Automated test sequence generation of Aspect-Oriented Programs based upon UML activity diagram

Sandeep Dalal [#1],
Maharshi Dayanand University, Rohtak
Haryana,India
[1] sandeepdalal.80@gmail.com
Susheela Hooda [*2]
Maharshi Dayanand University, Rohtak
Haryana, India
[2] susheelahooda@gmail.com

***Abstract*— Nowadays, Aspect-Oriented Programming paradigm (AOPP) is getting a lot of popularity because it is capable to handle all concerns which are required for developing complex software. In order to develop complex software, AOPP provides new constructs such as join-points, pointcuts, advices etc. These constructs bring new types of programming faults such as incorrect strength of pointcuts, incorrect advice type and incorrect order of aspect precedence etc. In this paper, a tool is developed for generating the automated test case sequences which is based on UML integrated activity diagram and also verifies the execution of the selected test sequences. The proposed approach is based on several coverage criterion such as action path coverage criteria, modified advice type coverage criteria and incorrect –aspects precedence coverage criteria.**

Keyword- Aspect-oriented program, Coverage Criterion, Fault types, UML activity diagram, test sequence generation

I. INTRODUCTION

Aspect-Oriented software development (AOSD) is a programming paradigm which overcomes the serious limitations in modularizing concepts, in traditional programming paradigms. AOSD introduces new concept of separation of concerns, which can be designed and implemented separately. Therefore, "they improve the reusability, extendibility and maintainability" [1][2].

AspectJ is a widely used programming language for AOP which is an extension of Java, which brings some new concepts like joinpoint, pointcut, advice and aspect [5].Pointcut specifies various joinpoint and joinpoints are well defined points in a program such as a method call, invocation of a constructor. An advice contains code which is to be executed and an aspect is a crosscutting concern that helps to execute the core concerns (basic program) e.g. synchronization, security etc. are aspects which helps to execute the banking system(core functioning).

In software development life cycle, an effective testing is a very challenging task before the software testers. "It is generally presumed that software reliability and quality is directly depends on how thoroughly testing was performed on the software"[7]. An aspect oriented software test is more difficult than traditional and object-oriented programming languages It is generally believed that manual testing is not effective to test large and complex software product as it requires more effort and time for adequately testing. That's why testers go for automatic testing.

"But an automatic test case generation from source code of the program is difficult and incompetent, especially for large and complex programmed "[6]. So software tester are showing their interest to test software with UML models which are constructed during design process because UML models helps the software testers to perceive the system in desirable way and find test information only after single processing of models as compared to code[7].Model based testing also helps to detect various kinds of faults which are not easy to detect other testing techniques and it also allows to generate the test cases in the beginning of the software development which leads to save time and resources and helps to make the test plan effective.

Nowadays, testing a software with UML techniques, is getting popularity. "UML activity diagram is an important diagram among 13 diagrams supported in UML 2.0",[8] .It represents simple and easy flow of the logic of the system. In recent years, test case generation from UML activity diagram has been gaining some attention. But most of the work have been reported in the literature indicates that UML activity diagrams are being used to test an object-oriented software [6][9].But usage of UML activity diagram for testing AOSSs is still in its infancy stage. Cui [4] proposed an approach of modeling and integrating aspects with UML activity diagram in aspect-oriented. D. Mouheb et.al [3] presented formal specifications for aspect weaving in UML activity diagram. Souymaya et.al [5] proposed an approach to generate manual test cases using activity diagram.

In this paper we have been proposed an automatic test sequence generation approach using the work of [4].This approach considers various coverage criteria such as activity path coverage criterion, modified sequence coverage criterion and multi aspect integration coverage criteria. This approach reveals with various faults such as incorrect advice type, weak pointcut, strong pointcut and incorrect aspect precedence.

The remaining part of this paper is organized as follows: section 2 describes review of literature,section 3 contributes in presenting an overview of the integrated activity model as discussed in [4] and introduces some basic concepts. Section 4 proposed an approach for generating test sequences from integrated activity diagram (IAD).Section 5 shows an architecture of proposed approach and shows the experimental results. Section 6 concludes the paper and discuss the limitation of the paper and emphasis the directions for the future work.

II. LITERATURE REVIEW

D.Mouheb et. al. [3] proposed a formal specification (syntactic and semantic) and implementation strategy for aspect weaving into UML activity diagram. Results are also verified in the terms of semantics and algorithms with the help of web-based application.

Z. Cui et. al. [4] proposed an approach for modeling and integrating aspects with UML activity diagram and a prototype tool "Jasmine AOI". The two case studies have been implemented with this prototype tool.

S. Madadpour et.al. [10] proposed an approach which derive test sequences manually from the activity model. This approach is divided into three main phases such as 1) Building a primary model and generate test sequences. 2) Building aspect model and integrate with primary model and generate test cases and 3) verification of execution of test sequences.

R. T. Alexander et.al. [12] proposed a fault model and six coverage criterions such as incorrect strength in pointcut patterns, incorrect aspect precedence , incorrect focus of control flow, incorrect changes in control dependencies etc.

In this paper, an approach has been proposed to automate the test sequences from the integrated activity diagram (IAD). Alexender [12] proposed six test sequence coverage criterions .However, we have been generated test sequences based on four coverage criterion such as incorrect aspect precedence, incorrect advice type, strong pointcut and weak pointcut.

III. BASIC CONCEPTS

This section describes a couple of basic concepts and terms which have been using in the rest part of the paper. UML activity diagram describes the ordering of the activities and different notations have been using while designing activity diagram which are inspired from the flowcharts. Activity diagram can be used to model the aspect-oriented programs [4] by extending it with the integration of aspects. An activity diagram integrated with aspects, called Integrated Activity Diagram (IAD).

In this paper, use the work of [4] to design an Integrated Activity Diagram (IAD) and extending IAD by introducing some other nodes while automating the whole test case generation process which consider the following nodes:

- Basic Nodes: Consider the nodes which we need to show the normal activities in AOP without integrating aspects.
- Adaptation Nodes: Consider the insertion point (point cut) for injecting aspects in AOP .It can have three values: before, after and around.(consider only first two values)
- Aspect Nodes: Consider the nodes which we need to execute in response to the adaptation node.

- Advice Nodes: These nodes contain functions which can be executed in response to the adaptation node.

Figure 1 shows an integrated activity diagram (IAD) of auto vending machine application [5]. It consists of CCoinBox (primary class) and two aspects are integrated such as DrinkCheck and VendControl into primary class CCoinBox. CCoinBox class contains four basic activities which are being used to carry out the basic behavior of an auto vending machine [5].

1. addQtr (): This function is used to insert coins into the vending machine.
2. retrunQtr (): This function returns all coins by setting inner counter to zero.
3. reset (): To Initialize the vending machine .
4. vend (): provide enough drink to the customer and return extra coins back to the customer.
   Additionally, DrinkCheck aspect performs inventory controlling and VendControl aspect performs vending control.

IV. OVERVIEW OF PROPOSED APPROACH

The proposed approach consists of two phases as shown in figure 2. The first phase generates test case sequences corresponding to the decision-to-decision-graph or DD graph and second phase verifies various aspect specific faults.

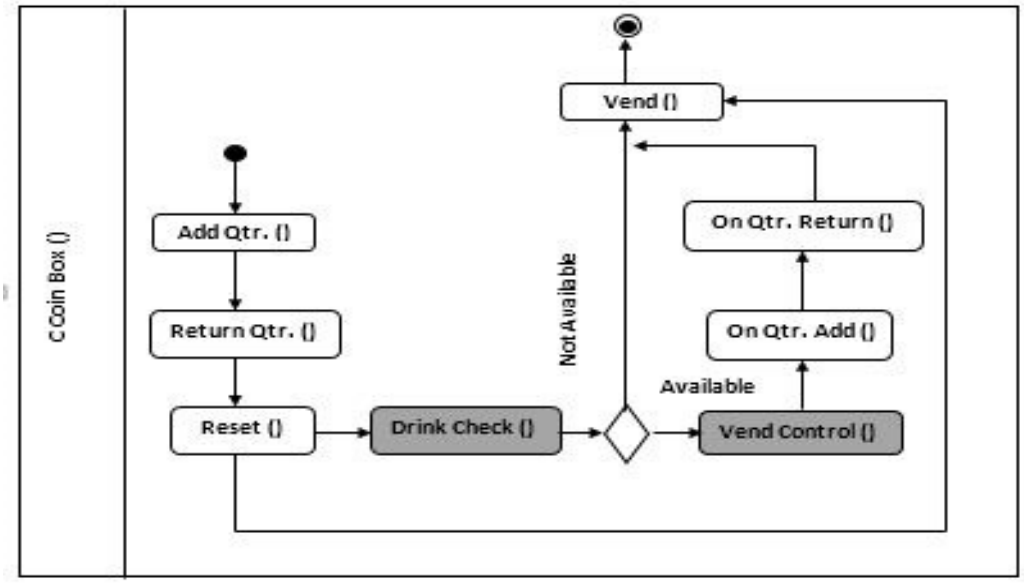

Fig. 1 An Integrated Activity Diagram of Auto vending machine

A. *Generation of test case sequences (Phase1)*

1. Generation of test case sequences (Phase1)

This subsection describes an approach (phase 1), to generate the automatic test case sequences which consist of three steps:

A. Modeling and an integrated aspects with UML activity diagram corresponding to AspectJ source code.[Zhanqi Cui]
B. Convert an aspect integrated activity diagram into control flow graph.
C. Generate test case sequences from control flow Graph.

A. Modeling and an integrated activity diagram corresponding to AspectJ source code.
Aspect model .Figure 1 shows the integrated activity diagram (IAD) of auto vending machine application.

B .Convert an aspect integrated activity diagram into Control flow graph

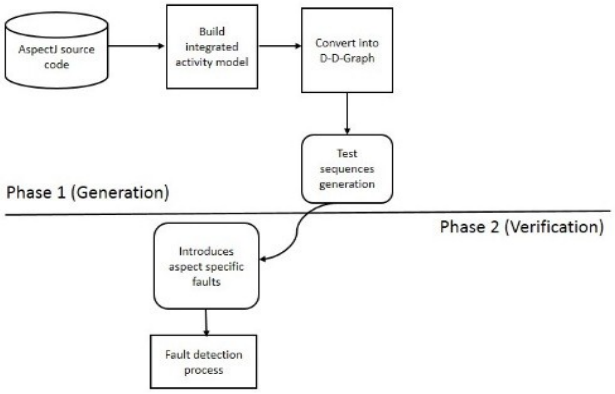

Fig. 2. Methodology of the proposed approach

C. *Generate the test case sequence using control flow Graph:-*

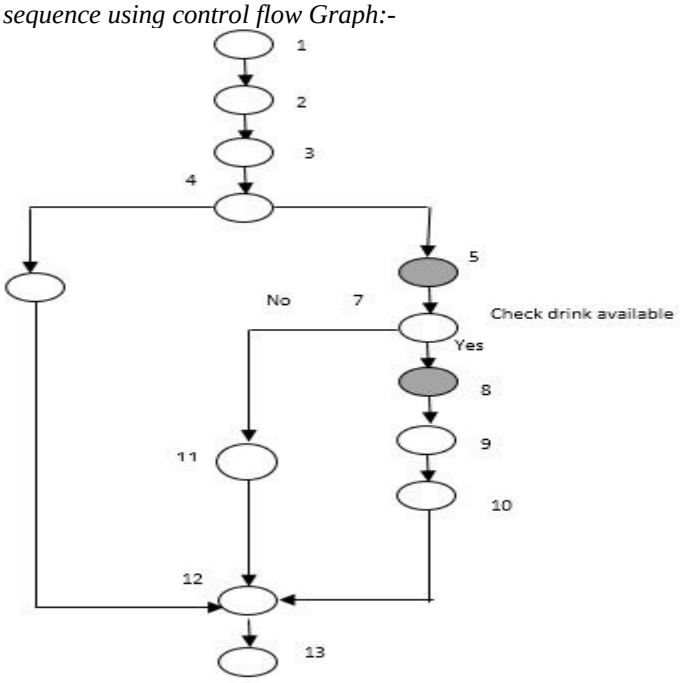

Fig.3 Control flow graph of Integrated Activity diagram of Auto-vending Machine Application

This subsection describes a method to generate test sequences using decision graph. The following steps represent the strategy in Fig.4.

1. Generate the control flow graph(CFG) corresponding to the Integrated Activity Diagram (IAD) with the help of matrix which store either value "1" or "0" respective to availability of edge between two nodes.
2. Generating the test sequences from control flow graph.
3. Test the test sequences entirely including aspects with the testing criterion (will be discussed in section 4.2)
4. end

Fig.4 Algorithm for generating test sequences using CFG of Integrated activity diagram

B. Testing coverage criteria

"Testing criteria is a rule or a set of rules that force conditions on testing game plans" [11].It is used to determine what should be tested regardless of how it is tested. Test cases can be derived from the fault model as discussed in [12].The first three criterion are based on the fault model [12] and fourth criteria is based on activity diagram.
In this paper, it is assumed that all primary concerns have been tested before weaving aspects and now our focus is to test the aspect-oriented programs after integrating the aspects.

B *.1 Incorrect aspect precedence coverage criteria*

Precedence means order of calling the advices when mutual dependent multiple aspects are integrated into primary concern which affects the overall behavior of the system.

"If the system behavior is affected by several mutually dependent aspects then the order of aspects must be tested. This test will test the sequences with the aspect precedence".

B.2 Incorrect advice type

Advice types are before, after and around which are pre-specified in an Aspect-oriented programs.

"This test will perform to test whether advices are calling according to specifications or not".

B.3 Incorrect strength pointcut

Pointcut contains the specifications which helps to select joinpoints of a particular type according to signature which includes in a pattern.

B.4 Action Path Coverage Criteria

It covers all non-concurrent (not parallel) activities from initial to end node in an integrated activity diagram (IAD) and also satisfy the priority relation "<" over a set of action sequence.

V. PROPOSED TOOL SUPPORTING OUR STRATEGY

An automatic test sequence generation tool has been proposed which also support the verification process. .NET technology, especially C#.NET technology is being using for implementing this Fig.5 presents the architecture of the tool and Fig.6 presents the main interface. This tool is illustrated with a case study discussed in section 4.

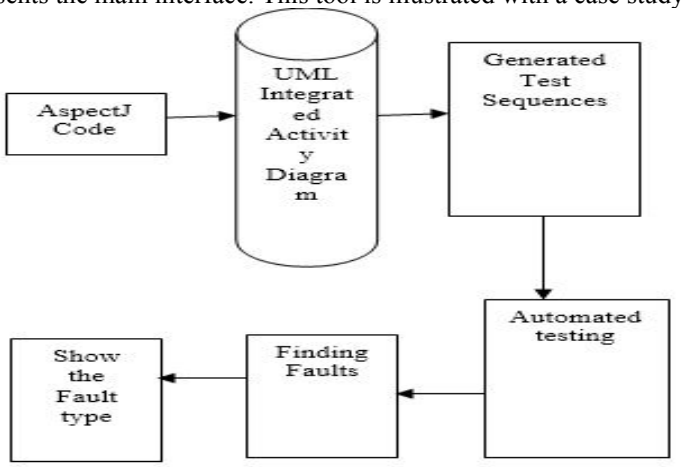

Fig.5 Architecture of proposed tool

The test sequence generation procedure is ground on the criteria discussed in the sub-section of IV .It is based on Integrated Activity Diagram(IAD).In the proposed tool, as exemplified section 3, will have eight test sequences. These sequences are being used while testing procedure is under gone. To test the generated test

sequences, introducing the faults [12] from fault model in the original sequences .The proposed tool show, in what follows, how the faults were detected.

VI. CONCLUSION

This paper proposed a technique to generate test sequences automatically from integrated activity diagram (IAD) which generates the test sequences from the control flow graph using transition coverage criteria, incorrect aspect precedence, incorrect advice type, incorrect strength of pointcut. A tool has been implemented which is based on proposed technique and has used it efficiently on an illustration. In future, planning to apply an evolutionary algorithm for optimizing test sequences.

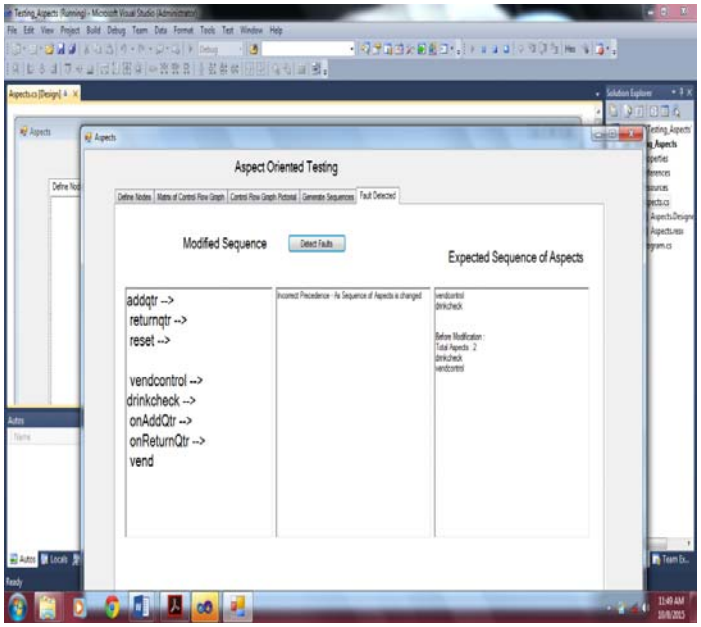

Fig.6 Interface of proposed tool

REFERENCES

[1] Aspect-oriented software development web site (AOSD), http://aosd.net/.
[2] T. AspectJ,"The AspectJ™ Programming Guide," 2002.
[3] D. Mouheb, D.Alhadidi and M.Pourzandi,"Aspect weaving in UML activity diagram: A semantic and algorithmic framework", 7th international conference on formal aspects of component software, 2010.
[4] Z.Cui, L.Wang and Xu. Li,"Modeling and integrating aspects with UML activity diagrams", ACM symposium on applied computing, 2009.
[5] C.H. Liu and C.W. Chang,"A state-based testing approach for aspect-oriented programming", journal of information science and engineering, 2008.
[6] M. Sarma ,D. Kundu and R. Mall, "Automatic test case generation from sequence diagram", 15th International conference on advanced computing and communication, pp 60-67,2007.
[7] D. Kundu ,D. Samanta, " A novel approach to generate test cases from UML activity diagram", journal of object technology ,Vol 8,pp. 65-83,June 2009.
[8] D.Pilone and N.pitman,"UML 2.0 in a Nutshell.O.Reilly", June 2005.
[9] A.A.A. Ghani and R.M. Parizi,"Aspect-Oriented Program Testing: An annonated Bibliography", Journal of Software, Vol.8, June 2013.
[10] S. Madadpur, S.Hassan and M. Hosseinabadi,"Testing aspect-oriented programs with UML activity diagram", Journal of Computer Application, 2012.
[11] P. Massicotte, M. Badri and L. Badri," Generating aspects-class integration testing sequences: A collaboration diagram based strategy", ACIS international conference on software engineering research, management and applications(SERA'05).
[12] R.T. Alexander, J.M.Bieman,"Towards the systematic testing of aspect-oriented programs", technical report CS-4-105,Colorado State University, Fort Collins, Colorado, USA, March 2004.

AUTHOR PROFILE

Dr. Sandeep Dalal, working as Assistant Professor in Maharshi Dayanand University, Rohtak, Haryana, India. He has more than 10 years experience of teaching and research. His areas of interest are software testing, data mining etc.

Ms. Susheela Hooda, working as Assistant Professor in B.S. Anangpuria Institute of technology and management, Faridabad, Haryana, India .She has more than 10 years of teaching experience. Her areas of interest are Software testing, cloud computing, Mobile Ad-hoc Network etc.