Enhanced Classification Tree Method for Modeling Pairwise Testing

Easter Viviana Sandin and Radziah Mohamad
Faculty of Computing, Universiti Teknologi Malaysia, 81310 Johor Bahru, Johor, Malaysia.
eastersandin@gmail.com

***Abstract*—Software testing is one of the most important activities to produce a high-quality system, which can increase the trust level of users. There are many types of software testing. One of those testing is called exhaustive testing. Exhaustive testing is used to produce a test suite that will be used in other testing types such as unit testing, system testing, integration testing and also acceptance testing. However, exhaustive testing is infeasible and will be time consuming. Therefore, the combinatorial testing is proposed to solve the exhaustive testing problem. There are many techniques of combinatorial testing. The popular one is called pairwise testing. It also is known as Allpairs or 2-way testing. It involves the interaction of 2 parameters. In order to perform the pairwise testing, there are procedures that need to be fulfilled. The first procedure is modeling of System Under Test (SUT). There are many models that can be used to design the test suite for pairwise testing. In this paper, the comparison for modeling of SUT in pairwise testing is performed, and the enhancement of Classification Tree Method is proposed. An example based on steps of proposed model method is also provided.**

***Index Terms*—Classification Tree Method; Modeling of SUT; Pairwise Testing.**

I. INTRODUCTION

Nowadays, intelligence technologies exist and grow as the demand grows. They put their trust on those technologies. For example, the web system such as food delivery website let people order their meal through that website without going to that restaurant. This alternative will save their time when doing important work. For an embedded system such as the airplane system requires 100% trust from customers as they use to carry many lives in them. However, the question is, how many people can put their trust on those technologies? Therefore, software testing is one of the important activities that should be performed in order to gain the software trustworthy.

Software testing is one of the important testing phases in Software Development Life Cycle. This phase is used to ensure the developed software will serve the high quality to users. It consists of black-box and white-box testing [1, 2]. Black box is focused on external behavior or functionality and white box is focused on internal implementation of software. In order to conduct the software testing, the test cases should be prepared first. The traditional way to generate the test cases is called exhaustive testing. Exhaustive testing is infeasible and time consuming especially in large or complex software system. Assume that the parameters are A, B and C. The values are as stated; A= (a1, a2), B= (b1, b2, b3), C= (c1, c2). The number of test cases generated through this method will be 2x3x2= 12 tests. The popular issue of exhaustive testing is high cost and time consuming [3]. Imagine if there are a large number of parameters and values, it may generate about thousand test cases. Hence, one of the popular test cases generation method was proposed to solve this issue. It is called as Combinatorial Testing (CT).

CT is a black-box testing type [4, 5]. It can provide better method for test cases generation. It can reduce the cost of testing and save the testing time in order to increase its effectiveness [4, 6, 7, 8, 9]. CT consists of one technique that is called t-way testing. This technique is a popular research area among researchers [7]. It needs all combinations of values of t-parameter that is tested at least once. There are 6 of t-way testing, which are 1-way, 2-way, 3-way, 4-way, 5-way and 6-way [10]. Among of these t-way, 2-way is the wildly technique in CT problems [5, 11]. 2-way testing is called as Pairwise Testing. It is used to decrease the number of test cases or test suite generated, in which it considers all interaction of at most two factors [12]. This means that they detect the constraint or problem that exists between the interactions of two parameters. The aim of this pairwise testing is to cover every pair of options in testing. Every pair of options must occur at least once and may occur more than once [10]. The other advantages of pairwise testing are, it is easy to manage and execute by testers [11].

In order to perform the pairwise testing, the first process is modeling of System Under Test (SUT). It is referring to the system that will be used for any operation such as software testing. Modeling of SUT is an important activity in pairwise testing since it is the fundamental of that testing [2, 8, 13, 14]. Each model of SUT should include the parameters, values, interaction of parameter-value and constraints [2, 4, 8, 13, 15, 16]. Parameters may represent the configuration parameters or user input parameters. The value indicates the values that consist by each parameter. Interaction shows the relationship between parameter and value. The constraint is conflict or impossible or invalid combination of parameter-value [8, 17]. All of the constraints should be detected and exclude from the list of generated test suite because they will cause the failure of the software.

Although modeling of SUT is very important to pairwise testing, there are fewer studies that had been conducted related to this research area, especially in black-box approach [6, 14]. In addition, there is no exactly the best modeling method for pairwise testing. Hopefully, many future studies will focus on this research area as the studies hold a high responsibility to help the tester understand the modeling concept.

This paper is written as follows: Section 2 explains the related work. Section 3 presents the research methodology. Section 4 shows the proposed work and Section 5 shows the case study by using proposing work. Section 6 concludes the study and mention the future work.

II. Related Works

There are many modeling that has been proposed by researchers. Different tester will prepare the different modeling methods. This is because the modeling is depending on their understanding, experience and creativity [2, 8, 13, 15]. The modeling methods can be classified into four categories, which are based on their inputs such as the requirement or functional specification, UML design artifacts, test scenario and source code [13]. The modeling methods for specification based are Category-Partition Method (CPM), Classification-Tree Method (CTM), Input Parameter Modeling (IPM) and Input Space Modeling [13]. UML design artifact based consists of Activity Diagram and Sequence Diagram. This study neglects the source code based modeling methods because pairwise testing is related to black-box testing only.

There are several modeling methods that had been proposed in the selected studies. CPM is a method that describes the parameters, values, interactions and constraints in formal test specification. The parameters are assigned as categories, while choices mean values. In [18], the author stated it can be accomplished by using six steps; Analyze specification, Partition the categories into choices, Determine constraints among choices, Write and process test specification, Evaluate generator output and lastly Transform into test scripts. However, in this version of CPM, it cannot cover the larger or complex software systems. Then [19] is the latest study that proposed the enhancement of CPM. In this study, CPM is performing the execution for behavior of functional unit, which mean that the specification of large software systems can be decomposed into functional units. The second contribution of this study is by preparing the checklist to detect mistakes. As we go through the study selection, the studies about CPM is hard to find. Since there is less number of documentation related to CPM, so this method exposes the minimal knowledge about their working model. Hence, this method may cause the lack of understanding about this model. Besides, the current CPM is still in manual modeling process.

As we go through this study, we found that CTM is more attractive than other modeling methods as their related studies is more feasible. This modeling method is improving ideas from CPM as they proposed the hierarchical form or tree form representation. This modeling process outperforms other modeling methods in term of understandable, documentation and easy to handle [14]. Not like CPM, CTM is suited for automation as they offer the graphical notation. The basic steps for CTM are the design of classification tree and the definition of test cases [14, 17]. The Definition of constraint step is added to CTM modeling [20]. It is used to define the invalid combination of input. In 2013, the new method for CTM was proposed [21]. Transformation of tree-structured to non-structured has reduced the complexity of CTM as the tester can directly define the parameters as parent node without grouping them into any categories. This method is supported by the latest study [7]. Although CTM is easier to understand and has lower complexity, they did not provide any checklist to detect any mistakes such as missing or overlapping parameters and values, and so on.

On the other hand, IPM presents the information of SUT in informal specification form. This matter causes the IPM difficult to create. There are eight steps involved in this method; Determine modeling approach, Identify parameters, Identify values, Check if IPM complete, Document constraints, Establish translation table, Add preselect test cases and Check if there is more IPM [15]. The parameters, values, interactions and constraints for this method are presented in table form and expressed in natural language (human-like language). This allows the tester to understand the concept of IPM easier. However, when information of SUT is expressed in natural language, it is more challenging to convert the information to pairwise testing standard. They also provide the checklist for detect mistakes.

Additionally, ISM is a modeling method that combines two techniques, which are Input Structure Modeling and IPM. Steps that should be followed by tester in performing the modeling by using this method is as follow; Divide the system into smaller systems (for larger software systems), Model input space of each system and Generate test cases using tools [6]. In second step, input structure modeling should be performed first. It is derived into either flat or graph techniques. The activity in this step is important especially for XML type software or systems. Then, for information about their SUT, it should be produced by conducting the IPM. As can be seen, this method is quite complicated and has high complexity because it needs to go through these two different techniques.

Activity and sequence diagrams modeling method shows the modeling process that is derived through the UML input based. In Activity Diagram [22], the steps involve are Input: UML Activity Diagram, Generate XMI files, CTDM parser parses XMI files as per the pre-defined rules and Output: CTDM model. Generally, these steps show that modeling using Activity Diagram begins from the diagram and then converted into model form. Besides, the study in [22] shows the steps involve in Sequence Diagram. They are almost the same as steps for Activity Diagram. The difference is in their parser type. The UML based modeling methods seem to reverse flows with specification based. This matter may confuse the tester. The authors of selected studies also mentioned that these modeling methods have high complexity.

As previously mentioned, the modeling is a fundamental test that should be done in order to ease the testing activities for testers and developers. However, the research focuses on the modeling method is also low. Therefore, this study is going to focus on modeling for pairwise testing. Every method has their advantages and also limitations. There is no exactly the best modeling method that can be used by them. Based on the advantages and limitations of existing modeling methods, this study endeavor to enhance the classification tree method (CTM) since it has lower complexity than other, and also the documentation for it is easy to find. Hence, it is flexible to be used by any level of users; either beginner or expert.

III. Research Methodology

In order to execute the study for this paper, there are a few flows that have been performed. It begins with conducting the literature reviews by identifying the related modeling methods for pairwise testing. There are some modeling methods that have been found which are CPM, CTM, IPM, ISM, Activity Diagram and Sequence Diagram. Then, this study identifies the criteria to compare those existing modeling methods. This paper refined the identified criteria into a table, namely Table 1.

Table 1
Comparison of modeling method

No	Modeling methods	SUT description form	Cover large system	Checklist	Documentation/ References	Understandable	Complexity
1	CPM	Test specification	Yes	Yes	Hard to find	High	Medium
2	CTM	Hierarchical form	Yes	No	Easy to find	Medium	Low
3	IPM	Informal specification	-	Yes	Hard to find	High	Medium
4	ISM	Test specification	Yes	Yes	Hard to find	Low	High
5	Activity Diagram	UML diagram description	-	No	Easy to find	Low	High
6	Sequence Diagram	UML diagram description	-	No	Easy to find	Low	High

Based on the Table 1, this study chooses the suitable model to enhance to. The chosen model is CTM as it is covered many criteria, such as covering a large system, easy to find the related documentations or references, highly understandable and low complexity. However, it does not provide the checklist and also not so easy to understand. Checklist criterion is important to consider because it will be used to check any mistakes such as missing parameters and values, overlap and so on. The checklist allows us to discover incomplete or wrong SUT information before implementing them into the test case generation approach. Besides, the understandable criterion is an important standard that needs to be fulfilled by CTM. Currently, the tester has to write the SUT information directly into tree form. The expert tester may not have any problems with that situation; however, it is quite challenging for beginner or novice user. To cover the weaknesses of this CTM, the concept in CPM and IPM can be applied in the checklist and understandable criteria.

The next flow is in enhancing the model by adding the steps in modeling process. This flow is as shown in the Section IV. After proposing the enhancement process, this study then applies it to the real case study, namely Pizza Option. It is as shown in the Section V. Finally, the study compares the proposed CTM with existing CTM. The summarization of research methodology for this study is stated in the Figure 1.

IV. Proposed Model Method

The modeling for SUT is the fundamental of testing. This activity is as a pre-process for pairwise testing. By performing the modeling method for SUT, it can help to ease the testing process for testers and developers. In pairwise testing, the modeling method is used to manage the information of SUT which are the parameters, values, and constraints. The addition, update and deletion of that information are more manageable through modeling process.

As mentioned in the previous section, the paper focuses on CTM enhancement. In order to show the processes involve in that enhanced modeling method, we use Systems Process Engineering Meta-model (SPEM). Figure 2 shows the enhanced modeling method that will be used in this research.

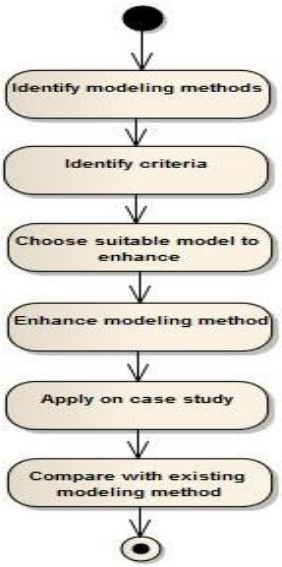

Figure 1: Research methodology for this study

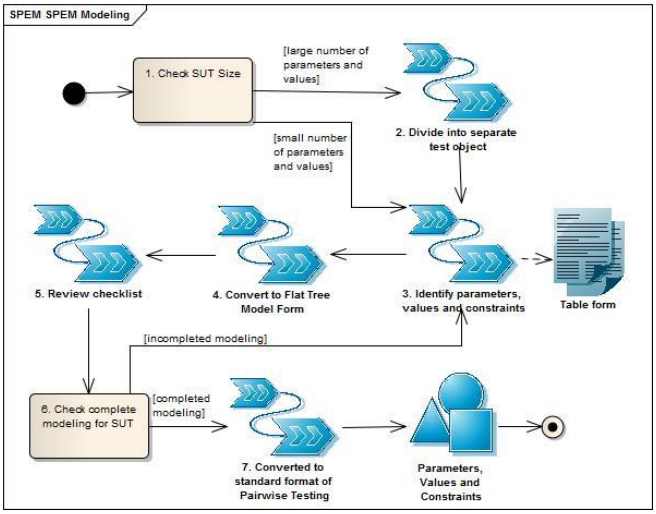

Figure 2: Steps in enhanced model method

A. *Check SUT Size*

This step is to determine whether the SUT size for selected case study is a large or small system. If it is large, then go to the second step and if it is a small system, then proceed to the third step.

B. *Divide into Separate Test Object*

If the case study is a large system, then it will be "chunked" or divided into smaller modules. It is used to reduce the difficulties in identifying the information about that case study. This step also has been in the CPM and IPM. Hence, through this step, this method can cover the large system criteria.

C. *Identify Parameters, Values and Constraints*

This step is the compulsory action in the modeling of SUT for pairwise testing. It is done on all modeling methods. In our enhanced modeling method, that information will be defined in formal specification form and more to natural language. Through this way, the novice or beginner will understand how to identify and derive the information of SUT. Therefore, this step covers the understandable criteria for modeling in pairwise testing.

D. Convert to Flat Tree Model Form

After deriving the information into formal specification form, the next step is to convert them to flat tree model. There are two main reasons why we choose to convert it to this flat tree model form. Firstly, it will be easy to trace the maintenance of SUT information. In order to add, update or delete the parameters, values, and constraints, it can be seen clearly in tree form. Besides, the flat tree model is easier to convert to standard pairwise testing compared to formal specification form. This matter has been proved when the flat tree model is proposed. Hence, the complexity of this modeling method is lower than other modeling methods.

E. Review Checklist

The checklist is used to avoid the missing parameters and values and also to avoid the invalid parameters and values. By having the checklist feature in this modeling method, all of those problems can be overcome. This step has been stated in CPM and also IPM. The checklist for our modeling method is as stated in the Table 2.

Table 2
Checklist

No	Problems	Checked
1	Missing factors:	
	Left parameters	
	Left values	
2	Overlap:	
	2 parameters consist same values	
3	Irrelevant factors:	
	Parameter with no values	
	Number of parameters<Number of values	
4	Repeated factors:	
	2 same values stated in a parameter	
	Each parameter stated more than once	
5	Irrelevant association of factors	

F. Check Complete SUT

This is the step to check if the SUT modeling method for the current case study is completed, before proceeding to the next step. However, if the SUT modeling method is not yet completed, the testers and developers can add, update or delete the SUT information by performing it at step four.

G. Convert to Standard Pairwise Testing

The last step in our enhanced modeling method converts the SUT flat tree model to standard pairwise testing. The standard pairwise testing that is mentioned here is as the PICT expressed in their standard pairwise testing. It is as stated in Figure 3.

```
Parameter 1: Value 1, Value 2, Value 3
Parameter 2: Value 4, Value 5
Parameter 3: Value 6, Value 7, Value 8
Constraints1
```

Figure 3: Standard pairwise testing

V. CASE STUDY

This section shows the example of how to conduct the modeling of SUT by using proposed method. The case study that will be used is Pizza Option. This case study refers to the menu option that is provided to the customers in order to order the pizza. There are five parameters involved; Pizza type, Crust, Toppings, Size and Delivery. Each of those parameters consists of their different values. There are some false conditions exist, as stated below:

i. Vegetarian pizza type should not take roasted chicken as their topping.
ii. Vegetarian pizza type should not take ground beef as their topping.
iii. Meat lover pizza type should not take mushroom as their topping.

All of these false conditions should be avoided from occurring in the test generation process.

Step 1: Check SUT Size: Simple? Yes (If yes, skip step 3)

Step 3: Identify parameters, values and constraints. The parameters, values and constraints for Pizza Option case study is identify as stated in the Table 3.

Table 3
Parameters, values and constraints for Pizza Option

Parameters	Values		
	1	2	3
A: Pizza type	Vegetarian cheese	Meat lover	
B: Crust	Thin crust	Extra thick	
C: Toppings	Roasted chicken	Ground beef	Mushroom
D: Size	Large	Medium	Small
E: Delivery	Eat in	Take away	
Invalid Combination: (A1, C1), (A1, C2), (A2, C3)			

Step 4: Convert to Flat Tree Model Form. The flat tree model for Pizza Option case study is defined as in the Figure 4.

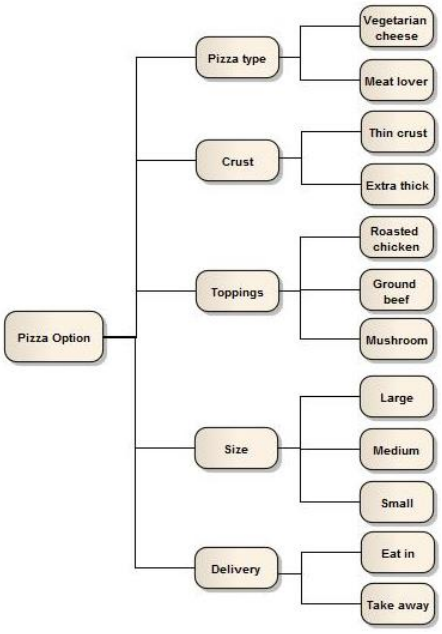

Figure 4: Flat Tree Model for Pizza Option

Step 5: Review Checklist. In order to trace the mistakes that might occur in this modeling process, the checklist need to be performed. Table 4 shows the checklist review for Pizza Option case study.

Table 4
Checklist for Pizza Option

No	Problems	Checked
1	Missing factors:	
	Left parameters	X
	Left values	X
2	Overlap:	
	2 parameters consist same values	X
3	Irrelevant factors:	
	Parameter with no values	X
	Number of parameters<Number of values	X
4	Repeated factors:	
	2 same values stated in a parameter	X
	Each parameter stated more than once	X
5	Irrelevant association of factors	X

Step 6: Check complete SUT: Yes, completed.

Step 7: Convert to standard Pairwise Testing. The standard Pairwise Testing form for Pizza Option case study is stated in the Figure 5.

```
# List of parameters and values
pizza type: vegetarian cheese, meat lover
crust: thin crust, extra thick
toppings: roasted chicken, ground beef, mushroom
size: large, medium, small
delivery: eat in, take away

# CONSTRAINTS
IF [pizza type] = "vegetarian cheese"
        THEN (NOT [topping] = "roasted chicken");
IF [pizza type] = "vegetarian cheese"
        THEN (NOT [topping] = "ground beef");
IF [pizza type] = "meat lover"
        THEN (NOT [topping] = "mushroom");
```

Figure 5: Standard pairwise testing for Pizza Option

As shown in the example above, it is found that this proposed modeling method can enhance the existing of CTM in technically. The new feature of SUT information that has been added is presented in table form, which is more to natural language. This feature can ease the beginner to understand how to use this modeling method. Besides, the checklist feature also has been inserted in this modeling method. By having this checklist, the missing of factors and existing of the invalid factors can be avoided. Table 5 shows the comparison between the existing CTM with the proposed CTM.

Table 5
Comparison of CTM modeling method

No	Modeling methods	SUT description form	Cover large system	Checklist	Documentation/ References	Understandable	Complexity
1	Proposed CTM	Test specification	Yes	Yes	Easy to find	High	Low
2	CTM	Hierarchical form	Yes	No	Easy to find	Medium	Low

VI. CONCLUSION

Modeling method for SUT is a pre-process to generate the test cases. Therefore, it is needed in the real software process in order to help the developers by easing the test case generation process, especially through documentation. This study also performs the automated modeling for SUT. There are many modeling methods that have been proposed by researchers. Each of them has their own advantages and weaknesses. In this study, the main aim is to enhance the existing of CTM based on its weaknesses compares to other existing modeling methods. CTM is chosen because it is easy to handle or manage, and also understandable and documentable. This reason makes CTM suitable for any level of users; from beginner to expert. The steps in accomplishing this proposed modeling method has been stated. An example, namely Pizza Option was chosen as a sample to be conducted in the proposed method. Then, as a result, it is found that this method can cover the weaknesses of existing CTM; lack of checklist. Besides, the objective of making this method flexible to be used by any level of user also has been achieved.

At the time of writing this paper, many other modeling methods have matured enough to assist the users. They provide many advantages that can be used to modeling SUT for pairwise testing. However, due to the main constraint of this study, which is time constraint, many other future works should be taken into considerations. The first suggested future work is through studying more papers about existing modeling methods for pairwise testing. By doing this, a lot of information that consist of strengths and weaknesses of the methods can be found. In addition, preparing the paper about automation of this proposed method also can be done. The automated modeling can help the users as currently, there only exist manual modeling.

ACKNOWLEDGMENT

We would like to thank Ministry of Higher Education Malaysia (MOE) for sponsoring the research through the FRGS grant with vote number 4F857 and Universiti Teknologi Malaysia for providing the facilities and support for the research.

REFERENCES

[1] S. K. Khalsa, and Y. Labicle, "An orchestrated survey of available algorithms and tools for combinatorial testing," in *25th IEEE International Symposium on Software Reliability Engineering, ISSRE 2014*, Naples, Italy, November 3-6, 2014, pp. 323–334.

[2] P. Satish, and K. Rangarajan, "A preliminary survey of combinatorial test design modeling methods," *International Journal of Scientific & Engineering Research*, vol. 7, no. 7, pp. 1455-1459, Jul. 2016.

[3] P. Purohit, and Y. Khan, "An automated sequence model testing (ASMT) for improved test case generation using cloud integration," *International Journal of Computer Science and Information Technologies*, vol. 6, no. 1, 488-494, 2015.

[4] M. Brcic and D. Kalpic, "Combinatorial testing in software projects," in *MIPRO, 2012 Proceedings of the 35th International Convention*, pp. 1508–1513, 2012.

[5] L. P. Mudarakola and M. Padmaja, "The survey on artificial life techniques for generating the test cases for combinatorial testing," *International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)*. vol. 2, no. 6, pp. 19-26, June 2015.

[6] M. N. Borazjany, L. S. Ghandehari, Y. Lei, R. Kacker, and R. Kuhn, "An input space modeling methodology for combinatorial testing," in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2013, pp.372-381.

[7] T. Kitamura, A. Yamada, G. Hatayama, C. Artho, E. H. Choi, N. T. B. Do, Y. Oiwa, and S. Sakuragi, "Combinatorial testing for tree-structured test models with constraints," in *2015 IEEE International Conference on Software Quality, Reliability and Security*, pp. 141-150, 2015.

[8] C. Nie, and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys*, vol. 43, no. 2, Jan. 2011.

[9] M. Patil, and P. J. Nikumbh, "Pair-wise testing using simulated annealing," *Procedia Technology*, vol. 4, pp. 778-782, 2012.
[10] D. R. Kuhn, R. N. Kacker, and Y. Lei, *Introduction to Combinatorial Testing*. London, UK: Chapman & Hall/CRC, London, 2013.
[11] J. Bach and P. J. Schroeder, "Pairwise testing-a best practice that isn't," in *Proceedings of the 22nd Pacific Northwest Software Quality Conference*, 2004, pp.180–196.
[12] L. Y. Xiang, A. R. A. Alsewari, and K. Z. Zamli, "Pairwise test suite generator tool based on harmony search algorithm (HS-PTSGT)," *International Journal on Artificial Intelligence*, vol. 2, Feb. 2015.
[13] S. Udai, "A literature survey on combinatorial testing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 4, pp. 932-936, Apr. 2014.
[14] G. Matthias, J. Wegener, and K. Grimm, "Test case design using classification trees and the classification-tree editor CTE," in *Proceedings of the 8th International Software Quality Week*, vol. 95, 1995, pp. 30.
[15] G. Mats and J. Offutt, "Input parameter modeling for combination strategies," in *Proceedings of the 25th Conference on IASTED International Multi-Conference (SE"07)*, ACTA Press, 2007, pp.255-260.
[16] P. Satish, K. Sheeba, and K. Rangarajan, "Deriving combinatorial test design model from UML activity diagram," in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2013, pp. 331-337.
[17] P. M. Kruse, *Enhanced Test Case Generation with the Classification Tree Method*. University of Berlin: Ph.D. Thesis, 2013.
[18] T. J. Ostrand, and M. J. Balcer, "The category-partition method for specifying and generating functional tests," *Communications of the ACM*, vol. 32, no. 6, pp. 676-686, 1988.
[19] T. Y. Chen, P. L. Poonb, S. F. Tang and T. H. Tse, "On the identification of categories and choices for specification-based test case generation," *Information and Software Technology*, vol. 46, no.13, pp. 887–898, 2004.
[20] P. M. Kruse and J. Wegener, "Test sequence generation from classification trees," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, 2012, pp. 539-548.
[21] T. B. Do, T. Kitamura, N. V. Tang, G. Hatayama, S. Sakuragi and H. Ohsaki, "Constructing test cases for N-wise testing from tree-based test models" in *Proceedings of the Fourth Symposium on Information and Communication Technology*, 2013, pp. 275–284.
[22] P. Satish, A. Paul and K. Rangarajan, "Extracting the combinatorial test parameters and values from UML sequence diagrams," in *IEEE International Conference on Software Testing, Verification, and Validation Workshops*, 2014, pp. 88-97.