritizing the Test Cases using Neural Network as Classifier ,*International Journal of Artificial Intelligence,V*ol.4,No.1,pp.14-19,March 2015.