A Preliminary Survey of Combinatorial Test Design Modeling Methods

Preeti Satish, Krishnan Rangarajan

Abstract— Combinatorial Testing is a systematic black box testing method. Combinatorial test model derivation is the first and primer step in it and till date it is practiced manually. The paper provides an insight into, to the best of our knowledge, the first survey specific to the test model derivation step of combinatorial testing. The survey focuses on comprehending and consolidating the state-of-art work carried. The survey reveals that there is huge scope for future work in terms of more empirical studies and new automated approaches as the automation of the same is still in a very naive stage.

Index Terms— Testing, model based testing, combinatorial testing, pairwise testing, test automation, test model, software quality;

1 Introduction

Software quality is an essential attribute of software and can be achieved through effectual testing. Testing is a method of exercising the software with precise inputs and observe the output for correctness followed by amending and revising the software as needed. There are two types of testing approaches namely white-box and black-box (BB) testing. The intention of white-box testing [1] is to test the internal implementations of the program like program statement and data structures. The bases for test case generation are the internal structures of the program such as the program code. The different approaches to white-box testing are control flow based testing, data flow based testing, and mutation testing. On the contrary, the intention of black box testing is to test the external behavior of the software. It tests the functionality of the system by exercising the different input and output conditions to the required coverage level and quality. Therefore, it is also known as functional testing or behavioral testing. The bases for test case derivation are the external descriptions of the software, like requirements document and design parameters. The different approaches to BB testing are Exhaustive testing, equivalence partitioning, Cause-Effect Graphing, Combinatorial testing, State-Based Testing and Error Guessing [2]. Among these BB testing techniques the Combinatorial Testing (CT) [3] is gaining high importance because of its spectacular results. CT follows a rigid testing steps namely (1) Test Model derivation, (2) Test case generation, (3) Test Execution, (4) Fault Identification and Analysis (5) Regression Testing. Among these steps, the test model derivation is a very fundamental and an imperative stage, as the next steps are mainly dependent on the model. Test Model derivation is practiced manually in the current practical scenario, and the automation of it is still in an emerging stage. Though various surveys articles are published in the literature corresponding to entire CT procedure [4] or test case generation of CT [5][6], there are no survey articles presented specific to CT modeling step. Therefore, considering this state-of-art and inherent importance of CT modeling, we bring out the potential of it in the form of a survey. The paper is organized as follows. Section 2 states the difference between functional testing and Model Based Testing (MBT) and CT, section 3 presents the test model overview, section 4 presents the survey conducted followed by the conclusion and future work.

- *Preeti Satish, Associate Professor, Dayananda Sagar College of Engineering, India, E-mail: preetisatish8@gmail.com*
- *Krishnan Rangarajan, Professor, CMR Institute of Technology, India, E-mail: krishnanr1234@gmail.com*

2 Functional Testing, MBT and CT

In any testing technique, test case generation plays an important role in the overall testing process. Tests Cases (TC's) form the heart of testing because the quality of TC's governs the quality of test conduction and the associated software quality. Most of the times TC's are constructed directly either manually or automatically usually from requirements documents as shown in fig.1a. MBT techniques [7] have shown prolific advances in the recent years. Model based testing is a testing technique, wherein first a model of the SUT is created from the requirements document and then the TC's are generated automatically from the model as depicted in fig.1b. Thus the main intent of MBT is automation. In MBT, the quality of testing is directly dependent on the quality of the model created. Therefore, the model must be concise, precise and abstract in nature. Developing a model at the right level of abstraction is the key success to MBT. MBT is seen as BB testing because TC's are generated from the model that is built from requirements and not source code. Models can be used to depict various facets of SUT. Examples of different kinds of models are FSM, state charts, activity diagrams, sequence diagram, Markov chains, and grammars. Among these, FSM, state charts, activity diagrams, sequence diagram are supported by Unified Modeling Language (UML), which has become a defacto standard for modeling and design of software systems [8]. UML encompasses a spectrum of diagrams to model the software and visually interpret them at behavioral, interactional or structural level. UML behavior diagrams are extensively used for test case generation in the recent years because they embed the dynamic aspects of the system. The volume of papers in the field of UML based MBT and its survey is enormous [9] [10]. However, CT is a systematic testing methodology that calls for a test model (test design model) to be developed first. A subset of TC's is then generated by carefully selecting from the test model as shown in fig 1c.

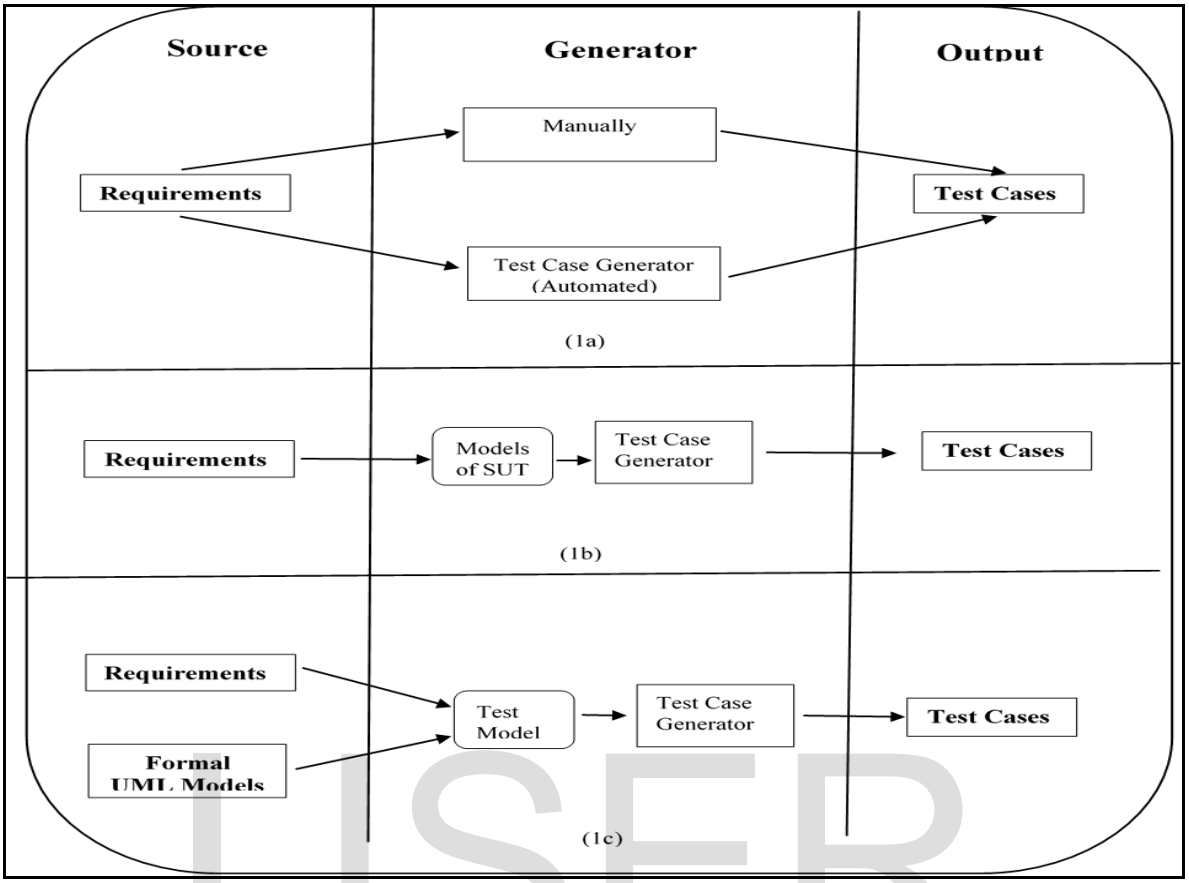

Fig. 1. (a) Manual and Automated testing (b) MBT (c) CT

3 Overview of Test Model

In combinatorial testing (CT), which deals with testing combinations of input, modeling the test space is an important pre-requisite and the efficiency of CT largely depends on this input space [4]. The test models or the Combinatorial Test Design (CTD) model enclose information about parameters, values, and the constraints. An example test model of an application is as shown in the fig. 2. The application here is expected to run on a variety of platforms like different kinds of operating systems. The test parameters are operating system and browser. Each parameter has multiple values like the parameter browser has two values Firefox and Internet Explorer and parameter operating system has two values Windows and Linux. An example of a constraint may be that value Linux of parameter operating system is chosen, then the value Internet Explorer of parameter browser cannot be selected. Examples of test factors are user inputs, configuration parameters internal and external events of the SUT [11].

Parameters	Operating Systems	Browser
Values	Windows	Mozilla Firefox
	Linux	Internet Explorer

Fig. 2. An example of CT Test Design Model

4 Survey

As per our understanding, the volume of papers dealing with the derivation of the test design model is very less, and no survey has been conducted so far. Hence, as a preliminary work, we concentrate and project the work carried out in the derivation of CT test design model for functional testing from UML models. We have collected around 20 main papers that cover different modeling methods.□

The classification tree method (CTM) introduced by Grochtmann et al. [12] has been successfully tried out on real examples in Daimler-Benz Group. The initial step is to identify the classifications (parameters) and classes (values). They define two types of classifications. P-type classification represents an input parameter, and e-type classification represents an envi-

ronment condition. Classes represent the subsets of values for each classification. Next, for the identified classification and classes a classification tree is constructed, followed by test case table and test case generation. Their experiments found that CTM method showed a good error detection rate and the method was amenable to automation. Their implication suggests that identification of parameters and values is a creative task and hence is not amenable to 100 percent automation.

Thomas J Ostrand et.al [13] proposed a systematic method called as the category partition method (CPM) to generate functional tests from specifications. CPM involves a series of decomposition steps. The tester decomposes the functional specification into separately testable functional units. The next level of decomposition involves identifying the parameters and environment conditions that affect the execution behavior of each of the functional units and partitioning them into categories by carefully reading the specification. The next step in the decomposition process is to partition the categories into distinct choices which comprise of different kinds of values, which are probable for the category. The tester then identifies constraints embedded among the choices. The identified categories, choices and constraints are written into a formal test specification that is further processed by Test Specification Language (TSL) test case generator tool.

AETG [14] present their experience in using AETGspec notation which is a part of AETG software system for representing the input space model. The authors applied the technique to four applications of Bellcore products and brought out that modeling the test space is very fundamental and needs domain knowledge so that it maps correctly

Chen et al. [15] explain that CPM and CTM methods for identifying categories and choices are adhoc in nature and the quality of test cases generated based on these techniques may be in question. The authors have conducted three empirical studies and based on their experience have articulated a checklist for detecting missing categories and problematic categories and choices. The checklist helps the testers to avoid them and also it provides an insight into developing systematic methods for identification of categories and choices. In [16] they present an algorithm to identify the categories and choices, and some of the choice relations based on the guard conditions in the activity diagrams. The guard conditions are associated with execution behavior of the software and therefore they are likely to hold the modeling information. However, the identified categories and choices should be further refined by tester manually.

Mats Grindal and Jeff Offutt [17] focuses on the overall flow of the modeling process. The author suggests two approaches to input parameter modeling namely interface-based- IPM and Functionality Based-IPM. The strength and weakness comparison of both the approaches reveals that Functionality Based-IPM includes more semantic information and useful for subsequent test case generation. The paper defines some additional properties like missing factors, irrelevant parameters, that a set of parameters must satisfy and some strategies for identifying values like boundaries, valid and invalid values, etc. Overall the paper gives more guidance for identification of model elements, and it also supports for evaluation of completeness of IPM.

Grindal M et al. [5] have performed an evaluation of five different combination strategies for selecting a subset of test cases from test model. The evaluation comparison was based on the number of test cases generated, the number and type of faults identified, the decision coverage and failure size. Their observation relating to IPM is, for a fixed number of parameter-values, it is good to have many parameters with few values for Each Choice (EC), Base Choice (BC), Orthogonal Array (OA), and AETG test selection methods and it is better to have few parameters with many values for All choice (AC) method. In their experiments, the model was kept stable, and the test selection strategies were varied. Experimenting with combinations of different IPM methods with different test selection strategies and investigating how and in what way it affects testing issues is still an open research issue.

Mehra N et al. [18] proposes an input space modeling strategy for combinatorial testing. The strategy comprises of 2 steps, input structure modeling(ISM) and input parameter modeling (IPM). First ISM tries to detect the structural relationship among the different components in the input space using two types of structures namely flat and graph. The graph structure is modeled using IML notations and is useful to represent composition relationships. After the input structure is modeled, the second step is IPM for which the basic methods of CPM and CTM are used.

Itai Segall et al. have worked on input space modeling for combinatorial testing. In one of the papers [19], they have presented a Cartesian product based method to derive the CTD model and represented the model using binary decision diagrams(BDD).

Itai Segall et al. in [20] introduce two new constructs namely counters and value properties in the CTD model. Counters are parameters that count values in other parameters. Properties are defined for parameters complexity of capturing the restrictions and simplify the modeling activity. And in [21] they list some common pitfalls regarding completeness, correctness, and redundancy that affects the CTD model. The authors have suggested solutions for some of the common patterns, like optional and conditionally excluded values, multi-selection, ranges and boundaries, order and padding, multiplicity and symmetry.

Preeti Satish et.al have described a rule based approach to derive a CTD model from UML activity diagram [22] and sequence diagram [23]. A parser has been implemented to parse the XMI representations of the UML diagrams based on the formulated rules and arrive at an initial set of parameters, values, and constraints. Further manual refinement of the model is necessary making the process semi-automatic.

Sabharwal et al. [24] propose to derive test model elements from source code. The source code is first converted to flow graph, onto which data flow analysis is performed to identify the CT interactions. Schroeder and Korel et al. [25] support modeling by recognizing the program input and program output relationship.

Sabharwal et al. [24] propose to derive test model elements from source code. The source code is first converted to flow graph, onto which data flow analysis is performed to identify the CT interactions. Schroeder and Korel et al. [25] support modeling by recognizing the program input and program output relationship.

Apart from how to derive the test model, validating the model is also an important research aspect. Paolo Validation [30] focuses on this aspect and present a paper, on how to validate the model. The validation factors considered are based on (1) consistency of constraints, (2) constraint implied by other constraint and (3) are the parameter-value identified are necessary?

Though the main focus of the authors in [27], [28], [29] has been the test suite generation, Lott et al. [27] have presented example system requirements along with guidelines for modeling the input space which serves as a tutorial for applying CT. Czerwonka [28] on how to model the input space in practical so that pure pairwise testing approach more applicable and Krishnan et al. [29] have given hints on how to identify the model elements from natural language.

We summarize the work carried in the chronological order as shown in table 1. The table heads illustrate the author and year, the input considered for deriving the CT test model, the method used and Automation level.

TABLE 1. SUMMARY OF WORK CARRIED IN CTD DERIVATION IN CHRONOLOGICAL ORDER

Authors	*Year*	*Source/Input*	*Method used for test Design*	*Automation Level*
Grochtmann & Grimm	1995	Requirements Specification	CTM	Manual
Thomas J Ostrand et al.	1988	Requirements Specification	CPM	Manual
S. R Dalal et al.	1999	Requirements Specification	AETGspec	Manual
Robert M. Herons et al.	2003	Formal Specification 'Z"	Signature and Predicate based	Semi-automatic
Chen T.Y et al.	2004	Requirements Specification	Choice relation framework based on CPM, CTM	Manual
Chen T.Y et al.	2004	UML Activity Diagram	CPM	Semi-automatic
Krishnan et al.	2007	Requirements Specification	Heuristics	Manual
Mats Grindal et al.	2007	Requirements Specification	CPM based	Manual
Itai Segall et al.	2012	Functional Specification	BDD based	Manual
Itai Segall et al.	2013	Functional Specification	Experience & Analysis based	Manual
Itai Segall et al.	2013	Functional Specification	Experience & Analysis based	Manual
Mehra Borazjany et al.	2013	Functional Specification	Input Structure Modeling(ISM) & IPM	Manual
Preeti Satish et al.	2013	UML Activity Diagram	Rule based Approach	Semi-automatic
Preeti Satish et al.	2014	UML Sequence Diagram	Rule based approach	Semi-automatic
Sabharwal et al.	2014	Source Code	Rule Based	Semi-automatic
M Spichkova et al.	2015	Test Scenarios	Rule based framework	Semi-automatic

5 CONCLUSION

We provide an insight into the research carried out in the modeling step of combinatorial testing. It is one of the most important step because the subsequent steps highly depend on the model. Modeling is an art, and the test designer needs domain understanding and experience. Therefore, it is not possible to completely automate the process. However, studies show that it can be semi-automated, thereby helping the test designer in his decisions. The survey also reveals that the various inputs considered for model derivation are requirements specifications, UML design artifacts, test scenarios, and source code. However more empirical study in each of the cases is still required along with new automated approaches, and hence CT modeling has a huge scope for future expansions.

References

[1] White-box testing - Wikipedia, the free encyclopedia. [https://en.wikipedia.org/wiki/White-box_testing]

[2] Black-box testing - Wikipedia, the free encyclopedia. [https://en.wikipedia.org/wiki/Black-box_testing]

[3] Kuhn, Rick, Yu Lei, and Raghu Kacker. "Practical combinatorial testing: Beyond pairwise." It Professional 10, no. 3 (2008): 19-23.

[4] Nie, Changhai, and Hareton Leung. "A survey of combinatorial testing." ACM Computing Surveys (CSUR) 43, no. 2 (2011): 11.

[5] Grindal, Mats, Jeff Offutt, and Sten F. Andler. "Combination testing strategies: a survey." Software Testing, Verification and Reliability 15, no. 3 (2005): 167-199.

[6] Khalsa, Sunint Kaur, and Yvan Labiche. "An orchestrated survey of available algorithms and tools for combinatorial testing." In 2014 IEEE 25th International Symposium on Software Reliability Engineering, pp. 323-334. IEEE, 2014.

[7] Utting, Mark, and Bruno Legeard. Practical model-based testing: a tools approach. Morgan Kaufmann, 2010.

[8] El‐Far, Ibrahim K., and James A. Whittaker. "Model‐Based Software Testing." Encyclopedia of Software Engineering (2001).

[9] Dias Neto, Arilo C., Rajesh Subramanyan, Marlon Vieira, and Guilherme H. Travassos. "A survey on model-based testing approaches: a systematic review." In Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies: held in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007, pp. 31-36. ACM, 2007.

[10] Utting, Mark, Alexander Pretschner, and Bruno Legeard. "A taxonomy of model‐based testing approaches." Software Testing, Verification and Reliability 22, no. 5 (2012): 297-312.

[11] Kuhn, D. Richard, Raghu N. Kacker, and Yu Lei. Introduction to combinatorial testing. CRC press, 2013.

[12] Grochtmann, Matthias, Joachim Wegener, and Klaus Grimm. "Test case design using classification trees and the classification-tree editor CTE." In Proceedings of Quality Week, vol. 95, p. 30. 1995.

[13] Ostrand, Thomas J., and Marc J. Balcer. "The category-partition method for specifying and generating fuctional tests." Communications of the ACM 31, no. 6 (1988): 676-686.

[14] Cohen, David M., Siddhartha R. Dalal, Michael L. Fredman, and Gardner C. Patton. "The AETG system: An approach to testing based on combinatorial design." IEEE Transactions on Software Engineering 23, no. 7 (1997): 437-444.

[15] Chen, Tsong Yueh, Pak-Lok Poon, Sau-Fun Tang, and T. H. Tse. "On the identification of categories and choices for specification-based test case generation." Information and software technology 46, no. 13 (2004): 887-898.

[16] Chen, Tsong Yueh, Pak-Lok Poon, Sau-Fun Tang, and T. H. Tse. "Identification of categories and choices in activity diagrams." In Fifth International Conference on Quality Software (QSIC'05), pp. 55-63. IEEE, 2005.

[17] Grindal, Mats, and Jeff Offutt. "Input parameter modeling for combination strategies." In Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering, pp. 255-260. ACTA Press, 2007.

[18] Borazjany, Mehra N., Laleh Sh Ghandehari, Yu Lei, Raghu Kacker, and Rick Kuhn. "An input space modeling methodology for combinatorial testing." In Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on, pp. 372-381. IEEE, 2013.

[19] Segall, Itai, Rachel Tzoref-Brill, and Eitan Farchi. "Using binary decision diagrams for combinatorial test design." In Proceedings of the 2011 International Symposium on Software Testing and Analysis, pp. 254-264. ACM, 2011.

[20] Segall, Itai, Rachel Tzoref-Brill, and Aviad Zlotnick. "Common patterns in combinatorial models." In 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, pp. 624-629. IEEE, 2012.

[21] Segall, Itai, Rachel Tzoref-Brill, and Aviad Zlotnick. "Simplified modeling of combinatorial test spaces." In 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, pp. 573-579. IEEE, 2012.

[22] Satish, Preeti, K. Sheeba, and Krishnan Rangarajan. "Deriving combinatorial test design model from UML activity diagram." In Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on, pp. 331-337. IEEE, 2013.

[23] Segall, Itai, Rachel Tzoref-Brill, and Aviad Zlotnick. "Common patterns in combinatorial models." In 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, pp. 624-629. IEEE, 2012.

[24] Sabharwal, Sangeeta, and Manuj Aggarwal. "Identifying Interactions for Combinatorial Testing using Data Flow Techniques." ACM SIGSOFT Software Engineering Notes 39, no. 6 (2014): 1-4.

[25] Schroeder, Patrick J., and Bogdan Korel. Black-box test reduction using input-output analysis. Vol. 25, no. 5. ACM, 2000.

[26] Spichkova, Maria, Anna Zamansky, and Eitan Farchi. "Towards a human-centred approach in modelling and testing of cyber-physical systems." In Parallel and Distributed Systems (ICPADS), 2015 IEEE 21st International Conference on, pp. 847-851. IEEE, 2015.

[27] Lott, C., Ashish Jain, and S. Dalal. "Modeling requirements for combinatorial software testing." In ACM SIGSOFT Software Engineering Notes, vol. 30, no. 4, pp. 1-7. ACM, 2005.

[28] Czerwonka, Jacek. "Pairwise testing in the real world: Practical extensions to test-case scenarios." In Proceedings of 24th Pacific Northwest Software Quality Conference, Citeseer, pp. 419-430. 2006.

[29] Krishnan, R., S. Murali Krishna, and P. Siva Nandhan. "Combinatorial testing: learnings from our experience." ACM SIGSOFT Software Engineering Notes 32, no. 3 (2007): 1-8.

[30] Arcaini, Paolo, Angelo Gargantini, and Paolo Vavassori. "Validation of models and tests for constrained combinatorial interaction testing." In Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on, pp. 98-107. IEEE, 2014.