On the Preliminary Adaptive Random Testing of Aspect-Oriented Programs

Reza Meimandi Parizi, Abdul Azim Abdul Ghani
Department of Information Systems, University Putra Malaysia,
43400 Serdang, KL, Malaysia
{parizi, azim}@fsktm.upm.edu.my

***Abstract*— Adaptive random testing (ART) is a new family of random-based test data generation and selection strategies that enhances the effectiveness of tests over the classical random testing (RT). ART has been widely investigated and studied in numerous research papers over the recent years. These studies have included proposing various techniques for implementing and improving the intuition behind ART (evenly spread of test cases over the input domain, measured by some distance measures) generally for procedural programs with numerical input domain and most recently object-oriented programs. However, there is currently no work available in the literature that discusses the applicability of ART to aspect-oriented programming (AOP), as it is gaining popularity in software development. Inspired by this, this paper aims to investigate the possible ways that ART can be applied to AOP. This investigation focuses on a multi-perspective analysis of the current ART-based techniques. In this respect, we identified three related perspectives based on the current state of art in the area of ART. Each perspective was analyzed in terms of its applicability and possibility for aspect-oriented programs, particularly its constituent distance measure. As a result, our study gives rise to some interesting points and outlines a number of potential research directions in applying ART to AOP. This can pave the way for efficient development on applying of ART to AOP and finally AOP success.**

Keywords-software testing; random testing; adaptive random testing; aspect-oriented programming; aspect testing.

I. INTRODUCTION

Aspect-oriented programming [1],[2],[3] is one of the prominent modularization techniques emerged to cope with the complexity of software development process. To realize the benefits of aspect-oriented programming, the programs developed by this programming paradigm should be effectively tested. The reason is that the aspect-related defects [4],[5], stemmed from the unique characteristics of AOP, can affect the quality of these programs and consequently their general benefits, i.e., enhanced modularity and maintainability.

Software testing as the most widely used practice of ensuring the program's correctness, is useful to help finding these defects (i.e., their presence) and thus to provide a higher level of software quality. However, it has to be said that there is comparatively little work on testing of AOP in the literature and very little on automated testing of AOP such as [6],[7],[8]. This obviously indicates an insufficiency of testing approaches for the aspect-oriented programs at the current time and provides a primary motivation for leveraging the current testing techniques and/or developing new techniques for these programs.

Adaptive random testing proposed by Chen *et al.* [9] (as a recent derivative of random testing [10]) is an active and interesting research topic, which has shown [11],[12],[13],[14],[15] to have higher fault detection effectiveness compared to classical random testing, with facility of test automation. This is why Jaygarl *et al.* [16] has noted that ART is one of the most effective technique in automated test generation. The essential idea of ART techniques is that the evenly spread random test cases over the whole input domain allows finding faults through fewer test cases than with classical random testing. ART has shown to reduce the number of tests required to reveal the first fault by as much as 50% over classical random testing [17]. Adaptive random testing has seen remarkable progress during the recent past years in order to address the notion of evenly spread of test cases. It seems reasonable to conjecture that ART would continue to be active and become popular among the other random-based testing strategies.

In line with importance of AOP testing and on the other hands its current insufficiency, we believe the idea behind adaptive random testing can be worthwhile and attractive for automated testing of aspect-oriented programs since current research on testing of AOP, especially automated has not been adequately performed and is still in stage of infancy. In order to investigate the applicability of ART to AOP, we indentified three perspectives/directions based on scouring the current ART-based techniques in the literature. Corresponding to each perspective and its underlying technique (i.e., distance measure), we analyzed and discussed the feasibility of the given technique to AOP.

As far as we are aware, this is the first attempt made in the literature to discuss the applicability of ART for aspect-oriented programs. In other words, this paper takes some initial steps towards addressing the ART concept for automated test data generation and selection of the aspect-oriented programs. The specific contributions made by the paper are:

- It makes the current vague realization of ART to AOP more understandable by providing thought-provoking perspectives on this matter. Specifically, it gives a theoretical analysis and comparison of three known ART criterions adopted (presented under three identified perspectives) to calculate the distance among different test cases for aspect-oriented programs.

- It analyzes and potentially guides the application of ART in AOP and discusses the potential of using current ART techniques and their results to foster the development of new testing techniques in area of aspect-oriented software development (AOSD).

The remainder of this paper is organized as follows. Section II provides the background on ART and overviews the current state of the art in this field of research; Section III presents and analyzes the perspectives on adaptive random testing of AOP; Section IV summarizes the results of the analyses; and Section V reports the conclusion and future work.

II. ADAPTIVE RANDOM TESTING (ART)

A. *Overview and Classification*

Random testing [18],[10],[19] as one of the eldest techniques that include automated test input generation and selection has been studied and applied in different programming paradigms and application domains for decades. The first emergence of the random testing was meant for programs with numerical input domain, however with passage of time and emerging different paradigms the interest in random testing has been substantially increased due to the merits it offers. This matter is evident by various studies in the literature that have extended/applied the RT to the area of their interest.

Random testing is normally referred as the opposite of systematic testing such as functional or structural testing. The techniques in this family, i.e., random-based, can be generally classified into *classical/pure random testing* (the word classical and pure are interchangeability used in this paper) and *enriched random testing* due to the strategies they use for test input generation and selection, see Figure 1.

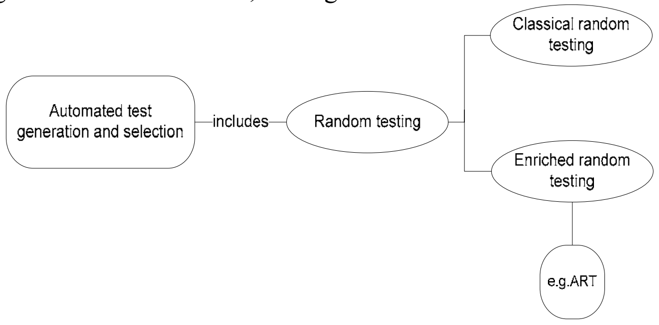

Figure 1. General classification of random testing techniques

By enriched, we mean those strategies that have been equipped with some guidance to their normal random generation process to pick up test inputs that give higher effectiveness in results, in contrast to the classical random testing in which test inputs are only picked at just random. In other words, both classical RT and enriched RT randomly generate test inputs from the input domain, but enriched RT uses additional guidance/criteria to help systematically test case selection rather than randomly selection. Note, in classical random testing test cases are generated by selecting random values of the input variables, which means the generation and selection are not two separated process but rather both imply each other and carried out randomly, see Fig. 2. (Note, in the classical RT, the test generation and test selection processes are the same but in the figure they have been separated for only the purpose of contrasting).

ART [9],[20] is the most dominant family of the enriched RT that suggests a selection criterion of "enforcing the test cases to be evenly spread over the entire input domain". Spreading evenly the test cases over the input domain is not only the basic idea underlying the ART but also Quasi-Random Testing (QRT) [21] and somewhat the Diversity-Oriented Test Data Generation (DOTG) [22]. These techniques emphasize on the idea of existence a correlation between the fault detection effectiveness and the evenness of the test case distribution in which the more even distribution of the test cases over the input domain the more fault detection capability with fewer test cases is gained.

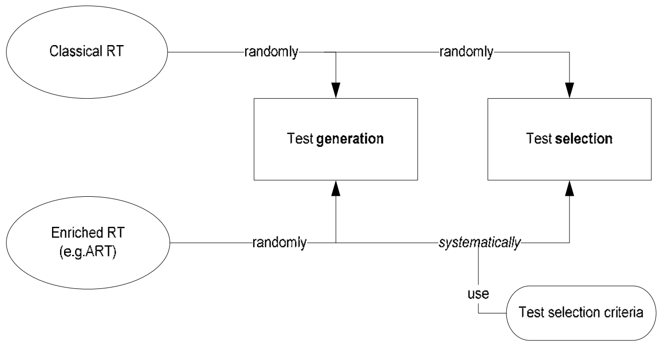

Figure 2. The contrasts between the classical and enriched random testing

In ART has been tried to enhance the fault detection effectiveness of classical RT by imposing some additional criteria on the test inputs selection process. As we mentioned before, the basic intuition of ART technique is that the evenly spread random test cases over the whole input domain allows finding faults through fewer test cases than with purely random testing. In literature several algorithms and variations of the techniques have been proposed to address the "even spread" intuition. The different ART algorithms give different test case selection criteria to ensure an even spread of the test cases. These algorithms attempt to maintain the benefits of random testing while increasing its effectiveness. For instance, one of the test case selection criterions used in one typical ART algorithm called the Fixed Size Candidate Set ART (FSCS-ART) [9] is as follows, which ensures the evenly spread of the test cases by means of a distance measure. The technique defines two test sets: the Executed Set, containing the test data that have been executed, and the Candidate Set, containing a set of randomly selected test data. The Executed Set is initially empty and the first test datum is randomly chosen. The Executed Set is then incrementally updated with the elements selected from the Candidate Set until a fault is revealed. The choice of the test datum from the Candidate Set requires the measurement of the distances of each candidate to all test data in the Executed Set. The chosen candidate is the datum that has the maximal value for the minimal distance among the distances to each test data in the Executed Set (furthest away from the already used inputs).

B. *State of the Art in ART*

Based on the idea of ART great deals of related algorithms, i.e., various implementation of the idea, have been proposed (distance-based ART, DART [23] was the first ART algorithm). The different algorithms give different test case selection criteria towards achieving this idea. Some of these algorithms are closely related to the ART, however, with slight changes. Example of these include the Restricted Random Testing (RRT) [24] or Ordinary Random Testing [25], while a plenty of them, as explained below, emphasize on the improvement to ART itself since its emergence [9].

Although ART has shown to be able to improve the fault detection effectiveness of RT, it requires additional computation overhead (considered as main problem associated with ART) to evenly spread test cases [26]. On this regard, a great deal of research has been proposed to minimize the boundary effect [27] and the overhead of primary ART algorithm. Mirror ART (called MART) [28], Fuzzy ART [13], ART by restriction [29], ART by localization [30], ART through dynamic partitioning [31], ART with CG constraints [32] are examples of these improvements which alleviate the pitfalls of the original ART algorithm, especially its overhead.

Further advancement to ART has also been provided by lattice-based ART. Lattice-based ART (L-ART) is a distinctive ART method that generates test cases by systematically placing and then randomly shifting lattice nodes in the input domain. The first introduction of L-ART [33] showed that L-ART is capable of yielding a better fault detection capability than RT, at the same generation cost. However, the test cases of L-ART may be highly concentrated on certain parts of the input domain and cause a *skewed distribution* of test cases. This skewed distribution of test cases can cause a tight coupling between the fault detection capability and the failure region location in the input domain. This means, when failure regions coincidentally reside in the area where L-ART selects a high density of test cases, L-ART may show a better fault detection capability than when failure regions are in the low density area. In reality, however, failure regions can be in any part of the input domain, therefore this dependency of fault detection capability on the failure region location is undesirable.

The issue of skewed test case distributions was addressed in an enhanced version of L-ART presented by Chen *et al.* [34]. The new L-ART not only had a less-skewed test case distribution, but also demonstrated better and more consistent fault detection capability compared to the original L-ART. This superiority of the fault detection capability of the new L-ART has been shown to be better than the results by Restricted ART by random partitioning [35], ART by bisection with restriction [36] and localization [37], ART through iterative partitioning revisited [38] and not revisited [39], ART with enlarged and high dimensional input domains [40], ART with randomly translated failure region [41], ART using Voronoi diagram [42], ART by balancing [43].

Distribution Metric Driven ART [44] has been conducted to measure how evenly an ART algorithm can distribute its test cases according to some distribution metrics such as discrepancy and dispersion, which reflect different aspects of the test case distribution. Discrepancy and dispersion are two commonly used metrics for measuring the equidistribution of sample points. Intuitively, low discrepancy and low dispersion, not in isolation, indicate that sample points are reasonably equidistributed [45] and finally implies an even spread of test cases. These distribution metrics have not only been used to measure and compare the equidistribution of various ART algorithms but also they have recently been adopted as criteria for the test case selection process aiming at improving the evenness of test case distribution and the fault detection capability of ART [45], [46].

More recently, a new family of ART [47] algorithms, namely adaptive random testing with dynamic non-uniform candidate distribution (ART-DNC) has been proposed. ART-DNC uses a new test profile called *failure driven* instead of uniform distribution or operational profiles used in the original ART algorithm to maximize the effectiveness of fault detection. These new algorithms showed better fault detection capabilities in contrast with the original ART and RT. Moreover, a new ART approach [48] based on the application of an evolutionary search algorithm, called Evolutionary Adaptive Random Testing (EART), was proposed lately.

As could be seen from above, there are so many different growing approaches that address the concept of ART and its further improvements. This matter may raise the question how the results of this work can be related to each other to come up with a completed and optimally effective ART approach. Recently, the work in [49] has taken into account this issue. This work presented a classification, amalgamation of the influential research work related to ART by highlighting the connections, and dependency relationships among the current work in this area.

The review of the current state of the art, as given in this section, shows that none of the presented work has discussed the applicability of adaptive random testing to AOP yet. This has primarily provided the motivation for the research in this paper to address this gap.

III. PERSPECTIVES ON ADAPTIVE RANDOM TESTING OF AOP

In this section, we present and discuss three perspectives on adaptive random testing of AOP. For each of the perspective, the discussion is based on the following:

- Its underlying technique and difference measure it encompasses
- Analysis (i.e., theoretical) of its applicability/ feasibility to AOP

A. Overview

It has been generally believed that how evenly an ART technique spreads test cases has an impact on how effectively it detects software failures, and an even distribution of test cases brings a good fault detection capability [11],[12],[13],[14],[15], [50]. However, this matter has only been proven for the numerical and recently objects input types, where there is no evidence on the other complex contexts such as aspect-oriented yet.

In order to be able to apply a typical ART technique (such as FSCS-ART) to a given program the following two issues should be generally figured out [49]:

(1) A strategy to help random sampling from the input domain of the program under test. In other words, this strategy is used to generate random test inputs/data.

(2) A mechanism to compare any two members of the input domain and determine the distance between them to select those test inputs that ensures the evenly spread of the test cases over the input domain. The distance measure should be able to represent the probability of common failure behavior

between two inputs. In other words, the distance measure can be viewed as a difference measure that tries to maximize the diversity of the inputs in which the smaller the distance, the more likely the test cases will show a similar failure behavior. Up to the present time, ART and its all variations in the literature are limited to programs with numeric inputs. On this regards, these studies have calculated the distance between two test cases, i.e., values from input domain, using the Euclidean measure.

Nevertheless, the first issue is common between any pure random testing and adaptive random testing techniques in which a given strategy needs to provide random generation of the test inputs (i.e., random testing). The second issue is meant to be only for ART techniques, i.e., solely unique to the adaptive random testing. It is worth mentioning that the first issue, which is RT, for different programming paradigms/languages and many application domains has been popularly resolved for decades, e.g., [51],[52],[53], [54]. In particular, there have been some recent attempts [55],[56] towards application of random testing to aspect-oriented programs, however the second issue has received lesser attention as the major challenge towards applying the concept of ART to AOP. Therefore, we place emphasis on discussing the second issue as the target objective in this paper.

The main question that we seek to provide insight into it is *how the concept of distance measure can be lifted or applied to aspect-oriented programs.* The answer to this question can consequently help developing adaptive random testing techniques towards automated testing of aspect-oriented programs.

According to the current evidence from literature, there are three perspectives in which this question can provoke discussion in the applying the notion of distance measure (second issue) or more generally ART to AOP. These perspectives are presented and discussed in the following sub-sections. Furthermore, in our discussion AspectJ [57],[58] is adopted as the target language. The reason is that the AspectJ is the most commonly used aspect-oriented programming language that warrants special attention.

B. Category and Choice-based Perspective

1) Underlying technique: This perspective is based on the concepts of categories and choices [59] to which the failure behavior of test cases (i.e., their ability to trigger faults) can be predicated according to the similarity of computation in the executions of them [49]. With regard to this idea, a difference measure (hereafter category and choice distance, CCD) for the category-partition method was first proposed by Kuo [60], who claimed that this measure can be used to help applying ART to a broad range of software input types.

The category-partition method is a specification-based testing approach. In this approach, the parameters and environment conditions that define the behavior of the program under test are first identified, which called as *categories*. Then, for each category, a set of mutual values that possibly triggers similar computation forms the *choices*. The more categories in which two inputs have various choices, the more diversifiable computation they trigger. Therefore, the number of categories containing differing choices is used as predictor of this difference measure, i.e., CCD.

In order to illustrate this difference measure, a simple object recognition system that is capable of distinguishing shapes, sizes and colors is presented as follows (taken from [49]). Suppose that the color of objects can only be light-red, red, deep-red, light-blue, blue, deep-blue, light-green, green and deep-green, and objects are spheres, cubes or pyramids in shape. The size is in the range (0,10] in m^3. The system behavior depends only on the object shape, the base color (i.e., red, blue or green), and whether the object is larger than 1 m^3. In this case, three categories can be defined: Color, Shape and Size; three choices for the Color category: red, blue and green; three choices for the Shape category: sphere, cube and pyramid; and two choices for the Size category: large and small. Some choices contain more than one possible value. For example, the red choice has light-red, red and deep-red as its possible values and large has any size more than 1 m^3. Consider two program inputs (i.e., test cases) T_1 and T_2, where T_1 is a light-red sphere of size 3.2 m^3, and T_2 is a deep-blue sphere of size 2.7 m^3. T_1 has the choices (red), (sphere) and (large) while T_2 has the choices (blue), (sphere) and (large). Therefore, there is only one category, color, in which T_1 and T_2 differ, thus the difference between the two inputs is 1 according to the given distance measure. This is to say that, these two tests are computationally similar as there is not much differences and thus might possibly have a similar failure behavior.

2) Analysis: The primary intension of Kuo [60] was to suggest the CCD difference measure as a generic metric for developing ART algorithms of non-numeric input types, but his primary work has not provided any practical example or case study to discuss this matter for modern programs such as object-oriented (OO) or aspect-oriented (AO). Thus, one might think of how this measure could be possibly generalized to these programs with non-numeric input types.

Following the same source of motivation that the CCD difference measure can be possibly applied to a broad range of program input types (as claimed by Kuo [60]), we have here analyzed its feasibility of the application to object- and aspect-oriented programs. To this end, we need to define what would be the categories and choices with respect to these programs and how truly they can represent the essential idea of ART.

In adoption of this measure to the object-oriented programs (as complementary to AOP), categories can be viewed as classes and their associated choices can be considered as instances of those classes, say objects. Therefore, the number of classes containing differing object's values would be a refined definition of the CCD measure for OO programs. Given this, recall the previous example (i.e., recognition system) and test inputs T1 and T2, we now assume this system is an object-oriented application containing three classes: Color, Shape and Size that does the same functionality but implemented in different programming paradigm, e.g., Java. In this case, we define three classes to represent the three categories, Color, Shape and Size respectively. Accordingly, three objects are instantiated to be as choices of the Color category that is red, blue and green. Likewise, three objects for the Shape category: sphere, cube and pyramid; and two objects for the Size category: large and small. According to the definition, there is only one class, color, in which T1 and T2 has different object' values, thus the difference between the two inputs is 1.

It can be said that the adaptive random testing of OO programs with respect to this category and choice-based measure (i.e., CCD) is possible to be performed. However, effectiveness of this measure would be another research effort that is worth further investigating.

Concerning the aspect-oriented programs, we now further assume that the recognition system example is an aspect-oriented application written in AspectJ that include the same classes as well as one more feature implemented in one aspect to keep track of the object's movement. The aspect is used to monitor the movement of the recognized objects to refresh the object's display whenever they actually move. Note, tracking movement of object is a crosscutting concern for the system, where it has been implemented as an aspect straightforwardly. If the aforementioned distance measure is chosen to be used for addressing the notion of evenly spread of test cases on this system, the only way to perform the adaptive random testing is to apply the given measure on the base code of the aspect-oriented program (by employing the aforementioned CCD for OO programs). The reason is that the aspects in most of AO languages (including AspectJ) do not have independent identity or existence in the system and cannot be instantiated. This articulates an aspect-related property known as *obliviousness* [61] in which objects, generally base code, are not aware of the aspects in the system. Consequently, such unique properties and characteristics related to AOP perhaps avoid adopting the categories and choices concepts to aspects, generally aspect code. (Typically, a given AO program such as AspectJ is comprised of two parts known as *base code* and *aspect code*. The base code contains all the classes and objects and provides the context execution (join points information) for the aspects. The aspect code contains all the existing aspects in the program and run based upon reaching certain join points in the base code. For more information on this please refer to [58]).

To sum up, we can state that the CCD measure is possible to be applied to adaptive random testing of AOP, however, it will not consider the direct testing of aspect code, specifically the aspect's constructs such as pointcuts and advice (as the focus is more on relationships between the affected/advised classes and aspects, i.e., base code). In this case, the tests mostly stress the integration between aspects and affected classes.

C. Object-based Perspective

1) Underlying technique: This perspective was inspired by two recent work on adaptive random testing of object-oriented programs. Since OO programs are considered as complementary parts to AO programs, thus the discussion regarding the prior application of ART to OO would be clearly helpful and connected to the objective of the paper, i.e., investigating the applicability of ART to AOP. Nevertheless, this work has been proposed for object-oriented programs written in Eiffel and Java languages, as briefly presented in the following.

a) ART for Eiffel: Ciupa *et al.* [17] propose adaptive random testing for object-oriented programs written in Eiffel, called ARTOO. Their approach initially share the idea of the DART approach [23] to select input objects (considered as test data/cases) from a testing pool. Since DART for object-oriented programs needs to calculate the distance between two arbitrary objects, accordingly they developed a new distance measure, *object distance* [62],[63] to be applied in adaptive random testing of OO programs. The proposed object distance was made up of the summation of three measure components namely elementary distance (i.e., the distance between the direct values of data types associated with objects), type distance (i.e., the distance between types of objects irrespective of object values), and field distance (i.e., the distance between matching fields of the objects). In addition to these three components, some weights and normalization were incorporated to the calculation process.

ARTOO is capable to automatically specify how to calculate the difference measure, however exponential calculation time, i.e., time complexity, imposed by increasing the dimension of the input domain is a major issue associated with object distance. For instance, checking the distance of integer type values are easier and quicker; however, calculating an object distance takes considerable much longer time (ARTOO takes 160% longer time compared to normal random testing [17]). Recently, in response to this issue, ARTOO has been further enhanced by Jaygarl *et al.* [16] for the purpose of more efficient testing of object-oriented programs. In this work, they suggested a simplified object distance that calculates object distance with lesser time complexity. They divided input data types into three categories− primitive types (including boxed types and a string type), array types, and object types. This separation was able to reduce unnecessary calculation of the ARTOO's object distance.

b) ART for Java: Lin *et al.* [64] propose a divergence-oriented technique to adaptive random testing of Java programs. The primary idea of this approach is to provide the program under test with a pool of test data each of which has considerable difference from the others (i.e., high divergence), and then to use the ART technique to select test data from the pool for the program under test. Unlike ARTOO that came up with a well-defined distance measure, this work employed only an intuitive divergence measure that was simply measured as distances of the objects in the pool, without providing any details about what this measure is and how it was calculated. This obviously makes the analysis of this measure's applicability to AOP difficult and therefore, it shall be excluded from the discussion in the analysis section in the following. Nevertheless, from an abstract point of view, since AspectJ is an AO extension of Java, the approach proposed by this work is likely to be applied to AOP, i.e., AspectJ programs. However, prior to that, a clear definition of the used distance measure along with further configurations to consider crosscutting constructs, e.g., advice and pointcuts, into the test generation process would be required.

2) Analysis: In the first place, one might think that the unique characteristics of AOP (including obliviousness property) can completely bar the notion of object distance (calculating the distance between two arbitrary objects) from applying to AOP and to some extent makes no sense of it, i.e., constructing difference measure between two arbitrary aspects is not feasible. The reason is that, contrary to the objects in object-oriented programs, in most of AOP languages such as AspectJ a given aspect does not have independent identity or existence in the system (i.e., the base code has no references to the given aspects) and cannot be instantiated. Note, in some special cases, it is possible to create several instances of a given aspect in AspectJ but by default, a unique instance of an aspect is only created and shared by all the objects when the application is launched. The aspect is then said to be a *singleton* [65].

However, it is important to note that it is just an instinctive misunderstanding. Because, in object-oriented programs (where the object distance was proposed for), the test data/cases to the programs are regarded as objects. Thus, in line with the idea of ART, measuring the distance between two objects would represent the difference between two test cases. Whereas, in the context of aspect-oriented programs it makes no sense to similarly measure the difference between two arbitrary aspects, while it should be between the tests for the aspects not aspects themselves.

Therefore, similar to the first perspective or specifically the category and choice-based measure (i.e., CCD), the object distance measure can only be used in the context of base code of the AO programs towards their adaptive random testing (i.e., the tests that stress the integration between aspects and affected classes). Because, the objects will form the base part of AO programs, i.e., base code.

It is also worth mentioning that, the object distance has an added advantage of requiring less effort compared to the first measure. This is why the object distance was originally developed and well-defined for OO programs, thus unlike the first measure no further effort would be required to leverage the underlying technique to OO programs, prior its application to AOP.

Finally, the explanations on the analysis of the object distance lead us to conjecture that the idea of the ART, using this measure, cannot be currently applied to aspect code of AOP (only base code). Hence, future research might include in-depth investigation of ART notion's applicability to AOP inspired by this measure, of course with a focus on adaptive random testing of aspects, i.e., aspect code. If one can figure out the feasibility or applicability of this matter then a metric model on top of object distance, as next step, will be required. This model should be designed in a way to capture an appropriate distance between arbitrary test cases (not aspects) for a given aspect under test to ensure the evenly spread of test cases (maybe "aspect distance" similar to its corresponding in object-oriented programs, object distance).

D. Coverage-based Perspective

1) Underlying technique: This perspective was motivated by some work related to coverage-based test case selection and prioritization [66],[67] in the context of regression testing. This work proposed methods to measure the distance between test cases based on coverage information such as statement and branch coverage, as presented below.

Zhou [66] proposes a metric, called the *Coverage Manhattan Distance (CMD)* as in (1), to measure the difference between any two arbitrary test cases, applicable to adaptive random testing. This measure uses the branch coverage information associated with the test cases. The formal definition of this measure is as follows. Given x as one test case, and E_x as a vector that records the branch coverage information related to x. The vector is defined to be $E_x = (x_1, x_2, \ldots, x_n)$, where $x_i \in \{0, 1\}$ for $1 \le i \le n$, and n is the total number of branches in a given program. The value of x_i is set to 1 if and only if the ith branch of the program has been exercised by execution of x; otherwise x_i is set to 0. Similarly, let y be another test case, and $E_y = (y_1, y_2, \ldots, y_n)$ records the branch coverage information of y. The Coverage Manhattan Distance (CMD) between x and y is captured by:

$$CMD\ (x, y) = \sum_{i=1}^{n} \left| x_i - y_i \right| \qquad (1)$$

Similar to the work by Zhou, Jiang *et al.* [67] suggested a distance measure based on the *Jaccard distance* of the two sets to be used as measured distance between two test cases. The Jaccard distance between two test cases x and y is defined as: $D(x, y) = 1 - |A \cap B| / |A \cup B|$, where A and B are the sets of the coverage of elements such as statements or branches exercised by x and y, respectively.

Empty-intersection set is a problem associated with Jaccard measure. That is, whenever the intersection between set A and B is empty the Jaccard measure just returns the maximum value of 1. This problem can result in capturing the distance between the test cases in a wrong way and consequently misguide the ART algorithm in picking the test case candidates (see [66] for example on this problem). However, this is not the case with CMD measure, whereas it is capable of yielding result that is more effective. This superiority led us to put emphasis on the CMD measure in the analysis of its capability to AOP, in the next sub-section.

2) Analysis: The two preceding measures, i.e., category and choice-based and object distance, focus on the input values (according to the program's input domain/space) as their sources of measurements. This dependency on input values makes these measures to be only applicable to certain types of programs (or at least more suited to some). On the contrary, CMD measure relies on a totally different source, which is independent of the input values. In our view, this measure is promising as it has the advantage (i.e., by using coverage information) that enables ART to be applied to a border range of programs with lesser limitations. In addition, the coverage fulfillment has been the most analyzed and required test criterion through the testing studies, which CMD has also taken into account.

In adoption of this measure to AOP, towards the ART, there can be two interesting ways of further exploration:

First, we suggest including the *aspectual branch coverage* [8] instead of the traditional branch coverage in the original CMD measure to record the required coverage information. Aspectual branch coverage is a coverage metric that captures the aspectual behavior, specifically the branch coverage within the aspect code (i.e., including branches from predicates in advice and methods in aspects). This metric has been previously used to guide the test generation in area of AOP testing [8],[6]. As a result, the selection of the test cases according to this adopted CMD measure (one may call it *Aspectual Coverage Manhattan Distance*, ACMD) would be based on test cases that are able to cover new aspectual branches that have not been covered by the previous executed test cases.

In order to make the point clear, a simple example showing the applicability of the coverage Manhattan distance to an aspect code is presented below. Given the aspect `ODRuleAspect` shown in Figure 2 (adapted from AspectJ examples by Laddad [58]):

```
public aspect ODRuleAspect
  pointcut debitExecution(Account account,  float
  withdrawalAmount)  : execution(void
  Account.debit*(float)  && this(account) &&
  args(withdrawalAmount);
  before(Account account, float withdrawalAmount)
  : debitExecution(account, withdrawalAmount) {
    Customer customer = account.getCustomer();
    if (customer == null) return;
    if (account.getAvailableBalance()>
  withdrawalAmount){
    float deductedAmount =
     account.getAvailableBalance()-
    withdrawalAmount;
    ...
     } else System.out.println("not enough
  money!");
    }
    ...
   }

public class Account {
  private float balance;
  private int accountNumber;
  private Customer customer;
  public Account(int accountNumber,Customer
  customer) { ... }
  public void debit(float amount) { ... }
...
}
```

Figure 2. An AspectJ example

In this case, there are two predicates (surrounded by a red box in Figure 2) which result in four aspectual branches in the given aspect, that is $n=4$. Suppose x and y are two test cases, where each of which contains a different instance of `Account` class, say `Ac1` and `Ac2` respectively. In addition, two calls to `debit` method (plus two parameter values for method's calls) on these instances are required to trigger the execution of the advice. Thus, for instance `Ac1.debit (95.60)` and `Ac2.debit (64.35)` would form the test cases x and y respectively. Assume, `Ac1. getCustomer` will return *null*, in this case x would be able to exercise only one branch, i.e., `customer == null`, hence $E_x = (1,0,0,0)$. Similarly assume, `Ac2. getCustomer` has not returned *null* and its `Ac2.getAvailableBalance` is 120 (which is higher than 64.35). Thus, the test case y is able to exercise two branches, i.e., `customer ≠ null` and `(account.getAvailableBalance()> withdrawalAmount)`, so $E_y = (1,1,0,0)$. Now, recall the metric in (1) the difference measured between these two cases would be of 1.

Alternatively, in order to obtain the proper coverage information to make use of the CMD measure in ART of AOP, we suggest employing the program's control flow graph of aspect-oriented programs. For this purpose, *aspect-oriented control flow graph* (AOCFG) proposed by Parizi *et.al* [68] (or other similar approaches such as [69]) would be a capable choice to help testers gain coverage-related information. This type of structural modeling and graph embodiment of aspects not only allows obtaining information related to the branch coverage but also a variety of coverage elements such as node, edge, etc. However, further research needs to be done to study the usefulness of these types of coverage information for ART, including coverage of elements in graphs/models used in aspect-oriented modeling.

In summary, the above analysis demonstrates that it is possible to construct more meaningful distance measure (using the idea of coverage information) in compared with the other presented measures for adaptive random testing of aspect-oriented programs. However, it still requires conducing further research to produce a well-suited coverage-based ART technique for aspect-oriented programs and then to proof the effectiveness of the produced technique through experimentation or proper case study.

IV. SUMMARY OF ANALYSES

For the brevity, a summary of the presented perspectives along with the analyses of the distance measure's properties, are presented in Table I.

TABLE I. SUMMARY OF THE DISTANCE MEASURES OF DIFFERENT PERSPECTIVES

Perspective	Distance/ difference measure	Source of measurement	Original Paradigm/ Application domain	Applicability to AOP
Category and choice-based	Category and choice distance	Input values	Procedural programs (with numerical inputs)	Base code
Object-based	Object distance	Input values	Object-oriented programs	Base code
Coverage-based	Coverage manhattan distance	Structural information (e.g., branch coverage)	Procedural and object-oriented programs	Base & aspect code

With respect to above table, the first column lists down the reviewed perspectives. The second column gives the original distance measure provided by the corresponding perspectives. The third, presents the source from which the measurement of the given measures are captured. The forth column lists the programming paradigms/application domains that the given measure were first proposed or applied to. Finally, the fifth column gives the possible applicability of the distance measures in terms of their suitability to adaptive random testing of aspect-oriented programs.

From the table, it can be clearly seen that only one measure, i.e., CMD, has the capability of being adopted to both base and aspect code, generally the whole AO program. Furthermore, the source of measurement used by this measure, it is more fine-grained and desirable compared to the other two measures.

Nevertheless, based on the theoretical analysis and interpretation shown among different perspectives and their distance measures and the fact that these measures are capable of providing different level of adoptability to AOP (i.e., relative advantages and weakness), at the moment and based on our understanding of these reviewed perspectives, the coverage-based perspective, to be exact the CMD measure, proposed by Zhou [66] shows to be one of the most suited (with respect to the unique characteristics of AO programs) and promising distance measure towards adaptive random testing of the aspect-oriented programs.

V. Conclusion and future work

Research on automated AOP testing is quite young and there is still a way to grow to its maturity. In ambition to advance the work with test automation of AOP and reaching to a plausible maturity, we have performed some preliminary research to investigate the applicability of one of the current automated test generation and selection techniques (i.e., ART) to AOP. The given investigation included the identification and presentation of the three related perspectives (by comparing their enclosed distance measures) on adaptive random testing of AOP and their general limitations and applicability.

As a general conclusion, our study shows that it is possible to apply the ART technique to AOP, however the current distance measures would not be all applicable or sufficient to address the notion of evenly spread of test cases suggested by ART. Two of the measures were intended to be only applicable to base code of AO programs while one was more applicable in nature, having potential of calculating distance between test cases meant for aspect code. Thus, aspect-oriented programs require evolving the discussed measures and/or developing new effective distance measure that can truly represent the notion of evenly spread of test cases with regard to the unique characteristics of these programs.

At last, we believe the work presented in this paper has provided new avenues of exploration within the area of AOP testing. Decidedly, this would be only the initial stage of leveraging a well-known testing technique to AOP; hence, it still requires further research to establish a concrete and useful ART-based technique for AOP in the future.

Acknowledgment

The authors acknowledge the support of the Malaysian Ministry of Higher Education for supporting this research (Fundamental Research Grant Scheme Phase 2/2010 (FRGS/2/2010/SG/UPM/01/2)).

References

[1] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin, "Aspect-Oriented Programming " in *Proceedings of the 11th European Conference on Object-Oriented Programming* 1997, p. 220−242.

[2] G. Kiczales, J. Lamping, C. V. Lopes, J. J. Hugunin, E. A. Hilsdale, and C. Boyapati, "Aspect-Oriented Programming," in *United States Patent 6467086*: Xerox Corporation, 2002.

[3] A. Colyer and A. Clement, "Aspect-Oriented Programming with AspectJ," *IBM systems journal,* vol. 44, p. 301−308, 2005.

[4] R. T. Alexander, J. M. Bieman, and A. A. Andrews, "Towards the Systematic Testing of Aspect-Oriented Programs," Colorado State University 2004.

[5] F. C. Ferrari, J. C. Maldonado, and A. Rashid, "Mutation Testing for Aspect-Oriented Programs," Proceedings of the 1st International Conference on Software Testing, Verification, and Validation, 2008, p. 52−61.

[6] M. Harman, F. Islam, T. Xie, and S. Wrappler, "Automated Test Data Generation for Aspect-Oriented Programs," in *Proceedings of the 8th International Conference on Aspect-Oriented Software Development*, Charlottesville, Virginia, USA, 2009, p. 185−196.

[7] T. Xie, J. Zhao, D. Marinov, and D. Notkin, "Automated Test Generation for AspectJ Programs " in *Proceedings of the 1st Workshop on Testing Aspect-Oriented Programs*, 2005, p. 1−6.

[8] T. Xie and J. Zhao, "A Framework and Tool Supports for Generating Test Inputs of AspectJ Programs," in *Proceedings of the 5th International Conference on Aspect-Oriented Software Development*, 2006, p. 190−201.

[9] T. Y. Chen, H. Leung, and I. K. Mak, "Adaptive Random Testing," in *Proceedings of the 9th Asian Computing Science Conference*, 2004, p. 320−329.

[10] R. Hamlet, "Random Testing," *Encyclopedia of software Engineering,* p. 970−978, 1994.

[11] J. Mayer and C. Schneckenburger, "An Empirical Analysis and Comparison of Random Testing Techniques," in *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, Rio de Janeiro, Brazil, 2006, p. 105−114.

[12] T. Y. Chen, F.-C. Kuo, and R. G. Merkel, "On the Statistical Properties of the F-measure," in *Proceedings of the 4th International Conference on Quality Software*, 2004, p. 146−153.

[13] K. P. Chan, T. Y. Chen, and D. Towey, "Good Random Testing," in *Proceedings of the 9th Ada-Europe International Conference on Reliable Software Technologies*, 2004, p. 200−212.

[14] Y. Liu and H. Zhu, "An Experimental Evaluation of the Reliability of Adaptive Random Testing Methods," in *Proceedings of the 2nd International Conference on Secure System Integration and Reliability Improvement* 2008, p. 24−31.

[15] T. Y. Chen, F.-C. Kuo, H. Liu, and W. E. Wong, "Does Adaptive Random Testing Deliver a Higher Confidence than Random Testing?," in *Proceedings of the 8th International Conference on Quality Software* 2008, p. 145−154.

[16] H. Jaygarl, C. K. Chang, and S. Kim, "Practical Extensions of a Randomized Testing Tool," in *Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference* 2009, p. 148−153.

[17] I. Ciupa, A. Leitner, M. Oriol, and B. Meyer, "ARTOO: Adaptive Random Testing for Object-oriented Software," in *Proceedings of the 30th International Conference on Software Engineering*, Leipzig, Germany, 2008, p. 71−80.

[18] J. W. Duran and S. C. Ntafos, "An Evaluation of Random Testing," *IEEE Transactions on Software Engineering,* vol. SE-10, p. 438−444, 1984.

[19] P. S. Loo and W. K. Tsai, "Random testing Revisited," *Information and Software Technology,* vol. 30, p. 402−417, 1988.

[20] T. Y. Chen, F.-C. Kuo, and H. Liu, "Distributing Test Cases More Evenly in Adaptive Random Testing," *Journal of Systems and Software,* vol. 81, p. 2146−2162, 2008.

[21] T. Y. Chen and R. G. Merkel, "Quasi-Random Testing," in *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, Long Beach, CA, USA, 2005, p. 309−312.

[22] P. M. S. Bueno, W. E. Wong, and M. Jino, "Improving Random Test Sets using the Diversity Oriented Test Data Generation," in *Proceedings of the 2nd International Workshop on Random testing* Atlanta, Georgia: ACM, 2007, p. 10−17.

[23] P. Godefroid, N. Klarlund, and K. Sen, "DART: Directed Automated Random Testing," in *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, Chicago, IL, USA, 2005, p. 213−223.

[24] K. P. Chan, T. Y. Chen, and D. Towey, "Restricted Random Testing: Adaptive Random Testing by Exclusion," *International Journal of Software Engineering and Knowledge Engineering,* vol. 16, p. 553−584, 2006.

[25] S. Xu, "Orderly Random Testing for Both Hardware and Software," in *Proceedings of the 14th IEEE Pacific Rim International Symposium on Dependable*, 2008, p. 160−167.

[26] T. Y. Chen, F.-C. Kuo, and Z. Q. Zhou, "On Favourable Conditions for Adaptive Random Testing," *International Journal of Software Engineering and Knowledge Engineering,* vol. 17, p. 805−825, 2007.

[27] J. Geng and J. Zhang, "A New Method to Solve the "Boundary Effect" of Adaptive Random Testing," in *Proceedings of International Conference on Educational and Information Technology*, 2010, p. 298−302.

[28] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and S. P. Ng, "Mirror Adaptive Random Testing," *Information and Software Technology,* vol. 46, p. 1001−1010, 2004.

[29] K. P. Chan, T. Y. Chen, F.-C. Kuo, and D. Towey, "A Revisit of Adaptive Random Testing by Restriction," in *Proceedings of the 28th Annual International Computer Software and Applications Conference*, 2004, p. 78−85.

[30] T. Y. Chen and D. H. Huang, "Adaptive Random Testing by Localization," in *Proceedings of the 11th Asia-Pacific Software Engineering Conference* 2004, p. 292−298.

[31] T. Y. Chen, R. G. Merkel, G. Eddy, and P. K. Wong, "Adaptive Random Testing Through Dynamic Partitioning," in *Proceedings of the 4th International Conference on Quality Software*, 2004, p. 79−86.

[32] F. T. Chan, K. P. Chan, T. Y. Chen, and S. M. Yiu, "Adaptive Random Testing with CG Constraint," in *Proceedings of the 28th Annual International Computer Software and Applications Conference*, 2004, p. 96−99.

[33] J. Mayer, "Lattice-based Adaptive Random Testing," in *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering* 2005, p. 333−336.

[34] T. Y. Chen, D. H. Huang, F.-C. Kuo, R. G. Merkel, and J. Mayer, "Enhanced Lattice-based Adaptive Random Testing," in *Proceedings of the 2009 ACM Symposium on Applied Computing*, Honolulu, Hawaii, 2009, p. 422−429.

[35] J. Mayer, "Restricted Adaptive Random Testing by Random Partitioning," in *Proceedings of the International Conference on Software Engineering Research and Practice* 2006.

[36] J. Mayer, "Adaptive Random Testing by Bisection with Restriction," in *Proceedings of the 7th International Conference on Formal Engineering Methods*, 2005, p. 251−263.

[37] J. Mayer, "Adaptive Random Testing by Bisection and Localization," in *Proceedings of the 5th International Workshop on Formal Approaches to Testing of Software* 2006, p. 72−86.

[38] J. Mayer, T. Y. Chen, and D. H. Huang, "Adaptive Random Testing Through Iterative Partitioning Revisited," in *Proceedings of the 3rd International Workshop on Software Quality Assurance*, Portland, Oregon, 2006, p. 22−29.

[39] T. Y. Chen, D. H. Huang, and Z. Q. Zhou, "Adaptive Random Testing Through Iterative Partitioning," in *Proceedings of the 11th International Conference on Reliable Software Technologies*, 2006, p. 155−166.

[40] F.-C. Kuo, T. Y. Chen, H. Liu, and W. K. Chan, "Enhancing Adaptive Random Testing for Programs with High Dimensional Input Domains or Failure-unrelated Parameters," *Software Quality Journal,* vol. 16, p. 303−327, 2008.

[41] J. Mayer, "Adaptive Random Testing with Randomly Translated Failure Region," in *Proceedings of the 1st International Workshop on Random Testing*, 2006, p. 70−77.

[42] T. Y. Chen and R. G. Merkel, "Efficient and Effective Random Testing Using the Voronoi Diagram," in *Proceedings of the 17th Australian Software Engineering Conference* 2006, p. 300−308.

[43] T. Y. Chen, D. H. Huang, and F.-C. Kuo, "Adaptive Random Testing by Balancing," in *Proceedings of the 2nd International Workshop on Random Testing*, 2007, p. 2−9.

[44] T. Y. Chen, F.-C. Kuo, and H. Liu, "Distribution Metric Driven Adaptive Random Testing," in *Proceedings of the 7th International Conference on Quality Software*, 2007, p. 274−279.

[45] T. Y. Chen, F.-C. Kuo, and H. Liu, "Adaptive Random Testing Based on Distribution Metrics," *The Journal of Systems and Software,* vol. 82, p. 1419−1433, 2009.

[46] T. Y. Chen, F.-C. Kuo, and H. Liu, "Enhancing Adaptive Random Testing through Partitioning by Edge and Centre," in *Proceedings of the 18th Australian Software Engineering Conference*, 2007, p. 265−273.

[47] T. Y. Chen, F.-C. Kuo, and H. Liu, "Application of a Failure Driven Test Profile in Random Testing," *IEEE Transactions on Reliability,* vol. 58, p. 179−192, 2009.

[48] A. F. Tappenden and J. Miller, "A Novel Evolutionary Approach for Adaptive Random Testing," *IEEE Transactions on Reliability,* vol. 58, p. 619−633, 2009.

[49] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T. H. Tse, "Adaptive Random Testing: The ART of Test Case Diversity," *Journal of Systems and Software,* vol. 83 p. 60−66, 2010.

[50] T. Y. Chen and F.-C. Kuo, "Is Adaptive Random Testing Really Better than Random Testing," in *Proceedings of the 1st International Workshop on Random Testing*, Portland, Maine, 2006, p. 64−69.

[51] C. Csallner and Y. Smaragdakis, "JCrasher: An Automatic Robustness Tester for Java," *Software: Practice and Experience,* vol. 34, p. 1025−1050, 2004.

[52] C. Oriat, "Jartege: A Tool for Random Generation of Unit Tests for Java Classes," in *Proceedings of the 1st International Conference on the Quality of Software Architectures*, 2005, p. 242−256.

[53] J. H. Andrews, S. Haldar, Y. Lei, and F. C. H. Li, "Tool Support for Randomized Unit Testing," in *Proceedings of the 1st International Workshop on Random Testing*, Portland, Maine, 2006, p. 36−45.

[54] B. Meyer, I. Ciupa, A. Leitner, and L. L. Liu, "Automatic Testing of Object-Oriented Software," in *Proceedings of the 33rd International Conference on Current Trends in Theory and Practice of Computer Science*, 2007, p. 114−129.

[55] R. M. Parizi, A. A. A. Ghani, R. Abdulla, and R. B. Atan, "Towards a Framework for Automated Random Testing of Aspect-oriented Programs," in *Proceedings of the ISCA 18th International Conference on Software Engineering and Data Engineering*, Las Vegas, Nevada, USA, 2009, p. 217−223.

[56] R. M. Parizi, A. A. A. Ghani, R. Abdulla, and R. B. Atan, "On the Applicability of Random Testing for Aspect-Oriented Programs," *International Journal of Software Engineering and its Applications* vol. 3, p. 1−20, 2009.

[57] G. Kiczales, E. A. Hilsdale, J. J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold, "An Overview of AspectJ," in *Proceedings of the 15th European Conference on Object-Oriented Programming* 2001, p. 327−353.

[58] R. Laddad, *AspectJ in Action: Practical Aspect-Oriented Programming*, first ed. Greenwich: Manning Publications Co. , 2003.

[59] T. J. Ostrand and M. J. Balcer, "The Category-partition Method for Specifying and Generating Functional Tests," *Communications of the ACM,* vol. 31, p. 676−686, 1988.

[60] F.-C. Kuo, "On Adaptive Random Testing," Melbourne, Australia: Swinburne University of Technology, PhD Thesis, 2006.

[61] R. E. Filman and D. P. Friedman, "Aspect-oriented programming is quantification and obliviousness," in *Proceedings of the Workshop on Advanced Separation of Concerns* 2000.

[62] I. Ciupa, A. Leitner, M. Oriol, and B. Meyer, "Object Distance and Its Application to Adaptive Random Testing of Object-oriented Programs," in *Proceedings of the 1st International workshop on Random Testing* Portland, Maine: ACM, 2006, p. 55−63.

[63] I. Ciupa, A. Pretschner, A. Leitner, M. Oriol, and B. Meyer, "On the Predictability of Random Tests for Object-Oriented Software," in *Proceedings of the 1st International Conference on Software Testing, Verification, and Validation*, 2008, p. 72−81.

[64] Y. Lin, X. Tang, Y. Chen, and J. Zhao, "A Divergence-Oriented Approach to Adaptive Random Testing of Java Programs," in *Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering*, 2009, p. 16−20.

[65] R. Pawlak, J.-P. Retaillé, and L. Seinturier, *Foundations of AOP for J2EE Development*: Apress, 2005.

[66] Z. Q. Zhou, "Using Coverage Information to Guide Test Case Selection in Adaptive Random Testing," in *Proceedings of the IEEE 34th Annual Computer Software and Applications Conference Workshops* 2010, p. 208−213.

[67] B. Jiang, Z. Zhang, W. K. Chan, and T. H. Tse, "Adaptive Random Test Case Prioritization," in *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, 2009, p. 233−244.

[68] R. M. Parizi and A. A. A. Ghani, "AJcFgraph-AspectJ Control Flow Graph Builder for Aspect-Oriented Software," *International Journal of Computer Science,* vol. 3, p. 170−181, 2008.

[69] M. L. Bemardi and G. A. Di Lucca, "An Interprocedural Aspect Control Flow Graph to Support the Maintenance of Aspect Oriented Systems," in *Proceedings of the International Conference on Software Maintenance* 2007, p. 435−444.