# Adaptive Random Testing by Localization

T. Y. Chen and D. H. Huang*
*School of Information Technology,*
*Swinburne University of Technology, Hawthorn 3122, Australia*
*Email: {tchen, dhuang}@it.swin.edu.au*

## Abstract

*Based on the intuition that widely spread test cases should have greater chance of hitting the non-point failure-causing regions, several Adaptive Random Testing (ART) methods have recently been proposed to improve traditional Random Testing (RT). However, most of the ART methods require additional distance computations to ensure an even spread of test cases. In this paper, we introduce the concept of localization that can be integrated with some ART methods to reduce the distance computation overheads. By localization, test cases would be selected from part of the input domain instead of the whole input domain, and distance computation would be done for some instead of all previous test cases. Our empirical results show that the fault detecting capability of our method is comparable to those of other ART methods.*

Keywords: random testing, adaptive random testing, localization

## 1. Introduction

It is widely recognized that exhaustive testing (testing a program with all possible inputs) is not usually feasible [1]. Hence, many test case selection strategies have been investigated to improve the probability of revealing program failures [8, 12, 13, 14, 15]. Random Testing is simple in concept and easy to implement [7, 9]. It randomly chooses test cases from the input domain until detecting a failure or exhausting test resource. However, it may be ineffective because it does not use any information of the specifications or the program code in selecting the test cases [11].

* corresponding author

A simple but effective improvement of Random Testing has been proposed [6, 10]. This enhanced method makes use of the information of the location of previous test cases and is based on the observation [6] that errors in different faulty programs may form different failure patterns. Failure patterns refer to the patterns of inputs that reveal failure. For example, in a 2-dimension input domain, the patterns can be classified into 3 major failure patterns: block, strip, and point as illustrated in Figure 1. The square represents the border of input domain and the shaded areas of the figure represent the failure patterns. They suggest that for non-point failure patterns, if test cases are more evenly spread and far separated from each other, fewer test cases would be required to detect the first failure.

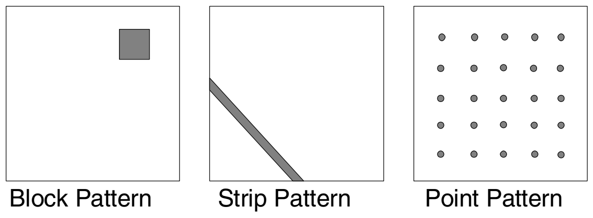


**Figure 1. Three types of Failure Patterns**

Based on this intuition, several ART methods have recently been proposed to improve Random Testing. However, distance computation is required to ensure an even spread of random test cases. We introduce the concept of localization to minimize distance computations. In this paper, we are going to illustrate how the concept of localization can be integrated with some ART methods

For the convenience of discussion, some notations are first explained. In an input domain denoted as **D**, **d**, **m** and **n** are used to denote the domain size, number of failure-causing inputs and number of test cases, respectively. The failure rate, **θ**, is defined as **m/d**. In this study, F-measure is used as the metric of fault detection effectiveness, which is defined as the number of test cases required to detect the first failure.

In this paper, Section 2 analyses the advantages and shortcomings of some ART methods. We then explain the rationale and our algorithm of ART by localization in Section 3. In Section 4, we present our simulation results. Finally, discussion is presented in Section 5.

## 2. Existing ART methods

The intrinsic characteristic of ART is to have randomly generated and widely spread test cases. Several ART methods have recently been developed.

Distance-based ART (D-ART) [10] basically has two steps applied alternatively: firstly, it randomly generates a set of test case candidates from the whole input domain, and secondly, it selects the next test case based on the criterion of maximizing the minimum distances between the test case candidates and all previous test cases.

Restriction Random Testing (RRT) [3] is another implementation of ART. It also has two steps. Firstly, it sets exclusion zones around all previous test cases, and secondly, it randomly generates test case candidates from the whole input domain but select those outside all exclusion zones as the test cases.

Effectively, both methods make use of the distance between the test case candidates and all previous test cases to determine the next test cases. The key difference between these methods is their test case selection strategies. For D-ART, the next test case is the best among all test case candidates with regard to the selection criterion of the farthest distance from all executed test cases. For RRT, a qualification is set prior to test case generation and the first qualified test case candidate is deemed as the next test case. In other words, D-ART is based on the notion of selecting the best while R-ART is based on the notion of satisfying the constraint/qualification. Though they are based on different notions, both involve distance calculation between test case candidates and *all* previously executed test cases.

Mirror ART (MART) [5] has been proposed to reduce distance computations. In MART, the whole input domain is first partitioned into disjoint subdomains, and the relevant ART method is only applied in one subdomain, and simple mirror functions are used to map the test cases into other subdomains. With **m** mirror subdomains, distance computations will be reduced approximately to $\mathbf{1/m^2}$ of the corresponding distance computations if ART method is applied alone. However, the number of mirror subdomains should be kept small to ensure the characteristic of randomness.

Two ART methods [4], which do not require distance calculation, have recently been proposed, namely ART by Random Partition and ART by Bisection. The main difference between these methods and the other methods is that these two methods specify where the test cases should be selected, rather than identifying the suitable ones amongst those randomly generated from the whole input domain. More specifically, these two methods divide the whole input domain into subdomains according to a partitioning scheme, and choose a subdomain from which the next test case is randomly generated. The chosen subdomain is referred as the test case generation region. The difference between these two methods is their partitioning schemes. ART by Random Partition divides the input domain by the most recently executed test case, and chooses the subdomain with the largest area as the next test case generation region; while ART by Bisection repeatedly divides the input domain into subdomain of equal size, and chooses the subdomain which has not contained any executed test case as the next test case generation region.

## 3. Our methodology

Both D-ART and RRT use distance as a gauge to measure whether the next test case is the farthest away or sufficiently far away from all previous test cases, respectively. Intuitively, the distance between the test cases is a good metric towards the measurement of the even spread of the test cases. Obviously, only those executed test cases, which are near the test case candidates, should be taken into consideration rather than all executed test cases in deciding the next test case. However, these two methods calculate the distance between all previous test cases and the test cast candidates. Furthermore, D-ART and RRT do not fully use the information of executed test cases. The spatial distribution of executed test cases actually provide hints about from which part of the input domain test cases should be selected.

ART by Random Partition divides the whole input domain into subdomains and chooses a subdomain as a test case generation region. It chooses the largest "blank" area, which has no executed test cases, as the test case generation region. Intuitively, a test case generated from this region will have a higher chance to be far away from previous test cases. Furthermore, the test case generation region effectively divides all previous test cases into two groups, namely the nearby and distant executed test cases. Nearby executed test cases are those located in the vertices of the test case generation regions, while the others are classified as the distant executed test cases. Nevertheless, ART by Random Partition does not make use of this useful information at all. It totally ignores the nearby executed test cases. The next test case may be very close to the nearby executed test cases, as shown in Figure 2, where the square represents the input domain, the points represent the test cases and the rectangle with darker edge represents the test case generation region. As shown in the figure, the next test case can be very close to the executed test cases.

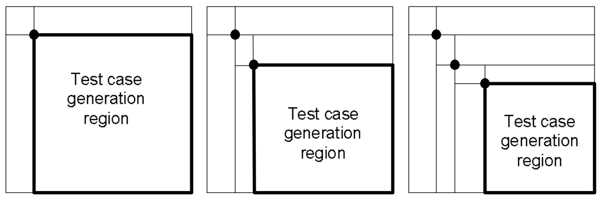

Test case generation region
Test case generation region
Test case generation region

**Figure 2. Worst case of ART by Random Partition**

In this paper, we propose an innovative approach to reduce distance computations. Our method is to integrate the notion of localization with some ART methods. By localization, there are two aspects: the first one is to restrict the selection of test cases from a part of the whole input domain, which is more likely to provide a test case distant from executed test cases; the second one is to confine distance calculation to those executed test cases which are near the test case generation region. Hence the implementation of our method has two corresponding steps. Firstly, localize the test case generation region and the previous test cases. Secondly, generate next test case from the restricted test case generation region and apply D-ART or RRT with the confined previous test case.

We use the same partitioning scheme as ART by Random Partition to divide the input domain and choose the subdomain with largest area as the test case generation region. The test case generation region not only acts as the region which is more likely to provide test case distant from previous test case, but also divides the previous test cases into two sets: nearby executed test cases and distant executed test cases. In our method, distance calculations are done only for the nearby executed test cases.

The algorithm described below is for testing a program with two real inputs, i, j, where $i_{min} \leq i \leq i_{max}$, $j_{min} \leq j \leq j_{max}$. Each vertex of a subdomain is denoted as V(x, y, Flag). "Flag" indicates whether it is an executed test case with "T" being yes and "F" being no. A subdomain is denoted by its 4 vertices in a clockwise order, say S ($V_1$, $V_2$, $V_3$, $V_4$). All subdomains are kept in an ascending sorted linked list L with respect to the sizes of their area.

### Algorithm ART by Localization

1. Initiate the subdomain linked list L with the whole input domain {($i_{min}$, $j_{min}$, F), ($i_{min}$, $j_{max}$, F), ($i_{max}$, $j_{max}$, F), ($i_{max}$, $j_{min}$, F)}, as its only element.

2. Remove the last element from L as the test case generation region. Set the nearby executed test case set E to be empty.

3. Check the "Flag" of each vertex of the test case generation region. If "Flag" is "T", add it to the nearby executed test cases set E. Denote the number of elements in E as *l*.

4. For the case of applying D-ART to the test case generation region, randomly generate a test case candidate set, C={$C_1$, $C_2$, …,$C_k$}. Calculate the Cartesian distance between test case candidates and nearby executed test cases and denote it by $dist(C_j, E_i)$. Choose a test case candidate $C_q$ as the next test case according to the following criterion.
$$\forall j \in \{1,2,...,k\}(\min_{i=1}^{l} dist(C_q, E_i) \geq \min_{i=1}^{l} dist(C_j, E_i))$$
For the case of applying RRT to the test case generation region, set the exclusion region around each element in E. Repeatedly generate random test case candidate in the test case generation region, until it is outside all exclusion regions, and denote it as the next test case, $C_q$.

5. If $C_q$ is a failure-causing input, report fault detection and terminate. Otherwise, divide the test case generation region into four test regions at $C_q$. Sort them in ascending order according to the size and insert them into the proper position of the ascending sorted linked list L. Goto Step 2.

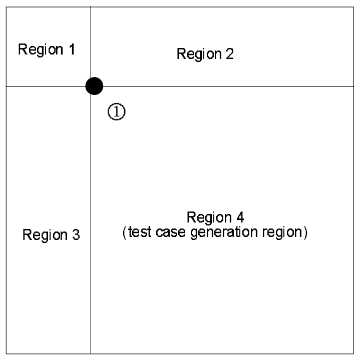
The operations of ART by localization are illustrated in Figure 3. At first, since the whole input domain is the only element in the subdomain linked list, it is obviously the test case generation region. Since there are no nearby executed test cases, test case candidate generation and distance calculation are not necessary. Within the test case generation region, a test case is randomly generated. Suppose this is not a failure-causing input (denoted as ①). Then, the test case generation region is further divided by the test case into four subdomains (Region 1, 2, 3 and 4). As shown in

Figure 3(a), Region 4 is the largest region, therefore it becomes the next test case generation region. Since ① is located in a vertex of the test case generation region, it is regarded as a nearby executed test case.

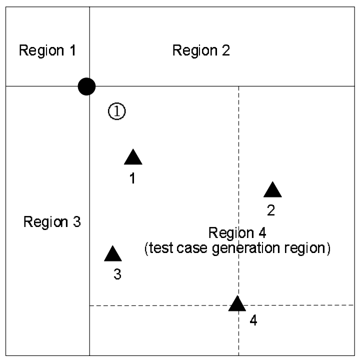
If we apply D-ART in the test case generation region (Figure 3(b)), randomly generate test case candidates within Region 4. Suppose 4 test case candidates are generated and denoted as **▲1, ▲2, ▲3** and **▲4**. In this case, test case candidate **▲4** is the farthest from ①, and therefore is chosen as the next test case. Suppose that no failure is detected. Region 4 is further divided into four subdomains. The algorithm is continued until a failure is detected.

If we apply RRT in the test case generation region (Figure 3(c)), an exclusion zone is set around the nearby executed test case ①. Suppose the first test case candidate (denote as **▲1**) is located within the exclusion zone, then it is discarded. The second test case candidate (denote as **▲2)** is outside the exclusion region and hence designated as the next test case.

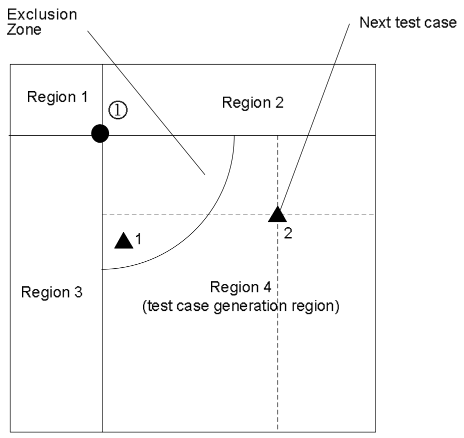
We would like to elaborate on the size of exclusion zone. Intuitively speaking, the size of exclusion zone should be determined by the size of the test case generation region. If the test case generation region is larger, then the radius of exclusion zone should be larger. For convenience, we use *Exclusion Radius Ratio* to describe the ratio of the radius of exclusion zone to the diagonal of test case generation region. It is easy to show that the maximum number of nearby executed test cases is 2. If there are 2 nearby executed test case, they must be diagonally linked. Therefore, the upper bound of Exclusion Radius Ratio must be 50%, otherwise the whole test case generation region may be covered completely by the exclusion zone, and no test case candidates can be generated. When the Exclusion Radius Ratio approaches 50%, the area other than exclusion zone becomes to disappear, and the number of attempts to generate next test cases grows dramatically. Hence in our simulation, we vary Exclusion Radius Ratio between 10% and 40%.



**(a)** 1St step for either applying D-ART or applying RRT



**(b)** 2nd step for applying D-ART

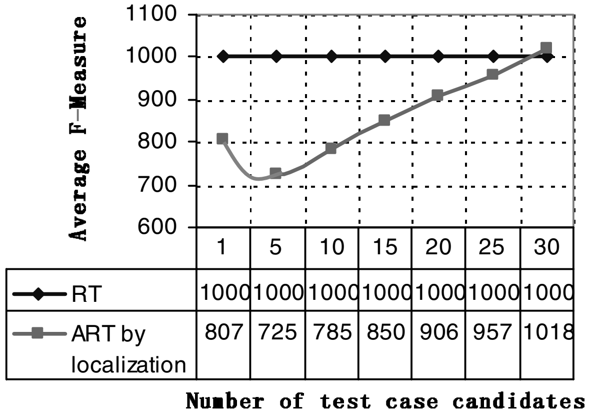


**(c)** 2nd step for applying RRT

**Figure 3. Operations of ART by Localization**

## 4. Simulations

We have conducted a series of simulation using a 2-dimension square input domain to measure the fault-detection effectiveness of our method. In each test run, a failure-causing region of the specified size and pattern was randomly assigned within the input domain. For block failure pattern, a square was used as the failure-causing region. For strip failure pattern, we randomly chose two points on the adjacent borders of the input domain. These two points were connected to form a strip with the specified size. For point failure pattern, 10 circular regions were randomly located in the input domain without overlapping each other. For each simulation, 5000 test runs were executed with failure rate of 0.001.



| | 1 | 5 | 10 | 15 | 20 | 25 | 30 |
|---|---|---|---|---|---|---|---|
| RT | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 |
| ART by localization | 807 | 725 | 785 | 850 | 906 | 957 | 1018 |

**Figure 4. Comparison of the F-Measures of RT and ART by localization (with D-ART) with different number of test case candidates (block failure pattern, θ=0.001, on average of 5000 trials)**

The first part of our simulation investigated the performance of ART by localization with D-ART under different test case candidate set sizes ranging from 1 to 30. If the number of test case candidates is equal to 1, then this algorithm is effectively the ART by Random Partition. As shown in Figure 4, the lowest F-measure occurs when the size of test case candidates is equal to 3. The lowest F-measure is less than that of RT by about 30%; while ART by Random Partition is less than RT by about 20%. When the number of test case candidates exceeds 3, F-measure will begin to increase and is greater than that of RT at the size of 30. However, with D-ART in the whole input domain, F-measure will decrease with the increase of the number test case candidates and become steady when the size is about 10.

This discrepancy can be explained as follows. When we apply ART by localization with D-ART and when the candidate set size is large, it is most likely that the next test case will be closer to a corner of the test case generation region. As a consequence, test cases will be clustered in narrow strips rather than evenly spread, as shown in Figure 5.

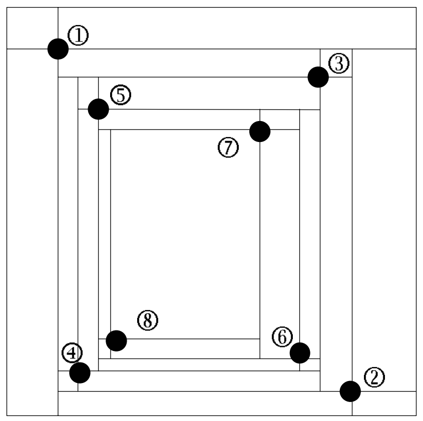
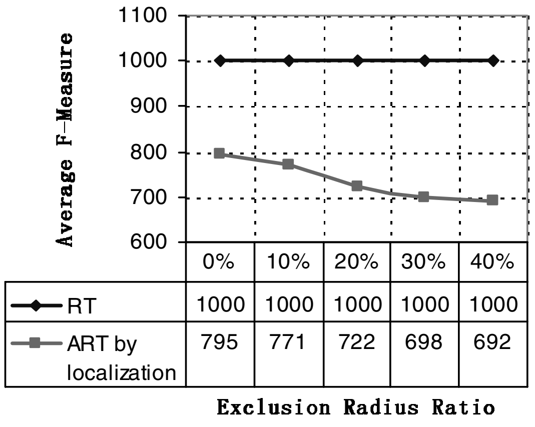


**Figure 5. Test case generation patterns for ART by localization with D-ART, with large number of test case candidates (the number represents the test case sequence)**



| | 0% | 10% | 20% | 30% | 40% |
|---|---|---|---|---|---|
| RT | 1000 | 1000 | 1000 | 1000 | 1000 |
| ART by localization | 795 | 771 | 722 | 698 | 692 |

**Figure 6. Comparison of the F-Measures of RT and ART by localization (with RRT) with different Exclusion Radius Ratio (block failure pattern, θ=0.001, on average of 5000 trials)**

The performance of ART by localization with RRT was investigated in the second part of the simulation. Here, we varied the Exclusion Radius Ratio from 0% to 40%. Again, when the Exclusion Radius Ratio is 0%, our method is effectively the ART by Random Partition. As can be seen from Figure 6, F-measure decreases while Exclusion Radius Ratio increases. At 40% of Exclusion Radius Ratio, the methods is less than that of RT by about 30%, while F-measure of ART by Random Partition is less than that of RT by about 20%.

In the third part of the simulation, we conducted our simulation using different failure patterns. Table 1 shows the results against other ART methods. (The data about D-ART, RRT and ART by Random Partition are from [2], [2] and [4] respectively). For block and strip failure patterns, our method performs better than ART by Random Partition but worse than D-ART and RRT in terms of F-measure. For point patterns the differences of all methods are insignificant.

In the last part of our simulations, we investigated whether the location of failure-causing input would have any impact on the performance of our methods. Following [2], the categories of locations of failure-causing input were defined as follows: The centre area (CENTRE) was defined as the central 80% of the whole input domain and the other area were defined as edge area (EDGE). In our simulations, the failure region was randomly assigned in anywhere of the input domain (ANYWHERE) or confined to specified area (EDGE or CENTRE). A block failure region with failure rate of 0.01 was used. The result shows that ART by localization has no preference to the types of locations (see Table 2). As a reminder, Chan et all. [2] has demonstrated that RRT favours EDGE type of failure patterns.

**Table 1. F-Measures for ART by localization and other ART methods (θ=0.001, on average of 5000 trials)**

| ART Methods | | Failure Pattern Type | | |
|---|---|---|---|---|
| | | Block | Strip | Point |
| D-ART | | 633 | 782 | 927 |
| RRT | | 621 | 710 | 953 |
| ART by Random Partition | | 797 | 965 | 980 |
| ART by localiz ation | Applying D-ART | 695 | 884 | 955 |
| | Applying RRT | 692 | 897 | 966 |

**Table 2. F-Measure of differently located failure-causing input for ART by localization. (Block failure pattern, θ=0.01)**

| Location of failure-causing input | ART by localization | |
|---|---|---|
| | Applying D-ART | Applying RRT |
| EDGE | 71 | 67 |
| CENTRE | 70 | 67 |
| ANYWHERE | 69 | 68 |

## 5. Discussion

This paper proposes an innovative approach to integrate ART and localization. Using the integrated approach, we only need to do distance computation involving the nearby executed test cases instead of involving all executed test cases. Another characteristic of our approach is that test cases are generated from certain part of the input domain rather than the whole input domain. Compared with D-ART and RRT, the overheads are considerably reduced while performance is only moderately affected in terms of F-measure. Compared with ART by Random Partition, the fault detection effectiveness has obviously been improved.

Compared with D-ART and RRT, the amount of distance computations in our method is reduced. In our method, distance computations are associated only with the nearby executed test cases, but in D-ART and RRT, distance computations are associated with all executed test cases. It should be noted that the maximum number of executed test case is two. Hence, the amount of distance computations for our method is just a linear function of the number of executed test cases. While for D-ART and RRT, the amount of distance computation is of the order of the square of the number of executed test cases.

Compared with ART by Random Partition, the overheads of partitioning the input domain and choosing the test case generation region in our method are the same. The extra work in our method is the application of D-ART or RRT in test case generation region.

The key idea of our approach is localization that makes use the information of the spatial patterns of previous test cases. The integrated method presented in this paper is just an implementation of the concept of localization. This method repeatedly partitions the input domain by most recently executed test case and selects the subdomain with largest area as test case generation region; and treats test cases located in the vertices of that region as nearby executed test cases. The shortcoming of this implementation is that it only identifies those previous test cases located on the vertices of test case region as nearby executed test case. However, some previous test cases may also be close to the edges of the test case generation region. This explains why the performance of our method is not as good as the situations of applying D-ART or RRT to the whole input domain.

We have started to investigate the performance of our proposed method to real-life programs. Integration of the concept of localization with other ART methods is also under investigation.

## References

[1] B. Beizer. *Software Testing Techniques.* Van Nostrand Reinhold, New York, 1990

[2] K.P. Chan, T. Y. Chen, F. –C. Kuo and D. Towey, "A revisit of adaptive random testing by restriction", *accepted to appear in the Proceedings of COMPSAC 2004.*

[3] K. P. Chan, T. Y. Chen and D. Towey, "Normalized restricted random testing", *8th Ada-Europe International Conference on Reliable Software Technologies*, Toulouse, France, June 16-20, 2003, Proceedings. LNCS 2655, pp. 368-381, Springer-Verlag Berlin Heidelberg 2003.

[4] T. Y. Chen, G. Eddy, R. Merkel and P. K. Wong, "Adaptive random testing through dynamic partitioning", *accepted to appear in Proceedings of the 4th International Conference on Quality Software (QSIC 2004)*

[5] T. Y. Chen, F. –C. Kuo, R. Merkel and S. Ng, "Mirror adaptive random testing", *Proceedings of the 3rd International Conference on Quality Software (QSIC 2003)*, pp. 4-11, IEEE Computer Society, 2003.

[6] T. Y. Chen, T. H. Tse and Y. T. Yu, "Proportional sampling strategy: a compendium and some insights", *The Journal of Systems and Software*, 58, pp. 65-81, 2001.

[7] R. Hamlet, "Random testing", *Encyclopedia of Software Engineering*, edited by J. Marciniak, Wiley, pp. 970-978, 1984.

[8] J. W. Laski and B. Korel, "A data flow oriented program testing strategy", *IEEE Transactions on Software Engineering*, Vol. 9, No. 3, pp. 347-354, 1983.

[9] P. S. Loo and W. K. Tsai, "Random testing revisited", *Information and Software Technology*, Vol. 30, No. 7, pp. 402-417, 1988.

[10] I. K. Mak, "On the effectiveness of random testing", *Master's thesis,* The University of Melbourne, 1997.

[11] G. Myers, *The Art of Software Testing*, New York: John Wiley & Sons, 1979

[12] J. Offutt and S. Liu, "Generating test data from SOFL specifications", *The Journal of Systems and Software*, Vol. 49, No. 1, pp. 49-62, 1999.

[13] J. Offutt, S. Liu, A. Abdurazik and P. Ammann, "Generating test data from state-based specifications", *Software Testing, Verification & Reliability*, Vol. 13, No. 1, pp. 25-53, March 2003.

[14] P. Stocks and D. Carrington, "A framework for specification-based testing", *IEEE Transactions on Software Engineering*, Vol. 22, No. 11, pp.777-793, 1996

[15] L. J. White and E. I. Cohen, "A domain strategy for computer program testing", *IEEE Transactions on Software Engineering*, Vol. 6, No. 3, pp. 247-257, 1980.