DOI: http://dx.doi.org/10.26483/ijarcs.v9i2.5574

ISSN No. 0976-5697

Volume 9, No. 2, March-April 2018

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

ASPECT-ORIENTED SOFTWARE TESTING TECHNIQUES: A REVIEW

Dr. Sandeep Dalal,
Assistant Professor, Maharshi Dayanand University,
Rohtak, Haryana, India

Susheela Hooda
Research Scholar, Maharshi Dayanand University,
Rohtak, Haryana, India

Abstract: Software testing is a prominent area of research as it ensures the quality and reliability of the software. Testing an aspect-oriented software is also the latest area of research. Although, research in aspect-oriented software testing has reported a couple of new testing techniques. But, there is no work yet indicated in the literature as studied, to focus on the relationship of aspect-oriented software testing techniques with traditional software testing techniques. Therefore, this position paper handles this shortcoming and presents the basic concepts of aspect-oriented programming, phases of aspect-oriented software development life cycle and testing phases for new researchers to gather the information on the subject. This paper also presents comprehensive details about the relationship of aspect-oriented software testing techniques with other traditional testing techniques.

Keywords: Aspect-Oriented programming (AOP), Testing Phases, Aspect-oriented software development life cycle, Aspect-oriented programming paradigm (AOPP), Aspect-oriented software system (AOSS).

I. INTRODUCTION

Aspect-oriented programming (AOP) was started in1997 at Xeox's PARC laboratories with the introduction of the "AspectJ" programming language[1].The main purpose was to introduce AOP , to address the issues of the separation of crosscutting concern such as security, logging, transaction from the logic of the main application. According to Jacobson et.al. [2], concerns reflect the system requirements. Requirements are prioritized according to the user. For example, in a train control system, breaks of train can be a primary concern. Security and safety are considered as policy concern, performance and reliability consider as quality of service concern. Apart from primary concerns, all concerns come under crosscutting concerns.

In traditional programming languages such as procedural and object-oriented languages, functional and non-functional requirements are twine together. Programs are developed using the concept of abstraction. Main constructs of these traditional programming paradigm are procedures and classes. Each construct addresses the functional properties as well as non-functional properties. But, in case of aspect-oriented programming paradigm, application's logic is divided into two concerns: 1) Primary concern or functional properties and 2) Crosscutting concern or non-functional properties. This technique leads to advance the modular programming with a new effective means of software development. The goal of aspect-oriented programming is to enhance the concepts of reusability, maintainability, security etc. Large and complex applications can be developed easily with aspect-oriented programming paradigm (AOPP). So, AOPP coping with complexity and helps to achieve the quality software.

Figure 1 shows the concept of aspect-oriented programming by considering banking information system in which withdrawal transaction and transfer transaction are the primary concerns. These concerns are associated with the system's primary purpose. Authentication, authorization and logging are also considered as security requirements. These security requirements are consider as crosscutting concern.

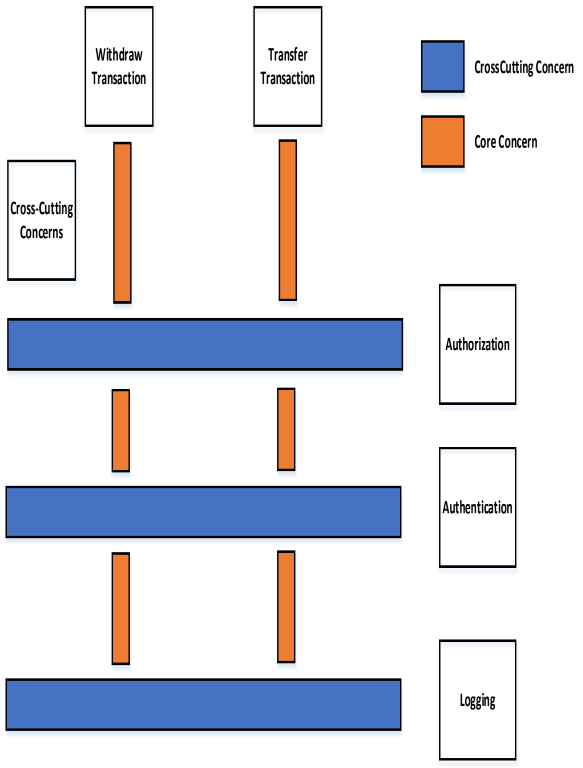

Figure 1 Concepts of Core and Crosscutting Concerns

II. ASPECT-ORIENTED SOFTWARE DEVELOPMENT PHASES

The main aim of aspect-oriented programming is to improve the modularity of the software system in order to make system easier and manageable. Therefore, aspect-oriented programming introduces an extra abstraction mechanism known as aspect. Various software development models have been used in the industry in order to satisfy the need of the business [3].Figure 2 shows the phases of software development life cycle (SDLC).Each phase of the SDLC performs certain prominent activity. But, in aspect-oriented programming (AOP), each phase of SDLC performs a different role.

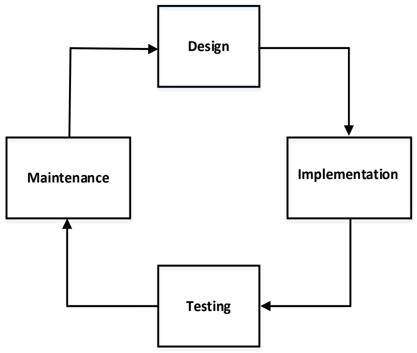

Figure 2. Software Development Life Cycle

Various role of AOP in each phase of SDLC has been described below [4]:-

2.1 Design phase:-

Design phase specifies the way to develop the software. Figure 3 shows the different activities to be performed in design phase of AOP.

- Crosscutting Concerns Recognition
- Initial Design Core Concern
- Crosscutting Concerns Designed

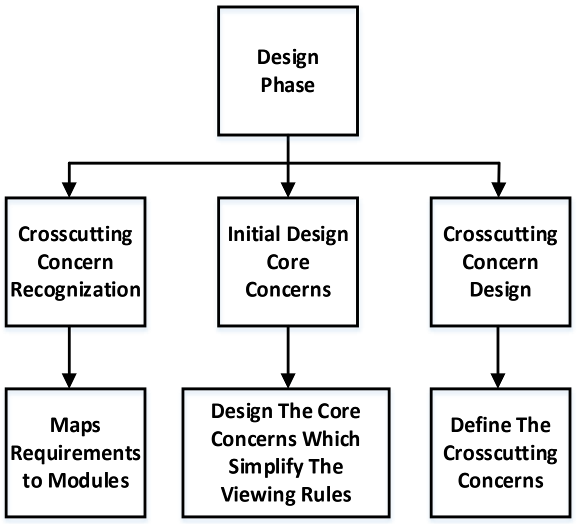
Design Phase
Crosscutting Concern Recognization
Initial Design Core Concerns
Crosscutting Concern Design
Maps Requirements to Modules
Design The Core Concerns Which Simplify The Viewing Rules
Define The Crosscutting Concerns

Figure 3. Activities of AOP in Design Phase

2.2 Implementation Phase

In the implementation phase, core and crosscutting concerns are implemented in aspect-oriented programming based languages such as AspectJ, AspectC++ etc. Almost all processes and methodologies are same between OOP and AOP in implementation phase. The following tasks have to be performed in AOP during the implementation of crosscutting concerns are:-

- **Join points identification**
  Identify the places in the code where crosscutting behaviour is needed.
- **Aspect Design**
  Aspects are designed using consistent patterns so that join points can be easily captured inside the aspects.

2.3 Testing Phase

After completing the design and implementation phase, testing phase is starting to execute the algorithm with test data to make sure that it has no logical errors. The most important activity is performed by the tester in AOP is to consider weaving process of aspects and imagine how a program will behave after weaving the aspects.

2.4 Maintenance Phase

This phase is considered as a most important phase in SDLC because much development effort goes toward maintenance. The following tasks are handle in this phase:-

- **Creating protection walls**
  This feature prevents the system from the new changes.
- **Implementing new features**
  In this task, new crosscutting concerns are implemented.

III. ASPECT-ORIENTED SOFTWARE TESTING PHASES

However, AOP increases the quality and reliability of the software but it does not take alone the responsibility to minimize the errors. Developers and programmers might make several types of mistakes in the program such as typological errors, wrong interpretation of user's requirements, prepare incorrect system design documents etc. Therefore, aspect-oriented programming paradigm could not be error free. Moreover, new constructs and features of aspect-oriented programming paradigm will introduce new types of faults. There should be a different mechanism to handle the AOP faults. Therefore, traditional testing techniques could not be directly applied to test aspect-oriented program. Henceforth, there should be a different approach to test AOP.

Various activities need to be performed while testing an aspect-oriented software system. These activities are categorized into three steps as shown below in Figure 4.

3.1 Creating Test Cases

Test cases of the program are created with little modification of the behavior.

3.2 Implementing Performance Testing

Many performance related problems occurred during the software deployment but these problems never faced during development phases. But one can able to handle these problems by using the dynamic profiling mechanism with AOP.

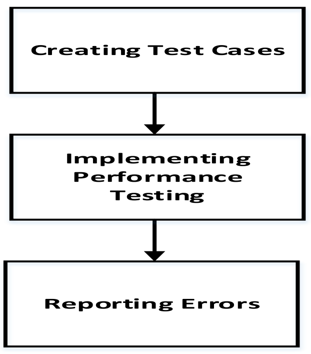

Figure 4.Aspect-Oriented Software Testing Phases

3.3 Reporting Errors

Aspects are used to collect the useful context. This flexible collection context can be used when problems may occur.

IV. TEST METHODS FOR ASPECT-ORIENTED SOFTWARE SYSTEM

In this section, various testing methods for aspect oriented software have been presented. Testing approaches are classified in order to make a clear understanding of testing approaches for AOSS. Figure 5 shows the classifications of testing methods for AOSS [5].

4.1 Data and control flow graph based testing

Data and flow graph are basically used to represent the structure of the software system. Thereafter, this graphical representation of the system is used to derive the test cases. The following testing methods are based on data and flow graph to evaluate the test cases for AOSS.

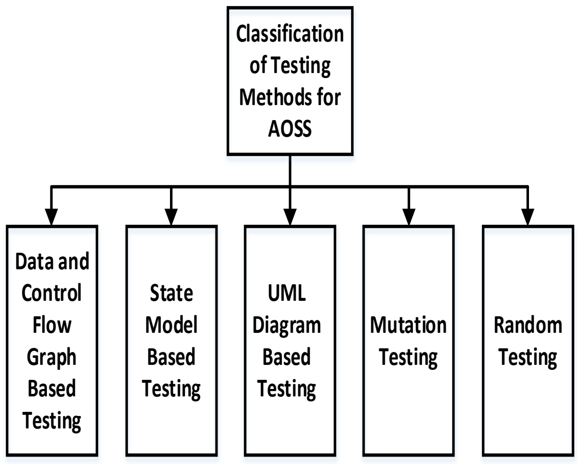
Classification of Testing Methods for AOSS
Data and Control Flow Graph Based Testing
State Model Based Testing
UML Diagram Based Testing
Mutation Testing
Random Testing

Figure 5.Classification of Aspect-Oriented Software System Testing Methods

Jhao[6] proposed a technique for selection of tests for two types of units(aspects and classes) for an aspect-oriented program which is based on control flow graph. This technique is used to compute the pairs of an aspect or class which is being tested. This technique is based on the concept of traditional graph based testing. A. Lemos et.al. [7] proposed a control and data flow model which supports structural testing to test unit of aspect-oriented program written in AspectJ. This paper has also proposed an approach for aspect-oriented programs by using data and control flow graphs. F. Wedyan et.al.[8] proposed a data flow testing approach for AOP in which various new data flow coverage criteria have been defined.

4.2 State model based testing

State models are used to represent the different states of the system and conditions allow to transit from one state to another. State based testing methods for AOSS are discussed below:-

W. Xu [9] proposed a testing approach based upon finite state machine to test aspect oriented program. Moreover, this approach was applied to various AOP problems to detect various faults in AOP. This approach identifies both kinds of faults; traditional faults and new faults which occur in AOP. C. H. Liu et.al.[10] also proposed a testing technique for aspect-oriented program which was based on object state diagram and constructed a weaving model of crosscutting concerns D. Xu et.al.[11] proposed a framework to test aspect oriented program. This framework also helps to detect many aspects fault. P. Wang [12] develop a tool to select test cases automatically. This approach applies to the basics of standard testing technology to select test cases.

4.3 UML diagram based testing

Test cases can be easily derived from the UML diagrams like activity diagram, collaboration diagram, state diagram etc. There are various testing techniques available for AOP. Some of testing techniques based on UML diagrams have been discussed below:-

M. Badri et. al.[13] proposed a technique based on UML state chart diagram to test unit individually for aspect-oriented programs(AOPs).Test cases are generated to find those faults which occur during the weaving process(Weaving means integrating aspects with classes).P. Massicotte et. al.[14] also proposed a technique for integration testing to integrate aspects with classes using collaboration diagram which covers various coverage criteria. In aspect-oriented program, integration of aspects with base classes. Moreover, a new kind of faults would occur during the integration of aspects with base classes. Therefore, P. Massicotte et.al.[15] proposed a new integration testing approach for aspect-oriented program. Although this approach is adopted from traditional integration testing but it completes its testing process in two steps: 1) test sequences are derived from the interaction of aspects with classes and 2) then verifies these test sequences. The proposed approach is based on collaboration diagram. One more incremental testing technique was also proposed by D. Xu et. al.[16] using the UML state diagram. This approach enhances the concept of reusability.

Madadpour et.al.[17] also proposed a AOP testing approach which is based on UML activity diagram. In this approach, aspects (cross cutting concerns) are integrated with base class (primary concern) and this approach helps to detect faults which commonly occurred during the integration. C. Kaur et.al.[18] was extended the work of [16]. S. Dalal et. al. [19] was developed a tool to automate the work to detect faults occurred during the integration process.

4.4 Mutation testing

Mutation testing takes the program and design various mutants by making small changes in the original program. Mutation based testing approaches for AOP as discussed below:-

R. T. Alexander [20] proposed a fault model to handle the errors which occur during weaving process of aspects with the base class. M. Mortensen et.al. [21] also proposed a framework for fault based testing for AOP which supported various coverage criteria like statement, insertion, context and def-use coverage.

C. Babu et.al.[22] proposed an approach to detect various types of faults which occurs during the composition of aspects in AOP. This paper also proposed a fault model which caters to identify faults in the earlier phases of software development life cycle process. This approach extends the concept of modularity used in object-oriented programming paradigm. A.A. Ghani et.al. [23] also proposed a semantic mutation testing for aspect oriented programs which mutates the semantic of the language in which the program is written. This approach is considered as a complementary over syntactic mutation testing types of faults.

C. Zhao et.al. [24] proposed an approach to test AspectJ program which is based on fault model. Furthermore, interaction model and dependency model help to derive fault model. F. C. Ferrari et.al. [25] proposed automated mutation testing for aspect-oriented programs where they designed the mutation operators using AspectJ programming language. This paper covers many traditional faults as well as aspect-oriented faults.

4.5 Random testing

Random testing generates the test cases randomly and it is very simple and less costly technique. Only a few papers have devoted to test AOP using random testing technique. R. M. Parizi et.al. [26] proposed a framework for random testing technique to test an aspect-oriented program. This framework caters to generate random input and selection strategy for AOP.

V. DEPENDENCY OF ASPECT-ORIENTED SOFTWARE TESTING TECHNIQUES OVER TRADITIONAL SOFTWARE TESTING TECHNIQUES

This section describes the relationship between the traditional approaches for traditional software development and proposed testing approaches of aspect oriented software. However, an aspect-oriented program (written in AspectJ) with classes, methods, packages and interfaces is similar to object-oriented program (written in Java). But, aspect weaving programs can affect the types of faults commonly found in traditional programming (i.e. object oriented programs and procedural programs).Clearly, AOPs contain some different faults. Therefore, fault handling mechanism could be different from traditional programming but somewhat similar. Therefore, according to Amman et.al.[27],four types of models 1)Graph based 2)Code based 3) Domain Reduction and 4) Syntactic Structure models have been used to develop the sequential software. Moreover, test methods are also categorized into four types based on the model on which software is based on.

5.1 Graph - based coverage

Graph-based coverage means the way to evaluate test set according to graphical representation of the software. Graph based coverage criteria is divided into two types:-

1) Control Flow Graph
2) Data Flow Graph

5.2 Code based coverage

Code based coverage criteria have increased in recent time. Logic expressions are derived from the decision points of the program, finite state machines, state charts and requirements.

5.3 Domain partitioning based coverage

Domain partitioning means to divide the input space based on the requirement. This technique can be applied to each level of testing like unit testing, integration testing and system testing.

5.4 Syntax based coverage

In syntax based coverage criteria, tests are derived from the syntax of the software. Syntax is based on the programming language‘s grammar rules in which program is developed.

Table 1 shows the dependence of aspect-oriented testing methods over traditional testing methods and. **(**The symbol “✓”=based on traditional methods, “× ”=not yet determined).

Table 1.Dependency of aspect-oriented software testing techniques over traditional software testing techniques

Reference	**Aspect-Oriented Testing Methods**	**Traditional Testing Methods**				
		Graph Based Testing		**Code Based Testing**	**Domain Reduction Based Testing**	**Syntax Based Testing**
		Data Flow Based Testing	**Control Flow Based Testing**			
J. Zhao[2003]	Data-flow-based unit testing of aspect-oriented programs	✓				
W.Xu et al[2004]	Aspect flow graph for testing aspect-oriented programs	✓	✓			
D. Xu et al[2005]	A State-based approach to testing aspect-oriented programs		✓	✓		
M. Badri et al[2005]	Generating unit test sequence for aspect-oriented programs: Towards a formal approach using UML state diagrams		✓			
P. Massicotte et al[2005]	Generalizing aspects-classes integration testing sequences: a collaboration diagram based strategy		✓			
D. Xu et al[2006]	State based incremental testing of aspect-oriented programs		✓			
O.A.Lemos et.al.[2007]	Control and data flow structural testing criteria for aspect-oriented programs	✓	✓	✓		
C.H.Liu et.al.[2008]	A state-based testing approach for aspect-oriented programming		✓			
C.Babu et al[2009]	Fault model and test-case generation for the composition of aspects					✓
M.Badri et al[2009]	Automated state-based unit testing for aspect-oriented programs: A supporting framework		✓			
D.Xu et al[2010]	Testing aspect-oriented programs with finite machine		✓			
F. Wedyan et.al.[2010]	A Data Flow Testing Approach for Aspect-Oriented Programs”	✓		✓		
S. Madadpour et.al.[2011]	Testing Aspect-Oriented Programs with UML Activity Diagrams		✓			
R.M. Pairzi et al [2011]	On the preliminary adaptive random testing of aspect-oriented programs	✗	✗	✗	✗	✗
C. Kaur et al[2012]	Testing Aspect-Oriented Software Using UML Activity Diagram		✓			
P. Wang et.al[2012]	The Research of Automated Select Test Cases for Aspect-Oriented Programs			✓		
A.A.Ghani et.al.[2013]	Towards Semantic Mutation Testing of Aspect-Oriented Programs					✓
S.Dalal et.al.[2017]	Automated Test Sequence Generation of Aspect-Oriented Programs based upon UML Activity Diagram	✓				

VI. CONCLUSION

This position paper briefly evaluates the significance of aspect-oriented software paradigm, software development life cycle, software testing phases and various testing methods of AOSS. This paper also caters to comprehensive analyze the relationship between aspect-oriented software testing methods and traditional testing methods. This paper helps the testers to choose a particular testing technique based on his /her requirements regardless of trying all testing techniques.

REFERENCES

[1] I. Sommerville, *Software Engineering*, PHI, 2010

[2] I. Jacobsen and Ng, "Aspect oriented Software Development with Use Cases", Boston: Addison-Wesley.

[3] S. Dalal and R. Chhiller, "Software Testing Three-P' Paradigms and Limitations," *International Journal of Computer Applications*, Vol. 54, No. 12, pp.49-54, Sept. 2012.

[4] D. Vulture, "Aspect-Oriented Programming in Design Phases", *Available at: https://www.todaysoftmag.com/article/1427/aspect-oriented- programming-in-design-phases.*

[5] K. Banik, "Investigation of Methods for Testing Aspect Oriented Software," University of Skovde, Master Degree 2013.

[6] J. Zhao, "Data-flow-based unit testing of aspect-oriented programs," *Proc. Of Conference On Computer Software and Applications Conference*, pp.188-197, Dec. 2003.

[7] O. A. L. Lemos, A. M. R. Vincenzi, J. C .Maldonado and P. C. Masiero, "Control and data flow structural testing criteria for aspect-oriented programs", *Journal of System and Software*, Vol. 80, Issue 6,pp.862-882,2007.

[8] F. Wedyan and S. Ghosh, "A Data Flow Testing Approach for Aspect-Oriented Programs", *IEEE International Symposium on High Assurance System Engineering*, pp.64-73, 2010.

[9] W. Xu, "Testing aspect-oriented programs with state models", PhD Dissertation, North Dakola State University of Agriculture and Applied Science, May 2007.

[10] C. H. Liu and C. W. Chang, "A state-based testing approach for aspect-oriented programming*", Journal of Information Science and Engineering*, pp.11-31, 2008.

[11] D. Xu, O. El-Ariss, W. Xu and L. Wang, "Testing aspect-Oriented programs with finite machine", *Journal of Software Testing Verification and Reliability*, Vol.24, Issue 4, pp.267-293.2010.

[12] P. Wang and X. Zhao, "The Research of Automated Select Test Cases for Aspect-Oriented Programs", in Proc. of the International Conference on Mechanical ,Industrial and Manufacturing Engineering, 2012. doi: 10.1016/j.ieri.2012.06.002

[13] M. Badri, L. Badri and M. Bourque-Fortin, "Generating unit test sequence for aspect-oriented programs. *In Proc .of the 3rd International Conference on Information and Communication Technology*, IEEE Computer Society Press,Cario,Egypt,Dec.2005.

[14] P. Massicotte, L. Badri and M. Badri, "Generalizing aspects-classes integration testing sequences: a collaboration diagram based strategy", *Proc. Of the 3rd International Conference on Software Engineering Research, Management and Applications,* IEEE Computer Society Press, Aug. 2005.

[15] P. Massicotte, L. Badri and M. Badri, "Aspects-classes integration testing strategy: an incremental approach",*2nd International Workshop on Rapid Integration of Software Engineering Techniques*, Lecture Notes in Computer Science,Sept.2005.

[16] D. Xu and W. Xu, "State based incremental testing of aspect-oriented programs", *Proc. of the 5th International Conference on Aspect-Oriented Software Development*, pp.180-189, Mar. 2006.

[17] S. Madadpour and S. H. M. Hosseinbadi, "Testing Aspect-Oriented Programs with UML Activity Diagram", *International Journal of Computer Science and Applications*, Vol.33, No.8, pp.4-11, Nov. 2011.

[18] C. Kaur and S. Garg, "Testing Aspect-Oriented Software Using UML Activity Diagram", *International Journal of Engineering Research and Technology*, Vol.1, Issue 3, May 2012.

[19] S. Dalal and S. Hooda, "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.

[20] R. T. Alexander, J. M. Bieman and A. A. Andrews, "Towards the systematic testing of aspect-oriented programs," Technical Report, Colorado State University, Fort Collins, 2004. .

[21] M. Mortensen and R.T. Alexander, "Adequate testing of aspect-oriented programs," Technical Report, Department of Computer Science, Colorado State University, USA, Dec.2004.

[22] C. Babu and H. R. Krishnan, "Fault model and test-case generation for the composition of aspects", SIGSOFT Software Engineering Notes, 2009. Vol. 34, No. 1, 2009, pp. 1-6. http://dx.doi.org/10.1145/ 1457516.1457521

[23] A.A.A .Ghani, "Towards Semantic Mutation Testing of Aspect-Oriented Programs", *Journal of Software Engineering and Applications*, Vol. 6, No.10 A,pp.5-13,2013.

[24] C. Zhao and R. T. Alexander, "Testing AspectJ programs using fault-based testing", in proceeding of the 3rd Workshop on Testing Aspect-Oriented Programs, Vancouver, 12-13 pp.13-16,Mar. 2007.

[25] F. C. Ferrari, A. Rashid, E. Y. Nakagawa and J. C. Maldonado, "Automating the mutation testing of aspect-oriented Java programs", in proceeding of the 5th Workshop on Automation of Software Test (AST'10), pp.51-58, May 2010.

[26] R. M. Parizi and A.A.A.Ghani,"On the preliminary adaptive random testing of aspect-oriented programs", in proceeding of the 6th International Conference on Software Engineering Advances(ICSEA 2011),Barcelona,Spain,Oct.2011

[27] P. Amman and J. Offut, "Introduction to Software Testing", Cambridge University Press, 2008. ISBN: 978-0-521-88038-1.