International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869, Volume-3, Issue-2, February 2015

# Automated Test Data Generation Process Through Path Testing For Aspect Oriented Programming

**Juhi Khandelwal, Pradeep Tomar**

***Abstract*— Aspect-Oriented Programming (AOP) is a new programming paradigm that mainly focuses on the cross cutting concerns and implements them in form of Aspects. However, these Aspects are hard to deal in many stages of Software Development Life Cycle (SDLC) especially in software testing. Testing main aim is to achieve high code coverage to improve the software quality. High code coverage requires lots of test data and it is not feasible to generate those test data manually. Test data generation process can be automated with the help of various techniques and framework. This paper provides review of some of the recent work that has been done in the area of AOP test data generation. Based on those work, this work propose an process for generating test data for AOP using Genetic Algorithm (GA).**

***Index Terms*—Genetic Algorithm, Automated Test Data Generation, Path Testing**

## I. Introduction

Software testing is an important part of SDLC. It is the only way to assure the quality of a system. Software testing is basically of two types: Black-box testing and White-box testing. Black-box testing only concentrates on the input and the output of the system and not on how the inputs are being processed in the system. On the other hand, White-box testing take cares of the structure of the code along with the input and output. Structure of the code is the logic that has been implemented in the source code. Thus, in White-box testing test cases tries to achieve maximum logic coverage, as it is inversely proportion to the number of bugs in the software. Code coverage is an approach to measure the extent to which a system has to be tested with a particular test suite. Basis path testing is the structural testing which tries to achieve the code coverage with the help control structure of the program.

It analyzes the control flow graph of the program to find a set of linearly independent paths of execution. Manually generating test data to achieve maximum code coverage requires lot of effort. At the same time it can also be error prone, tedious, costly, and time consuming. So in this case Automated Test Data Generation (ATDG) comes as a savior. Despite the importance of ATDG, there is still a lot to achieve in this area especially for AOP which is a new programming paradigm that is based on the concepts of Object-Oriented Programming (OOP). AOP implements cross-cutting requirements into units called Aspects.

Manuscript received February 20, 2015.
**Juhi Khandelwal**, Computer Science Department, Gautam Buddha University, Greater Noida, India.
**Pradeep Tomar**, Computer Science Department, Gautam Buddha University, Greater Noida, India .

Aspects encapsulate the functionality that cross-cuts and coexists with other functionalities in a system. AspectJ is the widely adopted AOP language. AspectJ is an extension to Java language. This extension includes aspects, pointcuts, advice, intertype declaration and jointpoints. Advice is a piece of code that is executed after or before of a certain action e.g. exits of a thread. Pointcuts are the point on which the cross cutting concerns are executed. So, a combination of an Advice and the Pointcut can be referred as Aspect.
This paper is organized as follows: Section 1 provides an introduction, section 2 presents a survey of various research papers on ATDG techniques and frameworks for AOP. Most of these techniques have concentrated more on the aspectual behavior of the AspectJ programs. Section 3 introduces an process for test data generation for AOP using GA to achieve path coverage. Section 4 presents the conclusion.

## II. Automated Test Data Generation Techniques and Framework

Software testing basically has three important steps: Designing or generating the test data. Executing those test data and analyzing the results as per the requirement [1].Test data generation in software testing is the process to identify a set of data which satisfies the given testing criterion [2]. In recent years, there is an upsurge in automating the test data generation process in order to reduce the cost and increase the quality of the system. This section presents a related work on ATDG of AOP.

Mark Harman et al. [3] have presented an approach for ATDG for AOP. This approach tries to achieve branch coverage and automate test data generation by using Search Based optimization technique i.e. evolutionary Testing. They have also performed an empirical study to support the claim that search based testing is effective for AOP. At the same time, they have introduced AOP Input - Domain reduction techniques to improve performance of ATDG. They have shown that evolutionary testing is superior to random testing. Fig1 [3] provides an overview of the process adopted for generation of test data.

They have used evolutionary testing to generate automated test data that is able to achieve Branch coverage. They have done empirical study of a suite of 14 AspectJ programs in order to validate their work. Results were quite impressive and support the fact that evolutionary testing achieve better branch coverage than random testing and also require less effort.

While performing Domain Reduction they have achieved up to 100% domain reduction which represent there is no set of relevant parameter. Even in this case evolutionary testing can be used to identify relevant public fields. Their various empirical study thus performed were in favor of their research. They have only studied the branch coverage. They have identified the

aspectual branch of the code and then identified the relevant parameter so as to reduce the input domain. They have used evolutionary testing to generate the test data so as to achieve maximum branch coverage.

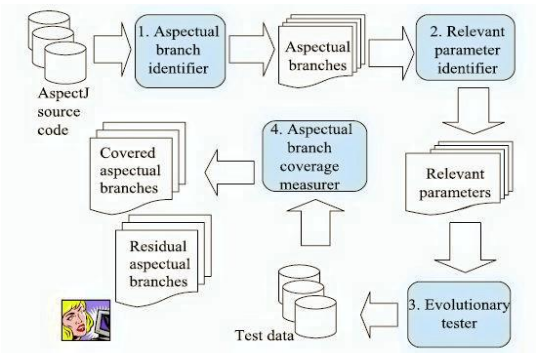


Fig1. ATDG using Evolutionary Testing

Anuranjan Misra et al. [4] have proposed a framework for ATDG by using evolutionary testing. They have also performed a comparison of evolutionary testing with random testing. Their framework was based on OOP framework. In their work they have concentrate on Aspectual Branches. They have used AspectJ programs as the input, which was converted in to Java programs.

They have provided an input domain reduction technique which involves in simplifying the constraints and starts with smallest domain of a variable to generate random inputs. Slicing was used to remove the irrelevant parameters. They have used 10 AspectJ programs to perform the empirical studies. In their empirical study they were able to support the fact that evolutionary testing is better than random testing in the terms of effort reduction and test effectiveness.

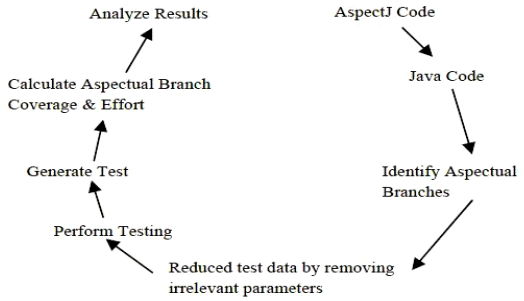


Fig2. Test Data Generation Framework for Evolutionary Testing

Fig2 [4] shows a flow chart of the technique suggested. In this chart An AspectJ code is first converted in to Java Code. In order to support their work they have calculated an Aspectual Branch coverage and effort. This work has supported the fact that evolutionary testing is better than random testing. They have used a basic technique of converting AOP to Java programs. Then they have generated automated test data for the branch coverage of the code and had adopted evolutionary testing.

According to Mourad Badri et al. [5] has suggested a unit testing technique called AJunit for aspect oriented programs. AJunit used dynamic behavior of classes and its related aspects as it is base. It generates test sequences as well as verifies them. A Java class can be related to one or more aspects, integrating one or more aspects to a class is very problematic. It is necessary to ensure that the original behaviors of classes are kept intact. In Unified Modelling Language (UML) statechart diagrams are prepared for the classes and their related aspects. In the next step related aspects are integrated in an incremental way. Statechart diagram generate the testing sequence which are further extended according to the behavior of related aspects. This extended chart is known as Extended StateChart (ESC) which represents the dynamic behavior of the class. Now this ESC model according to the testing criteria generates the testing sequences.

AJunit also have an advantage of retesting the specific part that has undergone some changes rather than retesting of the whole module. This approach also reduces the complexity of conflicts between aspects when they are integrated together. In this approach they try to identify the sequence that must be followed while testing. They have performed case studies in order to provide the detail of their approach.

Their approach not only generates the test sequence that has to be followed but it also supports its execution and verification. It also helps in doing the retesting and regression testing. Test cases are generated based on the testing sequence. Framework suggests the sequence of testing for state based unit testing i.e. a single aspect.

Tao Xie et al. [6] have presented a Wrasp framework for ATDG. It generates test data for both integration and unit testing. In their framework they generate a Wrapper class for the AspectJ programs and try to use the existing tools that are used for Java programs. Then these Wrapper classes are used for generation of test data rather than the woven base class.

For generation of integration test data two steps are followed, in the first step a wrapper class is generated for the base class and in the next step wrapper class act as a class under test for test generation techniques. For generation of Unit tests aspect classes are compiled using AspectJ compiler and later on these compiled classes are feed in to a test generation tool such as Parasoft Jtest for generation of test data. Jtest generates test data for the public methods.

Test generated through Parasoft doesn't provide the meaningful tests as the many features like aspect, advice are not covered in this. Their approach first generates a wrapper class of the AOP to address the weaving issues. Then generate test cases for this wrapper class using test-generation tools for Java.

Annasaro et al. [7] have provided a technique that generates sequence based test cases which are able to provide Optimum Code Coverage (OCC). Their technique is able to generate event based test cases by use of both Object Oriented and Aspect Oriented. They have compared there result with their previous work in which they have generated GUI based test cases. In their work they have used object oriented testing and aspect oriented testing to improve the code coverage.

In their approach they have firstly created the test cases based on the object oriented paradigm and later on aspect oriented test

case generation is used. Algorithms are used for both the processes. They have also used a pseudo code to calculate the code coverage. They have provided the analytical result to support their current work and have used the aspect and object oriented approach to generate the test cases rather than generating test cases for AOP. It takes account the interaction of aspects and the central system and has tried to address the mistake arise during the AOP testing. Their work has suggested an event based test case generation with the help of object oriented and aspect oriented event based test case generation. Then had compare their result with GUI based test case generation.

Reza Meimandi Parizi et al. [8] have suggested a framework that performs random testing of AOP. According to them many work that has proposed till now follows systematic testing of the AOP that is Test data was generated for structural testing. Randomness not only facilitates test generation but also influence the execution of test cases. Aspect Oriented Software Development (AOSD) contains some features and aspects that cannot be test with the traditional methods. AOSD provides new paradigm for software development by providing cross cutting concerns but it need to find out the techniques to verify and validate the software that uses AOSD approach.

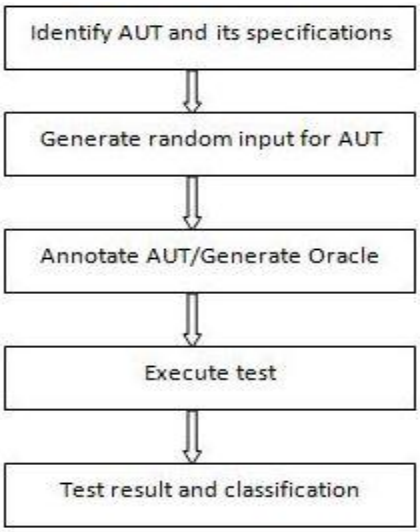


Fig3: Overview of Random Testing approach.

For generating reliable data random testing is the only option. Fig3 [9] represent the major five stages in their approach. Their approach starts with identifying the Aspect Under Test (AUT) and its specifications. They have used random testing to achieve the OCC.

## III. Automated Test Data Generation Process

In recent year, several researches have been done in the area of ATDG for AOP. All these approaches were for certain number of AOP programs. Most of these approaches have converted AOP programs to Java programs in order to use existing tools and techniques. These approaches have mainly concentrate on one perspective that is branch to achieve the maximum code coverage for AOP. But path coverage is better than branch coverage as it provide higher code coverage.

This paper proposed a technique for ATDG for AOP using GA for Path testing. According to Jin-Cherng Lin et al. [9] GA are able to search in a discontinuous space as well while searching methods can only work for continuous functions. GA is able to generate test data for discontinuous functions and most of the programs are discontinuous in nature. Following is the approach for the path testing of AOP using GA. In this the initial test cases are generated randomly using some existing automation tool. For implementing this approach, a fitness function is needed for identifying the test case for next level generation of test cases.

1. Generate Control Flow Graph of the source code.
2. Select the target path for path coverage
3. Generate test cases
4. Execute test cases and identify the test case that covers the target paths
5. If the target path is covered, output a successful message and go to step 10.
6. If the target path is not covered, identify the test case that survives the fitness function of GA.
7. Use that test case to generate next generation of test cases and execute the test cases to identify the test case that has covered the target path.
8. If the target path is covered and stop the GA identify the test case that has covered the target path and go to step 10.
9. After applying GA if the target path is not covered then iteration limit exceeds and output is a failure message.
10. Exit.

GA algorithm is based on Darwin theory of Survival of the fittest. It basically consists of 4 main stages: evaluation, selection, crossover and mutation. GA requires fitness function for selection of a single solution from a set of solutions. With the help of GA the test cases which have lower possibility of achieving the target path are discarded in the initial stages itself. Thus it helps in optimizing the results at the initial levels itself. Search based optimization techniques requires complex computation on the other hand GA greatest merit is their simplicity.

## IV. Conclusion

This paper has reviewed the research papers on ATDG of AOP and has found that most of the works concentrate on the branch testing and search based algorithms. Search based algorithms generate test data only for programs which are continuous in nature. With the help of the literature review, this paper has concluded that evolutionary testing is better than random testing in all prospects. Researchers and practitioners have used some set of already written AspectJ programs for ATDG. These AspectJ programs were converted in to Java programs with the

help of different techniques and have used some existing OOP tools for implementation. This paper has proposed a process using GA to generate the test data. GA is much simpler than search based algorithms in terms of testing. GA may get trapped in an infinite loop but the required path is already selected, so chances of being trapped are very low. As the numbers of iterations are also fixed, program will be executed in the limited time frame.

## REFERENCES


[1] Bogdan Korel, "Automated Software Test Data Generation", IEEE Transactions on Software Engineering, Vol.16. No.8, 1990.

[2] Hitesh Tahbildar and Bichitra Kalita, "Automated Software Test Data Generation: Direction of Research", International Journal of Computer Science & Engineering Survey, Vol.2, No.1, 2011

[3] Mark Harman, Fayezin Islam, Tao Xie and Stefan Wappler, "Automated Test Data Generation for Aspect-Oriented Programs", International Conference on Aspect-Oriented Software Development,2009.

[4] Anuranjan Misra, Raghav Mehra, Mayank Singh, Jugnesh Kumar and Shailendra Mishra, "Novel Approach to Automated Test Data Generation for AOP", International Journal of Information and Education Technology, Vol. 1, No. 2, 2011.

[5] Mourad Badri, Linda Badri and Maxime Bourque-Fortin,"Automated State-Based Unit Testing for Aspect-Oriented Programs: A Supporting Framework" ,Journal of Object technology Vol. 8, No. 3, 2009.

[6] T. Xie, J. Zhao, D. Marinov, and D. Notkin, "Automated test generation for AspectJ program", http://mir.cs.illinois.edu/marinov/publications/XieETAL05Wrasp.pdf

[7] Annasaro Vijendran and N.R.Suganya, "Generating Object-Oriented and Aspect-Oriented Sequence based Test Cases with Optimum Code Coverage", International Journal of Computer Applications (0975 – 8887) Volume 59, No.18, 2012.

[8] Reza Meimandi Parizi, Abdul Azim Abdul Ghani, Rusli Abdulla and Rodziah Binti Atan, "Towards a Framework for Automated Random Testing of Aspect-oriented Programs", 18th International Conference on Software Engineering and Data Engineering , 22-24, 2009.

[9] Jin-Cherng Lin and Pu-Lin Yeh, "Automated test data generation for path testing using GA", Information Sciences Journal 131, 47-64, 2001.