# Combined Architecture for Early Test Case Generation and Test suit Reduction

**Mr. Saurabh Karsoliya, Prof.Amit Sinhal, Er.Amit Kanungo**

**Computer Science and Engineering, MANIT**
**Bhopal, M.P., India**

**Information Technology Department, TIT**
**Bhopal, M.P., India**

**Information Technology Department, TIT**
**Bhopal, M.P., India**

## Abstract

Model based combinatorial testing is the most effective and efficient method of systematic interaction testing. In this multiple variables can interact with test model in the form of combination and each set is considered as a test pairs. It is used mainly for reducing the test pair's size, complexity and time of test generation. Accomplishment of this reduction task is the most vibrant and tedious activity because this consist of test data extraction, boundary analysis, path measurement analysis, condition coverage etc.

The problem with Model based combinatorial testing is optimization of cost and effort which can only be achieved by early test case generation methods. For this we are using UML diagrams from design phase.

For reducing the test size and complexity we have proposed a new MBTGA framework and an MTGIPO. In this paper we focus our research on empirical study and result obtained by developed framework. We are also proposing a new UML design data extraction (DDE) algorithm for getting the useful information from a given UML diagram as input.

***Keywords***: *Combinatorial test, Pairwise test, UML, NP-complete, Genetic algorithm, MBT, UML, MBTGA, MTGIPO, DDE*

## 1. Introduction:

### Combinatorial Testing As Research Domain

Systematic testing of highly-configurable software systems, e.g. systems with many optional features, can be challenging and expensive due the exponential growth of the number of configurations to be tested with respect to the number of features. It is estimated that 30% of an enterprise's IT budget is devoted to the original development and 70% is for enhancements and fixing bugs not discovered during original development [3]. Thus the usage of Combinatorial Interaction Testing (CIT) technique can improve the effectiveness of the testing activity for these kinds of systems, at the only cost of modeling the system's configurations space.

Combinatorial testing can help detect problems like interaction failures of combinations this early in the testing life cycle. The key insight underlying *t*-way combinatorial testing is that not every parameter contributes to every failure and most failures are triggered by a single parameter value or
interactions between a relatively small numbers of parameters [4]. In fact, CIT consist in systematically testing all possible partial configurations (that is, involving up to a fixed number of parameters only) of the system under test. The most commonly applied combinatorial testing techniques pairwise testing, which consists in applying the smallest possible test suite covering all
the pairs of input values (each pair in at least one test case). It has been experimentally shown that a test suite covering just all pairs of input values
can already detect a large part (typically 50% to 75%) of the faults in a program [3, 4]. Moreover, incorrect behaviors or failures due to unintended feature interaction, detected by CIT, may not be detectable by other more traditional approaches to systematic testing [1, 5].

**2.Purpose of Study (Problem Statement):**The immediate purpose of this research is to Design & Develop a MBTGA Framework for Programmed Combinatorial Test (PCT) that can help to provide assurances of reduced overhead of testing . Some techniques for providing such assurances have been developed in the past, but no single technique has provided a complete solution to the problem. Thus, this thesis will explore the effectiveness of combining two such techniques (Unified Modeling & Combinatorial Testing) into a single tool. The more general purpose of this research is to improve the available methods of software testing.

There are several major challenges that completely resolved by our tool with testing modern software.

Some are as follows

- Automation of the test case generation and their execution.
- Development of domain and software engineering expertiseneeded for adequate testing.
- Formalization and modeling of the software specifications and implementations, and software testing process and effects. The reduction in growing complexity of the modern software-based systems.
- Generating Test Cases Criteria at the Time of Design and Requirement Analysis (Early Test Case Creation saves time & cost).
- Extracting the data from UML diagram to generate test cases.
- Generating the reduced number of test cases (Test Suite Size).
- Controlling the Test Case Creation by Constraints & Relations.
- Providing the maximum test coverage (100% for 2-way pairwise testing).
- Reducing the execution time for testing.

Thus, by considering the entire above research objective we have developed a Automated Tool which follow the proposed design architecture of MBTGA & Implemented an algorithm for this interaction testing based on Enhanced MBTGA based IPOG (MTGIPO). We also proposed a UML based data extraction algorithm (DDE). Our tool is capable of generating the minimum test cases in comparison with other publicly available tool.

### 2.1 Literature Survey

**Proposed Methodology: -** Complexity of software needs to identify better techniques for different functionalities in the software development life cycle. This complexity is truly reduced by deterministic decision environment of parameters in quality assurance of software. It is mainly done by ways of testing, which is an activity that faces constraints of both time & resources. Testing the outbound of any software is been judged by means of special type of test in black box testing known as combinatorial testing. It is a well known dynamic approach for quality improvements because it provides effective error detection at very low cost. Creation of optimal set of test will effectively decreases complexity of the software system by pairing the input parameters through pairwise testing using orthogonal arrays & Latin squares. Hence an efficient strategy is required to reduce the number of test cases formed by above mentioned method.

MBTGA with enhanced IPOG has contributed to enhance many known CA and MCA that exist in the literature. In our research & implementation we proves that the given algorithm & design architecture of MBTGA (Model Based Test Generation Architecture) is well defined for improving efficiency thorough multiple parameters (Size, Time, Complexity, Cost etc.).We have implemented our tool based on two proposed approaches. First is MBTGA Algorithm & Design Architecture for 2-way (Pairwise Testing ) & a Modified Test Generation IPO (MTGIPO) for 3,4 way interaction testing. The MTGIPO uses algorithmic approach based on IPOG (In Parameter Order General) strategy for test case reduction with improved parameters. While for pairwise test suite creation phase of MBTGA requires searching the best pair combination for pairtest (2 way Test).

**Design Architecture**

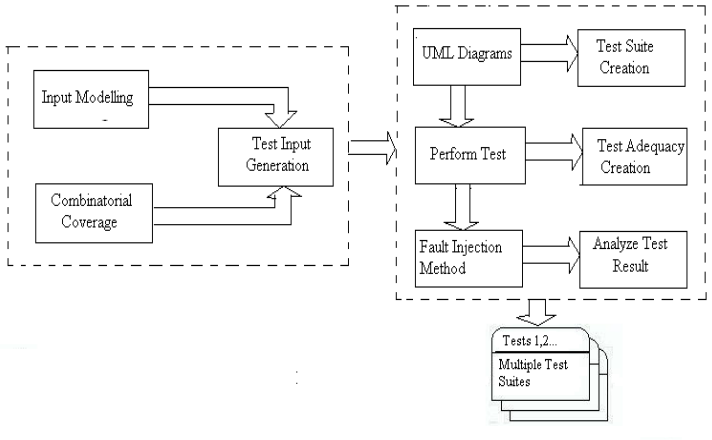



Fig:- A New Model Based Test Design Architecture (MBTDA) For Test suite creation

## 3. PROPOSED SOLUTION:

During our research & implementation we had used a result which is been proven that the problem of combinatorial testing is NP complete & whose solution may be achieved by heuristic or genetic based search algorithm. We are calculating our result of reduced pairtest through genetic algorithm. It has two basic operator's mutation & crossover. We first construct the activity diagram for the given problem and then randomly generate initial test cases, for a program under testing. Then, by running the program with the generated test cases, we obtain the corresponding program execution traces. Next, we compare these traces with the constructed activity diagram according to the specific coverage criteria. We use path coverage as test adequacy criteria. Next, we propose a novel approach to generate test cases from test scenarios using UML activity, sequence and class diagram. First we generate test scenarios from the activity diagram and then for each scenario the corresponding sequence and class diagrams are generated. After that we analyze the sequence diagram to find the interaction categories and then use the class diagrams to find the settings categories. After analyzing each category using category partitioning method, its significant values and constraints are generated and respective test cases are derived. Finally, we propose a technique to optimize the generated test cases automatically. We define an error minimization technique in our approach, which works as the basic principle for optimized test case generation. Transition coverage is used as test adequacy criteria in this approach.

### 3.1 Developing a Common Framework (MBTGA)

Several approaches to design test cases and application of software testing have been proposed by researchers. We also know that testing is a very important phase of software development and always comes at the last. No such technique is been developed to generate test cases before the implementation of code. So we are focusing our research on development of a framework on the basis of which it is been possible of developers and stakeholders to generate different

test scenarios previously. We are also contributing our work towards the optimization of test case generation strategies via combinatorial testing. Thus by enriching the above two main goals of early generation of test case by UML through Model based testing & improving the test case efficiency by enhancing the combinatorial testing methodologies, we had developed a UML based combinatorial approach(MBTGA) framework. It will fulfill all the requirements and serve best to accomplish our research objectives. Also the prototype tool being developed to prove efficiency and effectiveness of the proposed methodology.

We separate our research in two domain in which first one focuses on reducing the test case size, complexity and efforts. Second one is early generation of test cases, at the time of requirement analysis and design which saves cost, time and efforts. Our project also makes it possible to identify the bugs which tend to come at the time of development and installation. To overcome our first research domain objective we developed the solution in various phases. For this initially we proposes a new flow structure to inculpate both the concept in one. Then we develop an algorithmic approach based on Unified Model Architecture of MBTGA for pairwise testing (Interaction 2-way) & MTGIPO (Enhanced Unified In Parameter Order General for Interaction > 2 way) strategy for test case reduction with improved parameters. In this test suite creation phase of MBTGA requires searching the best pair combination for pairtest (2 way Test).We also applying test suite prioritization methodology to enhance the performance of testing & also reduces the test suite size & complexity. To accomplish our second research domain goal we use model based testing (MBT). From this we uses UML, which supports object-oriented technology, is widely used to describe the analysis and design specifications of software development. UML models are an important source of information for test case design. UML activity diagrams describe the realization of the operation in design phase and also support description of parallel activities and synchronization aspects involved in different activities perfectly. Now this part must extract data from UML diagram so to overcome this challenge we had also proposed a new UML test data extraction (DDE) algorithm.

### 3.2 Taking UML as Model in Model Based Testing

Model based testing (MBT) refers to the type of testing process that focuses on deriving a test model using different types of formal ones, then converting this test model into a concrete set of test cases [5]. Models are the intermediate artifacts between requirement specification and final code. Models preserve the essential information from the requirement, and are the basis for implementation. Instrumentation of models into testing process is the prime subject of concern of our thesis. Testing methodologies which uses model is called *model based testing (MBT)*. Development of *unified modeling language (UML)* has helped a lot to visualize/realize the software development process. At the earliest stage of *software development life cycle (SDLC)*, no one including user and developer can see the software; only at the final stage of the product development it is possible. Any errors/problems found out at the final stage, it incurs a lot of cost and time to rectify, which is very much crucial in IT industry.

UML is the modeling language, which supports object-oriented features at the core. In the last few years, *object-oriented analysis and design (OOAD)* has come into existence, it has found widespread acceptance in the industry as well as in academics. We concentrate here on widely accepted practices based on the use of the Unified Modeling Language (UML) to support an object-oriented development process [6]. The main reason for the popularity of OOAD is that it holds the following promises:

• Code and design reuse

• Increased productivity

• Ease of testing and maintenance

• Better code and design understandability

UML accomplish the visualization of software at early stage of SDLC, which helps in many ways like confidence of both developer and the end user on the system, earlier error detection through proper analysis of design and etc. UML also helps in making the proper documentation of the software and so maintains the consistency in between the specification and design document. The key advantage of this technique is that the test generation can systematically derive all combination of tests associated with the requirements represented in the model to automate both the test design and test execution process.

### 3.3 Problem Statement

To develop early test case generation strategy Our one of the main research objective is to make it possible for designer and developer to generate the test case at the time of requirement gathering & design phase. From this early information of test case failure & success reports the designer can easily remove the bugs & error at very early stages of project development. It improves quality of product in well defined standards. For this we are using the UML diagram which is part of design phase & flowchart which is a part of requirement gathering phase to extract the sufficient amount of test data. Later on the proposed reduction algorithm is applied to get better results.

To reduce the Test Suite size, complexity & cost here our aim is to develop a strategy which can be able to generate combinatorial test cases for variable parameters value which reduces efforts and cost. It also extracts what information is necessary to test the integration of components in the process of system composition;

To extracts test data from different design diagrams (UML) Here we investigate which individual or combination of UML diagram types, offer sufficient information to generate test cases; Also shows how the proposed strategy reports on the amount of testable information contained in a model.

### 3.4 UML Test Data Extraction Algorithm

Extracting the data from UML diagram is most difficult task. For this there is no such software developed which will extract the things from a png or jpeg format. It should be considered as new project. So we focus our main aim towards extracting all the useful information from a fixed textual format which is automatically generated from an well recognized tool PlantUML. It is a UML Based tool which takes the diagram as input and gives the respective textual notation for UML. To accomplish this reading we use pattern matching functions and operators or regular expression and then transform the values to a text file.

In our proposed strategy we had used activity diagram as a test model or a formal model. Here activity diagram is used to parse the information to generate test scenarios for various path available in it. Covering all the path will ensure that the maximum coverage is been achieved in it.

First of all our approach parses the activity diagram and generates the test scenarios which satisfy the path coverage criteria. As activity diagrams represent the implementation of an operation like the flow chart of code implementation and an executing path is a possible execution trace of a thread of a program, the executing paths are derived directly from the activity diagrams. We have considered path coverage in our approach, since it has the highest priority among all the coverage criteria for testing. Our approach also handles the complicacy of nested fork joins using a criterion that checks whether the target activity state of a transition is a fork or an activity state. If the target of the transition is a fork, then the fork has higher priority over the activity state. So it should be considered first and then only the other path is considered. As a result of this priority criterion the complicated nested fork-join pair is handled properly in our approach. After all the possible test scenarios are generated we generate the corresponding sequence diagram, and class diagrams for each scenario. Then test cases are then derived by finding significant values of environment conditions and parameters.

**The Proposed Algorithm for UML test data extraction (DDE):-**

**Algorithm:** DDE (Parameter, values)

1. Initialize the diagram as @startuml and assign it parameter 0, initial value is "Start"
2. Scan the file completely
3. Check for →" "
   If (pattern==found) consider it as next parameter and value in " " as its parameter value

   Repeat above till all identified

4. Now again scan for →< >
   If found read for next two [ ]

   For [Yes] add value in →" " as its value in above parameter

   For [No] add default value *

   Else for no value found consider both as parameter value.

5. Now check for York condition pattern
   If (===_===→" "equals to ===_===→" ")

   The above condition is consider as values of same parameter

6. Repeat till 5 until find →(*)
7. If(found== →(*))
   Then add next parameter and its value as "End"

8. Scan till all pattern matches
9. Exit

**Mathematical Expression**

**Result Achieved: -** We demonstrate our result by showing the improved performance MBTGA based on MTGIPO over other existing strategies. For this evaluation certain parameters is been identified and the result is been compared. These parameters are Reduced Test Size, Coverage, Time required, Complexity, Don't Care Conditions and last one is most important term possibility of early generation of test case.

To compare against other existing strategies we found that about 40% of the conditions of the program were usually covered by random test data generation, genetic approach covered 60% of the conditions and pairwise testing outperform former two by a considerable margin in most of our experiments. Genetic search achieved about 85% condition-decision coverage on average, while the random test-data generator consistently achieved just over 55%. So the pairwise testing strategy is proved to be an efficient test generating strategy. The following table shows the size of generated test set obtained by our technique as well as two other methods. Note that the size of test suite in case of pairwise strategy using MTGIPO is less than other tools and pairwise testing result obtained is in the form of Comparison Table's, Graphs, Utilities Functions, Features, Parameter Covered tables.

**Table: - Test Data 1**:- 5 parameter, 15 parameter values
**Method**:-Variable value

| Tool Name | Number of Pairs Covered in all Combinations | Test Case Generate | Coverage Achieved | Time Required | Don't care Conditions |
|---|---|---|---|---|---|
| ACTS | 270 | 40 | 0.254 | 0.015 sec | Not Count |
| PICT | NA | NA | NA | NA | NA |
| TCG | 270 | 43 | NA | NA | Not Count |
| **MBTGA** | **270** | **40** | **0.467** | **0.019 ml sec** | **Not Present** |

Table 1:-Comparison of MBTGA with other tools available for test data 1 for 3 way

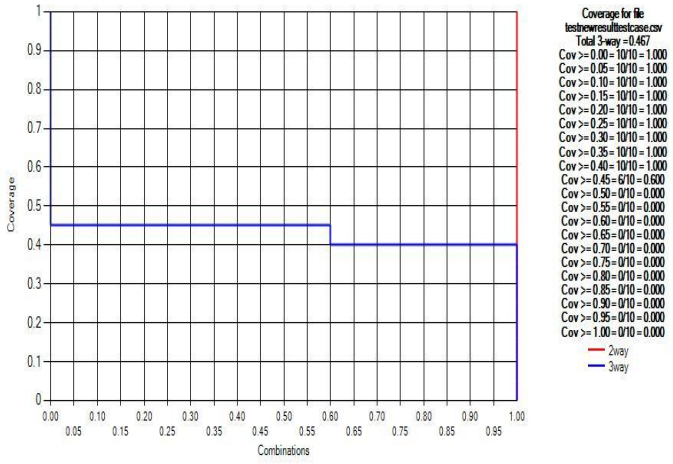
**Table Conclusion:-**The above table shows that the tool is capable of achieving reduction up to 85% and covers all pairs in just 40 test cases with 3 way interaction probability in just 0.019 mili second. It also shows that how the different tools perform under same conditions.

**Table 11:-Empirical study of MBTGA on the basis of various parameters**

| Data Extraction(DDE) | Pairs Covered | Reduced Test Cases | Tool Name MBTGA | Coverage Achieved | Time | Don't care | Comap-atibility |
|---|---|---|---|---|---|---|---|
| 6 Param, 9 values | 32 | 6 | Activity Diagram 1 | 1 | 2.609 ml sec | 0 | N/A In Any tool |
| 5 Param, 7 Values | 19 | 4 | Activity Diagram 2 | 1 | 0.683 ml sec | 0 | N/A |
| 7 Param, 9 Values | 34 | 4 | Activity Diagram 3 | 1 | 0.806 6 ml sec | 0 | N/A |
| 6 Param, 9 Values | 32 | 6 | Activity Diagram 4 | 1 | 0.849 ml sec | 0 | N/A |
| 11 Param, 15 Values | 101 | 6 | Activity Diagram 5 | 1 | 1.5089 ml sec | 0 | N/A |

**Table Conclusion: -** The above table shows an effective early test case generation feature of our tool which implies on the data extraction from UML activity diagram. We used our developed DDE algorithm for getting this result. We shows the performance and result evaluation of our tool on the basic of 7 parameters. Firstly DDE is been capable of extracting the correct data from given UML textual notations. The table shows how our tool effectively reduces the test size in very less time and gives maximum coverage. The feature which we have proposed and implemented is not present in any combinatorial testing tool and serves as add on module for our research. In future its improved versions are likely to be developed.

**Graph:**




**Graph Conclusion: -** The above graph is been generated from combinatorial coverage measurement tool by NIST. It shows the coverage achieved by our MBTGA tool based on MTGIPO and gives 100 % for 2 way (Red) and 0.467 for 3 way (Blue) interaction test, which is higher than any other available tool.

## 4. Conclusion & Future Work: -

The empirical study & the above chart easily prove that the strategy which we are proposing is effective & efficient for pairwise coverage problem. The test cases will 70 % reduced in most of the cases depending upon the seed value selected as per the given test criteria based on requirement. Future work will more effectively enhance the above proven results in a systematic way so as to generalize the tool. Also some researchers were focusing on improving solution domain of this genetic theory by NP complete & hard relations. So while proposing a new strategy in combination with pair wise approach always kept in mind its practical implementation so as to make the tester's work easy. MBTGA Tool is a great deal in test suite reduction & in addition it also provided maximum coverage based on genetic theory.

## 5. References:

[1] Pairwise Testing concept & strategy available from ;http://www.pairwise.org/tools.asp.

[2]B. Selic, and J. Rumbaugh, "UML for Modeling Complex Real Time Systems", Available Online Via www.rational.com/Products/Whitepapers/100230.Jsp.

[3]E. Holz, "Application of UML within the Scope of New Telecommunication Architectures", GROOM Workshop on UML, Mannheim:Physicaverlag, 1997.

[4]G. Booch, J. Rumbaugh and I. Jacobson, "The Unified Modeling Language User Guide", Addision Wesley, Reading, MA,1999.

[5]Clementine Nebut, Franck Fleurey, Yves Le Traon, "Automatic Test Generation: A Use Case Driven Approach", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 32,NO. 3, MARCH 2000

[6]A. Abdurazik, J. Offutt, and A. Baldini. "A controlled experimental evaluation of test cases generated from UML diagrams". Technical report, George MasonUniversity, Department of Information and Software Engineering, 2004.

[7]Lei, Y and Tai, K. C., In-Parameter-Order: A Test Generating Strategy for Pairwise Testing, IEEE Trans. on Software Engineering, 2002, 28 (1), 1-3.

[8]J.Bach and P. Shroeder. Pairwise testing - a best practice thatisn't. In Proceedings of the 22nd Pacific Northwest Software Quality Conference, pages 180– 196, 2004.

[9]Mandl, R., Orthogonal Latin Squares: An Application of Experimental Design to Compiler Testing, Comm. ACM, 1985, 28 (10), 1054-1058.

[10]Cohen, D. M., Dalal, S. R., Fredman, M. L., and Patton,G. C., The AETG system: An Approach to Testing Based on
Combinatorial Design, IEEE Transaction on Software Engineering, 1997, 23 (7), 437-443.

[11]Colbourn, C. and Dinitz, J. (Ed.) The CRC Handbook of Combinatorial Design, CRC Press, 1996.

[12]A. Andrews, R. France, S. Ghosh, and G. Craig. "Test adequacy criteria for UML design model. Software Test Verification and Reliability", 13:97–127, 2003.

[13]Stefania Gnesi, Diego Latella, and Mieke Massink. "Formal test- case generation for UML state charts". In ICECCS '04: Proceedings of the Ninth IEEE International Conference on Engineering Complex Computer Systems Navigating Complexity in the e-Engineering Age, pages 75–84, Washington, DC, USA, 2004. IEEE Computer Society.

[14]J. Hartmann, C. Imoberdorf, and M. Meisinger." UML-based integration testing". In ISSTA '00: Proceedings of the 2000 ACM SIGSOFT international symposium on Software testing and analysis, pages 60–70, New York, NY, USA, 2000.ACM.

[15]S. Helke, T. Neustupny, and T. Santen, "Automating Test Case Generation from Z Specifications with Isabelle," ZUM '97: The Z Formal Specification Notation, LNCS 1212, pp. 52-71.
J.P. Bowen, M.G. Hinchey and D. Till, eds. Springer-Verlag, 1997.

[16]Davis, Lawrence (1985). Job Shop Scheduling with Genetic Algorithms, *Proc. Int'l Conference on Genetic Algorithms* and their Applications.

[17]S. R. Dalal, A. Jain, N. Karunanithi,J. M. Leaton, C. M. Lott, G. C. Patton "**Model-Based Testing in Practice"** To appear in *Proceedings of ICSE'99, May 1999 (ACM Press).*

[18]P. E. Ammann and A. J. O_utt. "Using formal methods to derive test frames in category partition testing". In Ninth Annual *Conference on Computer Assurance (COMPASS'94),* Gaithersburg MD, pages 69–80,1994.

[19]P. E. Ammann and P. E. Black. "A specification-based coverage metric to evaluate test sets". In Proceedings of Fourth IEEE International High-Assurance Systems Engineering Symposium (HASE 99), pages 239–248. IEEE Computer
Society, November 1999. Also NIST IR 6403.

[20]P. E. Ammann, P. E. Black, and W. Majurski." Using model checking to generate tests from specifications". In Proceedings of the Second IEEE International Conference on Formal
*Engineering Methods (ICFEM'98),* pages 46–54. IEEE Computer Society, Dec. 1998