ISSN:0975-9646

Performance Analysis of Automated Sequence Model Testing (ASMT) using Cloud Integration

Priya Purohit
(Research Scholar, RGTU)
Dept. of CSE
JIT Vidya Vihar
Borawan(Khargone),MP(India)

Mahesh Malviya
(HOD)
Dept. of CSE
JIT Vidya Vihar
Borawan (Khargone),MP(India)

Yunus Khan
(Assistant Professor)
Dept. of CSE
JIT Vidya Vihar
Borawan(Khargone),MP(India)

Abstract-Software is a set of programs which provide the desired execution and reduces the human efforts. The code generated by the software must be check against the various rules and guidelines before delivering or deploying it. The approach which assures the correct functionality of the software deliverables are termed as testing methodology. Software testing is the most important aspect of SDLC as it looks into the behaviour of its functions and operations. Bu testing is applied after the complete development of code which sometimes gives changes which requires heavy test beds results in increased cost of the coverall system Thus there must be some way which identifies the bugs and errors before the complete development of the software. Thus, if there is any change then it must be accommodated into it during the development. Taking model into consideration in testing will provide early test case generation along with design verification, integration and configurations for path associated with the activity diagrams of the UML. Thus, the work focuses on generating the test case combinations using combinatorial approach with some specific functionality developed for getting effective results. Apart from the above solution, the work had also proved its applicability on the recent area of cloud. Integration of the ASMT approach with the cloud gives an enormous reduction in the execution time because of Map Reduce technology. It applies the criteria with generation modules to create novel test cases. The typical deployment of ASMT integrated with cloud held in five stages: setting up test criteria, test model designing, test suite creation with the help of Aneka cloud infrastructure, performing test & analyzing the result.

***Index Terms*— Cloud, Aneka, Model Based Testing (MBT), Selection Criteria, Test Suite Creation, UML, Automated Sequence Model Based Testing (ASMT);**

I. INTRODUCTION

Testing of the software requires more than half of the complete cost of software development. So it is a complex process & needs to be reduced by an automated test generation system One approach to do this would be to produce input data to the program to be tested program-based test data generation. The primary issue we face amid testing is dealing with the huge number of test cases we have to make and execute. 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 [1]. 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 modelling the system’s configurations space [2]. For better results test coverage criteria are additionally included in this automated component. It characterizes the rules used to produce test cases from the software model.

There are two sorts of criteria: data flow and control-flow. They characterize the effort and the nature of the results created automatically by an MBT approach. During the time a few endeavors in automatic test data generations have been made. The thought of path testing is to produce a rundown of test sets that capture all conceivable paths of component parameter values from every parameter [3]. It alludes to the process and techniques for the automatic deduction of abstract test cases from abstract formal models and designs, the creation of concrete tests from abstract tests, and the manual or automated execution of the ensuing concrete test cases. 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.

Model based testing (MBT) refers to the type of 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. 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 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 [6]. 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.

Integration with Cloud

In software engineering test case generation and evaluation is the main key to quality assurance. Analysis of test cases may give high quality software. In fact the process of test case generation and execution is complex task since it involves many iterations and decisions on the available set of parameters to choose appropriate set of test cases. Although automated test case generation process reduces the work but it also requires computation time and infrastructure. The advantage of automated process is only when it generate output efficiently and effectively. Integration of this process with cloud infrastructure may help to get better results on a large set of parameters as an input. Cloud composites of a powerful infrastructure based on map and reduce approach will definitely be having advantage over traditional approach to calculate the test cases.

II. RELATED STUDY

Literature gives prior views of code behavior after complete connections are established. Use of program paths to capture underlying program behavior is evidenced which try to achieve path coverage in test-suite construction. Research As we know that the program follows a path & constitutes a unit of interconnected modules to each other. It also gives the behavior of software codes. Thus to identify the bugs earlier before actual development starts, design diagrams need to be taken into consideration.. Hence, any method which covers various possible behaviors of a given program while avoiding path enumeration can be extremely useful for software testing. Various researchers had worked on the path based selection criteria for testing. Among them, test generation problem is deemed to be an important issue in software testing research [7]. Different approaches are developed to solve the above mentioned issues, few of them are PSO (Particle Swarm Optimization) [8], ACO (Ant Colony Optimization) [9], genetic algorithm [10] etc.

In the paper [11] data mining concept based tool generates a novel automated test case that is much superior, less complex and easier way. Where in this Tool, information from the UML Class diagram extracted and mapped, tree structure is formed with the help of those information's, Genetic Algorithm implemented as data mining technique, where Genetic crossover operator applied to discover all patterns and Depth First Search algorithm implement to Binary tree's formed to represent the knowledge i.e., test cases. In the paper [12] a novel approach for generating test cases of concurrent systems with the help of UML Sequence Diagram is shown. The approach consists of transferring the Sequence Diagram into a Concurrent Composite Graph (CCG). The CCG is traversed by an effective graph traversing technique like BFS (Breath-First-Technique) and DFS (Depth-First-search) using message sequence path criteria to generate the test cases for concurrent systems. The path coverage criterion is an important concept to be considered in test case generation is concerned [13]. In this work [14], gives first single model that is generic enough to study GUI and web applications together. It uses the model to define generic prioritization criteria that are applicable to both GUI and web applications. The ultimate goal is to evolve the model and use it to develop a unified theory of how all EDS should be tested. Evaluation results show the effectiveness of the approach in the correct manner.

This paper [15] deals with automatic generation of feasible independent paths and software test suite optimization using artificial bee colony (ABC) based novel search technique. In this approach, ABC combines both global search methods done by scout bees and local search method done by employing bees and onlooker bees. The parallel behavior of these three bees makes generation of feasible independent paths and software test suite optimization faster. Test Cases are generated using test path sequence comparison method as the fitness value objective function. The paper also presents an approach for the automated generation of feasible independent test path based on the priority of all edge coverage criteria. Finally, this paper compares the efficiency of ABC based approach with various approaches.

In 2012 Nirpal et. al. in [16] shows that the genetic algorithms can be used to automatically generate test cases for path testing. Using a triangle classification program as an example, experiment results show that Genetic Algorithm based test data can more effectively and efficiently than the existing method does. The quality of test cases produces by genetic algorithms is higher than the quality of test cases produced by random way because the algorithm can direct the generation of test cases to the desirable range fast. This paper shows that genetic algorithms are useful in reducing the time required for lengthy testing meaningfully by generating test cases for path testing.

The paper presents a novel approach to generate the automated test paths [17]. Due to the delay in the development of software, testing has to be done in a short time. This led to automation of testing because its efficiency and also requires less manpower. In this proposed approach, by using one of the most standard Unified Modeling Language (UML) Activity Diagram, construct the Activity Dependency table (ADT), then generate the Test paths. Then the test paths are prioritized by using the Tabular search algorithm. The prioritized test path can be used in system testing, regressing testing and integration testing.

The paper [18] gives an overview of Model based slicing, including the various general approaches and techniques used to compute slices. To understand and test a large software product is a very challenging task. One way to use this is program slicing technique that decomposes the large programs into smaller ones and another is a model based slicing that decomposes the large software

architecture model into smaller models at the early stage of SDLC (Software Development Life Cycle). From the given literature this has been listed out that for model based slicing techniques, there is the use of dependency relation, control and data flow, UML/OCL constraints, model language are present in literature with great emphasis on dependency relation.

An orchestrated survey of the most prominent techniques for automatic generation of software test cases, reviewed in self-standing sections proposed in [19]. The techniques presented include: (a) structural testing using symbolic execution, (b) model-based testing, (c) combinatorial testing, (d) random testing and its variety of adaptive random testing, and (e) search-based testing. Each section is contributed by world renowned active researchers on the technique, and briefly covers the basic ideas underlying the technique, the current state of art, a discussion of the open research problems, and a perspective of the future development in the approach. As a whole, the paper aims at giving an introduction, up-to-date and (relatively) short overview of research in automatic test case generation, while ensuring comprehensiveness and authoritativeness.

The paper [20] proposes a novel approach for diverse model based test case generation. It selects a subset of the generated test suite in such a way that it can be realistically executed and analyzed within the time and resource constraints, while preserving the fault revealing power of the original test suite to a maximum extent. In this article, to address this problem, we introduce a family of similarity-based test case selection (STCS) techniques for test suites generated from state machines. The paper also proposes a method to identify optimal tradeoffs between the number of test cases to run and fault detection.

III. PROBLEM DEFINITION

Software testing and test case generation mechanism is very time consuming. Also There accuracy and coverage must be maximum. If the designing of the test case fails then the testing methodology will misleads the evaluation of software's. Also the test automation along with its integration must have some predefined selection rules on the basis of which the test cases can be prioritize [21]. Test automation reduces the overall time for SDLC which serves high reliability and reduced cost. Also, the testing was applied in later stages of the project which sometimes detect those changes which only get completed in early stages of project or involves high modification cost associated with them. So we focused our study on Model Based testing approach for both test case generation and test case optimization to achieve some of the goal. Deficiency detection using test cases got from imprecise and ambiguous models could be extremely troublesome. Developing a model at the privilege level of abstraction for effective testing is one of the main difficulties for model-based testing [22]. Early generation of test case must be conceivable, however this model based testing yet extracting the data obliged an intermediate graph development will increase the trouble of cost and efforts. Automated test generation in model-based testing can rapidly generate an extensive number of test cases. Notwithstanding, the increase in test cases does not improve the quality of the test suite fundamentally and may compromise its efficiency. Normally, the effectiveness of a test suite is measured in terms of satisfying test requirements (i.e. Faults & coverage) and the efficiency is measured by the cost to attain the test requirements [23]. Here are some outlined objectives decided for the work is:

- To propose some generalized techniques to generate test cases for object-oriented software's using UML Activity Diagram diagrams.
- To propose a generalized technique for optimized and early test case generation using UML diagrams.
- To implement the proposed methods and evaluate their effectiveness for cloud computing.

IV. PROPOSED SOLUTION

The immediate purpose of this research is to design an ASMT Framework [24] integrated with cloud environment which provides 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 the suggested tool with testing modern software. Some are as follows:

- Automation of the test case generation and their execution.
- Development of domain and software engineering expertise needed 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).

The proposed work is parted in two identified domains: First is combinatorial test case optimization and second is design based test data generation (Early Generation). For this accomplishment of task we had proposed step by step solution. It gives the better result each time while comparing it with random test case generation. To formally draft the approach of automated sequence model based testing (ASMT) the work had also presented with design architecture. The main problem with testing is about managing the expansive number of automated test suite creation with smaller size & less complexity. Consequently, the work is focusing towards an automatic and effective test case handling concept taking in mind the early generation of test cases. The tool shows how diverse combinations of specification of system could be tested efficiently. It alludes to the process and techniques for the automatic derivation of abstract test cases from a formal model, the generation of concrete tests from abstract

tests, and the manual or automated execution of the resulting concrete test cases from proposed framework. The typical deployment of ASMT experiences five stages .It begins with setting up the test criteria for most astounding priority tests or to guarantee great coverage of the system behaviour. At that point we design a test model which speaks to the expected behaviour of the system under test (SUT), standard modeling language, such as UML are utilized to formalize the control points and perception points of the system, expected dynamic behaviour of the system. Next, for automated test suite creation, we apply all the above collected subtle elements to our next proposed enhanced ASMT based interaction algorithm for path oriented test generation. A detail of the approach along with the architectural description is given with the paper.

In this each generated abstract test case is typically a sequence of abnormal state SUT actions, with input parameters and expected yield values for each action of the test store is carried out by updating the test model. Later in the work accomplishes our test suite results with other existing approaches and apparatuses. After all the activities of automated test generation, we examine the result through a continuous system. The key concern will be on determining which combination of UML diagrams, and their associated constraints, may be utilized to automatically, or semi-automatically, generate test cases for pair wise & combinatorial testing.

ASMT Architecture Integrated With the Cloud

A typical deployment of ASMT goes through five stages starting from the test criteria settings and ended with result analyses. First step takes an infinite number of possible tests could be generated from a model. The test analyst chooses test generation criteria to select the highest priority tests or to ensure good coverage of the system behaviour. Secondly, model designing is performed to represents the expected behaviour of the system under test (SUT). Standard modeling languages, such as the Unified Modeling Language (UML) are used to formalize the control points and observation points of the system, the expected dynamic behaviour of the system. Now once the model is defined then the test creation is performed using cloud integration. This is an automated process that generates the required number of high-level (abstract) test cases from the test model using cloud infrastructure. Parameters collected from above steps are given as an input to the compute infrastructure, this action results in the test cases. Each generated abstract test case is typically a sequence of high-level SUT actions, with input parameters and expected output values for each action of the test repository is done by updating the test model, then automatically regenerating the test suites.

Next is to check generated test that are typically executed within a standard automated test execution environment, such as path wise interaction test tool. Alternatively, it is possible to execute tests manually – i.e. a tester runs each generated test on the SUT, records the test execution results, and compares them against the generated expected outputs. Either way, when the tests are executed on the SUT, we find that some tests pass and some tests fail. The failing tests indicate a discrepancy between the SUT and model, which need to be investigated to decide whether the failure is caused by a bug in the SUT. At the last result is analyzed against the real system which is to be tested and accepted by the user. The effectiveness of test cases can be evaluated using a fault injection technique called mutation analysis. Mutation testing is a process by which faults are injected into the system to verify the efficiency of the test cases. For this we are using pairwise approach whose problem domain is NP Complete, so the solution must be in accordance.

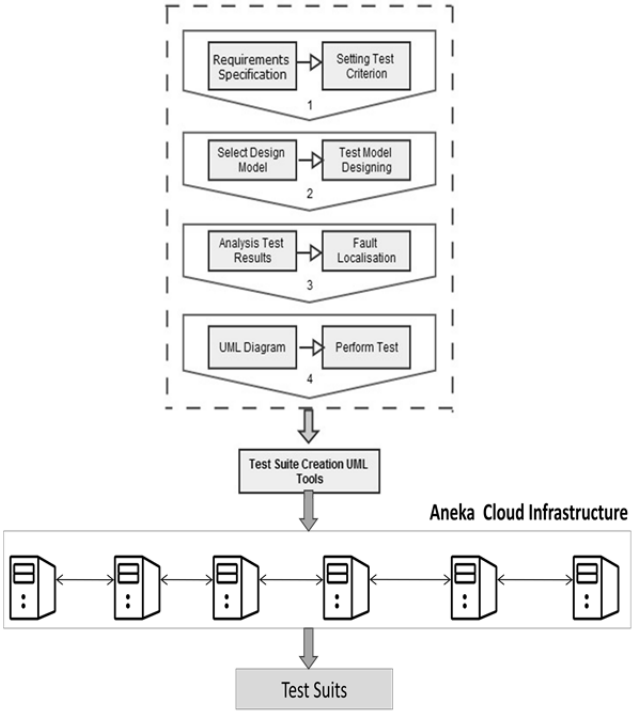


Figure 1: Design architecture of Proposed ASMT Model

The work and robust experimental analysis results presented in next section proves that the given design architecture ASMT is well defined for improving efficiency and performance through multiple parameters (Size, Time, Complexity, Cost etc.) and use of powerful cloud infrastructure. It is a well defined dynamic approach for quality improvements because it provides effective error detection at very low cost. ASMT with enhanced test generation integration with cloud is used to increase the performance of path interaction and model based testing in many aspects. It is intended that our strategies will convey to a designer, how much information is sufficient to enable automatic generation of test cases in an optimized manner. Automated test generation in model-based testing can quickly generate a large number of test cases.

V. RESULT ANALYSIS

The work demonstrates the results by showing the improved performance ASMT through cloud integration 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 it is 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 ASMT 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 Conclusion:-Table 1 takes the different sets of test with variable number of parameters and their values. Table gives an idea that how many possible combinations of test cases are possible and number of test cases selected by ASMT system which is giving 100% coverage in significantly lesser time, which saves lots of efforts of testers.

Table 1 shows the test data extraction from UML diagram. We used our developed UTDE algorithm to achieve this result. On the basis of 7 parameters we shows the performance and result assessment of UMBCA tool. UTDE is been capable of extracting the data from given UML diagram then convert it in to textual notations fetched by our algorithm. The table shows reduced test size in very short time and gives maximum coverage. The proposed feature implemented is not present in other combinatorial testing tool and serves as add on module for our research. In future its improved versions are likely to be developed.

Activity Diagram Coverage Status

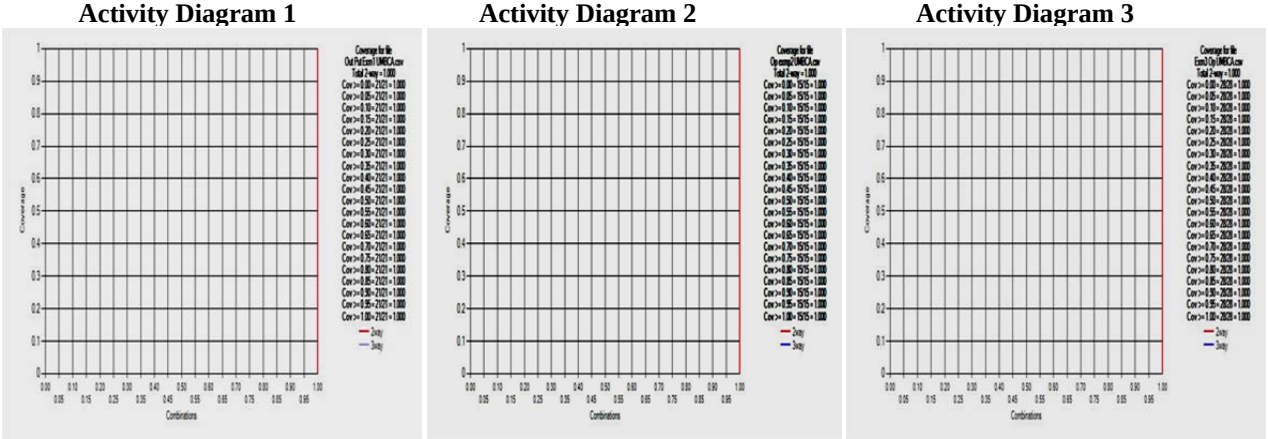
The below graph is used to analyze the coverage achieved by our tool for various activity diagrams (AD). Red one shows for 2 way interaction testing. The prime aim is achieved because the graph shows that the tool is been giving 100% coverage(Maximum) in all the cases of various types of activity diagrams including complex fork and joins with different decision conditions.

Graph 1 shows that we were getting the 100% coverage in case of activity diagram even after extracting the data from a different type of diagram from design phase. The above graph will also mention the details ratio of size and coverage.

Table-1: Generated Test Suites along with the Parametric Evaluation

Test-Sets	No. Parameters	No. Parameter Values	No. of Pairs Covered in all Combinations	No. Of Test Case Generated	Coverage Achieved	Time Required	Don't care Conditions
#1	5	15	90	13	1.00(100%)	1.5769 ml sec	0
#2	4	11	44	12	1.00(100%)	1.0527 ml sec	0
#3	5	48	921	118	1.00(100%)	47.937 ml sec	0
#4	7	20	166	20	1.00(100%)	4.221 ml sec	0
#5	4	8	24	6	1.00(100%)	0.809 ml sec	0

Activity Diagram 1 **Activity Diagram 2** **Activity Diagram 3**



Graph 1:- Coverage analysis Graph of ASMT for different Activity Diagrams (2-Way and 3-Way)

Benefits of Approach

Advantages of ASMT integrated with cloud over other UML based combinatorial approach is an innovative and high-value approach compared to more conventional functional testing approaches. The main expected benefits of ASMT integrated with cloud may be summarized as follows:

- Extracted the data from UML diagram to generate test cases.
- Generating the reduced number of test cases (Test Suite Size).

- Controlled Test Case Creation by Constraints & Relations with reduced test suites.
- Providing the maximum test coverage (100% for 2-way pairwise testing).
- Reducing the Test generation complexity.
- Independence from the test execution using cloud infrastructure.
- Powerful Map Reduce based cloud infrastructure tremendously reduces the time required to produce test suits.
- Systematic coverage of functional behavior;
- Definition of action words (UML model operations) used in different scripts;
- Efficient Test script generation;
- Generation of skeleton code for a library of automation functions;
- Because of cloud integration, independence from the test execution robot.

VI. CONCLUSION

The thought of UML-based pair wise testing is to utilize an explicit abstract model of a SUT and its environment to automatically infer tests for the SUT: the behaviour of the model of the SUT is interpreted as the intended behaviour of the SUT. The technology of the ASMT integrated with powerful cloud infrastructure test case generation has matured to the point where huge scale deployments of this technology are becoming commonplace. The prerequisites for success, such as qualification of the test team, integrated apparatus chain accessibility and methods, are presently identified, and an extensive variety of commercial and open-source tools are accessible. Despite the fact that ASMT won't take care of all testing problems, it is an important and valuable technique, which brings significant advancement over the state of the practice for functional software testing effectiveness, and can increase productivity and improve functional coverage.

REFERENCES

[1] Arilo C. Dias Neto, Rajesh Subramanyan, Marlon Vieira & Guilherme H. Travassos, "*A Survey on Model-based Testing Approaches: A Systematic Review*", in WEASELTech'07, November 5, 2007, Atlanta Georgia, USA, ACM, ISBN 978-1-59593-880-0/07, June 2007.

[2] Jon Edvardsson, "*A Survey on Automatic Test Data Generation*", in Proceedings of the Second Conference on Computer Science and Engineering in Linkoping, pages 21{28.ECSEL, October 1999.

[3] Renee C. Bryce, Ajitha Rajan & Mats P.E. Heimdahl, "*Interaction Testing in Model-Based Development: Effect on Model-Coverage*", in 13th Asia Pacific Software Engineering Conference (APSEC'06), ISBM-0-7695-2685-3/06, Aug 2007.

[4] Usman Farooq, Chiou Peng Lam & Huaizhong Li, "*Towards Automated Test Sequence Generation*", in Proceedings of 19th Australian Conference on Software Engineering ASWEC 2008 (pp. 441-450). Australia: Dec 2008.

[5] Robert M. Herons, "*Oracles for Distributed Testing*", in School of Information Systems, Computing, and Mathematics, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK, 2010.

[6] Suresh Thummalapenta, Saurabh Sinha, Debdoot Mukherjee & Satish Chandra, "*Automating Test Automation*", in Publication of IBM T.J. Watson Research Center, Sep 2011.

[7] X. Chen, Q. Gu, J. Qi and D.Chen, " *Applying Particle Swarm Optimization to Pairwise Testing*", in IEEE 34th Annual Computer Software and Applications Conference, ISBN No.0730-3157/10,Oct 2010.

[8] Praveen Ranjan Srivastava & Km Baby, "*Automated Software Testing Using Meta-heuristic Technique Based on An Ant Colony Optimization*", in International Symposium on Electronic System Design (ISED), ISBN: 978-1-4244-8979-4, pp 235 – 240, Dec 2010.

[9] Premal B. Nirpal & K. V. Kale, "*Using Genetic Algorithm for Automated Efficient Software Test Case Generation for Path Testing*", in Int. J. Advanced Networking and Applications, Volume: 02, Issue: 06, Pages: 911-915, 2011.

[10] Anuranjan Misra, Raghav Mehra, Mayank Singh, Jugnesh Kumar & Shailendra Mishra "*Novel Approach to Automated Test Data Generation for AOP*", in International Journal of Information and Education Technology, Vol. 1, No. 2, June 2011.

[11] Dawei Qi, Hoang D.T. Nguyen & Abhik Roychoudhury, "*Path Exploration based on Symbolic Output*" in Proceedings of ACM Conference, ESEC/FSE'11, Szeged, Hungary, ISBN 978-1-4503-0443-6/11/09, Sep 2011.

[12] Monalisha Khandai, Arup Abhinna Acharya & Durga Prasad Mohapatra, "*A Novel Approach of Test Case Generation for Concurrent Systems Using UML Sequence Diagram*", in IEEE Transaction, ISBN 978-1-4244-8679-3/11, Dec 2011.

[13] A. V. K. Shanthi & Dr. G. Mohankumar, "*Automated Test Case Generation For Object Oriented Software*", in Indian Journal of Computer Science and Engineering (IJCSE), ISSN : 0976-5166, Vol. 2 No. 4 Aug -Sep 2011.

[14] Renee C Bryce, Sreedevi Sampath & Atif M Memon, "*Developing a Single Model and Test Prioritization Strategies for Event-Driven Software*", in IEEE Transactions on Software Engineering, Vol. 37, No. 1, Jan 2011.

[15] Soma Sekhara Babu Lam, M L Hari Prasad Raju, Uday Kiran M & Swaraj Ch, "*Automated Generation of Independent Paths and Test Suite Optimization Using Artificial Bee Colony*", in International Conference on Communication Technology and System Design, Published by Elsevier Ltd, ISSN 1877-7058, 2012.

[16] Premal B. Nirpal & K. V. Kale, "*Comparison of Software Test Data for Automatic Path Coverage Using Genetic Algorithm*", in International Journal of Computer Science & Engineering Technology (IJCSET), ISSN : 2229-3345, Vol. 1 No. 1, Sep 2012.

[17] A.V.K. Shanthi & G. MohanKumar, "*A Novel Approach for Automated Test Path Generation using TABU Search Algorithm*", in International Journal of Computer Applications, ISSN 0975 – 888,Volume 48– No.13, June 2012.

[18] Rupinder Singh & Vinay Arora, "*Literature Analysis on Model based Slicing*", in International Journal of Computer Applications, ISSN 0975 – 8887, Volume 70– No.16, May 2013.

[19] Saswat Anand, Edmund Burke et. al., "*An Orchestrated Survey on Automated Software Test Case Generation*", in Journal of Systems and Software, Feb 2013.

[20] Hadi Hemmati & Andrea Arcuri, "*Achieving Scalable Model-Based Testing Through Test Case Diversity*", in ACM Transactions on Software Engineering and Methodology, Vol. 22, No. 1, Article, Feb 2013.

[21] J.Srinivas1, K.Venkata Subba Reddy, Dr.A.Moiz Qyser, " Cloud Computing Basics" published in International Journal of Advanced Research in Computer and Communication Engineering in Vol. 1, Issue 5, July 2012

[22] ManjraSoft white paper on "Developing MapReduce.Net applications using Aneka 2.0" published in OCT 2010.

[23] Mr. B.Suresh Kumar, Mr. Girish Paliwal ,Mr. Manish Raghav, Mr. Sudeep Nair, " Aneka As PaaS (Cloud Computing) " published in Journal of Computing Technologies 2012.

[24] Priya Purohit & Yunus Khan, "An Automated Sequence Model Based Testing for Reducing Test Suite Size Through Cloud Integration", in IJCSIT, Vol. 6, Issue 1, Jan 2015.