An Automated Sequence Model Testing (ASMT) for Reducing Test Suit Size

Ms. Priya Purohit
Department of Computer Science & Engineering
JIT, Vidhya-Vihar, Borawan, Khargon (M.P), India

ABSTRACT
Quality assertion of software is primarily done by means of testing an activity that faces constraints of both time & resources. Conventional testing strategies required the heavy resources & it can be done after the complete code is developed. So to apply some changes after the development requires lots of effort & cost. It also affects the time of deployment. So model based testing is a well accepted & dynamic approach for quality improvements because it provides affecting error detection at very low cost. It gives scalable & systematic solution to the test case reduction problem. To better understand the need of this early test case generation overall scenario of collaboration & development strategies is to be considered. The best way to get better knowledge of internal structure design models is the best option. Thus in this work we are focusing our research on path based testing by which internal structure of complete design can be verified & tested. Model based testing on path coverage is applied with standard coverage criteria for the large software's & generates larger test counts. Our aim is to reduce those test counts with maximum coverage of testing at the early phases of SDLC even before the actual development starts.

This paper proposes a novel automated sequence model based testing (ASMT) approach selects the test criteria based on UML diagram like activity, sequence, class etc. It applies the criteria with generation modules to create novel test cases. The typical deployment of ASMT goes through five stages: setting up test criteria, test model designing, test suite creation, performing test & analyzing the result. Thus by combining all the above methods improved test strategies can be designed. At the initial level of our research the approach seems to be better that others & will proves its effectiveness in future implementation.

Keywords
Model Based Testing (MBT), Selection Criteria, Test Suite Creation, UML, Automated Sequence Model Based Testing (ASMT);

1. INTRODUCTION
Testing of the software requires more than 50% of the complete cost of software development. So it is a complex process & needs to be reduced by some sort of automated test generation mechanism One way to do this would be to generate input data to the program to be tested program-based test data generation [1]. The main problem we face during testing is managing the large number of test cases we need to create and execute. One of the most important components in a testing environment is an automatic test data generator, a system that automatically generates test data for a given program. For better results test coverage criteria are also included in this automated mechanism. It defines the rules used to generate test cases from the software model. There are two types of criteria: dataflow and control-flow. They define the effort and quality of the results generated automatically by an MBT approach [2]. Through the years several attempts in automatic test data generations have been made. The idea of path testing is to generate a list of test sets that capture all possible paths of component parameter values from each parameter. We proposed a new strategy how the modelling of specification of system could be tested efficiently using automated sequence model based testing (ASMT). It refers to the process and techniques for the automatic derivation of abstract test cases from abstract formal models, the generation of concrete tests from abstract tests, and the manual or automated execution of the resulting concrete test cases.

UML is a powerful modelling language used to represent the research problems visually. A lot of literature is available for modelling problems by the use of UML, but limited research papers are reported in literature on applications of UML for the interaction path using sequence model problems [3]. By the use of UML, path based software testing problems can be solved and performance can be judged after modelling of the problem. We present the proposed UML based design process architecture which is a five phase model maturity used for automated sequence based path test suite creation methodology. The schemas will assess the diagrams in a model for sufficient test related information. The intention, of the proposed exploration, is not to impose restrictions upon the modelling process; however, it is intended that our strategies will convey to a designer, how much information is sufficient to enable automatic generation of test cases so as to reduce the number of test suites.

An objective of this exploration is to present a designer with confirmation that the diagrams in a system model include sufficient information for automatically generating a suite of test cases by sequence based test criteria generation [4]. There are two major aspects of the proposed study, which will be explored in five phase process. The first aspect relates to the testable information contained in a UML model; while the second aspect relates to the development of a technique to generate a test suite from the acquired diagram data. It can be considered as a test oracle problem which can be solved by distributing testing [5]. But a performance factor seems to be unaffected. Initially, as part of the first phase, we must determine what information is being collected by requirement gathering phase then select a proper design model for test case extraction. Once taxonomy of this generic information is established, we must determine which diagrams can provide the necessary information. In the second phase the navigational diagrams that offer this information are identified after that we will apply the test data on validation tool to clarify the maturity of test cases. The templates will form part of an application which produces a report on the amount and quality of the test related information contained in a model's diagram. Then we will explore which techniques might be suitable for the test

data extraction process. With this approach some of the available information will come from different diagram types. In some of the highly effective approaches for keyword driven & path oriented testing, backtracking is used. It explores the alternative flows, for automating manual tests in the context of keyword-driven automation [6]. Once all the information is being gathered we evaluate the result through injecting some fault in software & analyze our ASMT architecture for these fault detection Once the information is extracted we will develop a test suite format, that will enable the execution of test cases against the SUT. Finally, we then evaluate our technique for effectiveness and efficiency, against other random test case generation methodologies.

2. BACKGROUND

Testing software is an essential & complicated process of SDLC which is quite expensive. In research and industry the primary concern for the practitioners is to focus on finding automatic cost-effective software testing and debugging techniques. Maintaining high fault detection and localization ability will ensure high-quality software productions. Nowadays software testing research mainly concerns such issues as test coverage criterion design, test generation problem, test oracle problem, the regression testing problem and fault localization problems. Among these issues, the test generation problem is deemed to be an important issue in software testing research [7]. Various 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.

2.1. The Purpose of the Study

The proposed study will investigate and develop strategies and techniques to derive effective test cases from system-level, Automated Sequence Model Based Testing (ASMT). The focus will be on determining which combination of UML diagrams, and their associated constraints, may be used to automatically, or semi-automatically, generate test cases for path oriented interaction testing. Prototype tools will be developed in future to demonstrate the techniques and strategies derived from the proposed investigation.

In summary, the study aims to:

1. Determine what information is necessary to test the integration of components in the process of system composition;

2. Given item 1, investigate which individual or combination of UML diagram types, offer sufficient information to generate test cases; The results of this aim, will affect aspects of activities 1 to 5 in the figure

3. Develop a strategy that reports on the amount of testable information contained in a model.

4. Develop a UML based technique for information extraction based on the information required for component integration, from single and multiple UML diagram types;

5. Evaluate our overall strategy and techniques.

2.2. Why UML & path testing

UML based path testing approach is an innovative and high-value approach compared to more conventional functional testing approaches. The main expected benefits of ASMT may be summarized as follows:

- It gives QoS functional requirements:
- Complete test generation and testing coverage:
- The fully automatic process reduces the tester's efforts.

2.3. Understanding Testing

It is the process of identifying the bugs & errors to overcome the futuristic problems. It is a step by step process through which generates the test cases. At the start the test strategy is decided to identify the boundaries of testable information. After which the requirements need to be finalized. On the behalf of this information, estimations are made about the coverage, cost & efforts.

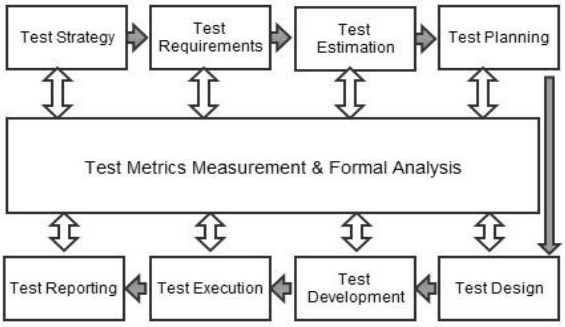

Figure 1: Test Development and Execution Process scheme.

Planning & estimation are done simultaneously. Test design is the next phase by which final test generation is measured which later on gives the outer. It gives the result which is formally analyzed & provides the needful in developing. After all the above process is completed execution & reporting is done for correctly generated test. The above process is developed & continuously assessed through few measurement parameters. Figure 1 shows the detailed process of test development & execution.

3. RELATED STUDY

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. It 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. 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. Few of them give their work as:

In 2011, Dawei Qi et. al. developed an approach for partitioning of program paths based on the program output [11]. Two program paths are placed in the same partition if they derive the output similarly, that is, the symbolic expression connecting the output with the inputs is the same in both paths. In this work the grouping of paths is gradually created by a smart path exploration. An experimental result shows the benefits of the proposed path exploration in test-suite construction.

Monalisha Khandai et. al. in 2011 presents a novel approach of generating test cases for concurrent systems with the help of UML Sequence Diagram [12]. 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 proposed approach is applied to concurrent systems for test case generation and found to be very effective in controlling the test case explosion problem. The generated test cases are useful to detect interaction, scenario, as well as operational faults in case of concurrent systems.

In 2011, G. Mohankumar et. al. Proposes a data mining concepts that are designed and used to generate test cases. The Tool generates a novel automated test case that is much superior, less complex and easier to implement in any Testing system. 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. The path coverage criterion is an important concept to be considered in test case generation is concerned [13].

In this work [14], Bryce et. al. provides the 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. Threats to construct validity are factors in the study design that may cause us to inadequately measure concepts of interest. The study made simplifying assumptions in the area of costs. In this the author also measures the test suite prioritization. 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.

This 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 Modelling Language (UML) Activity Diagram, construct the Activity Dependency table (ADT), then generate the Test paths. Then the test paths are prioritized by using the Tabu search algorithm. The prioritized test path can be used in system testing, regressing testing and integration testing. Then also from the Cyclomatic diagram to check the efficiency of the test scenario.

In 2013, literature analysis by Rupender & Vinay et. al. present in [18] 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.

In 2013, 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.

In 2013 Hemmati, Archuri & Briand et. al. 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.

4. PROBLEM IDENTIFICATION

Model based software testing generates test cases based on models of the specifications. Models preserve the essential information from requirement specification and are base for the final implementation. However, in order to generate complete and effective test cases for functional and system testing, behavioral models are necessary. Therefore, in MBT, behavioral models are used at the start in order to determine the valid test scenarios of a system from which the relevant test cases are then selected. The UML Sequence Diagram is one of several behavioral diagrams in UML with particular strengths in modelling the object and control flows aspects of a system. One of the key features of the AD is the built-in modelling support for concurrency and synchronization. It can specify multiple sequences of operations executing concurrently and control their execution order with built-in fork and join constructs.

Though, there is lots of modelling language such as UML, SDL, Z- Specification, we will use UML as a test model, which is a semi formal modelling language. Fault detection using test cases derived from imprecise and ambiguous models could be very difficult. Developing a model at the right level of abstraction for effective testing is one of the main challenges for model-based testing. Early generation of test case can only be possible though this model based testing but extracting the data required an intermediate graph construction will increase the burden of cost and efforts. Automated test generation in model-based testing can quickly generate a large number of test cases. However, the increase in test cases does not improve the quality of the test suite necessarily and may compromise its efficiency. Usually, 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 achieve the test requirements.

The problem identified with this approach is that multiple object accessing the same function at the same time cannot be handled .It also not gives any of the priority to user for the high priority test case. Also for the above described methodical construction of composite graph is necessary which seems to restrict us and also increases an overhead.

In summary, the research aims:

(i) To understand the automatic Test Case Generation Process

(ii) To develop early test case generation strategy

(iii) To reduce the Test Suite size, complexity & cost

(iv) To extract test data from different design diagrams (UML)

Considering the above mentioned issues this work proposes a new Automated Sequence Model Based Testing (ASMT) strategy.

5. PROPOSED APPROACH

The main problem with testing is about managing large number of automated test suite creation with smaller size & less complexity. Thus we are focusing on automatic and effective test case handling concept taking in mind the early generation of test cases. To accomplish the above basic requirements we proposed new design architecture of Automated Sequence Mode Based Testing (ASMT) to accomplish how different combinations of specification of system could be tested efficiently. It refers 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 goes through five stages .It starts with setting up the test criteria for highest priority tests or to ensure good coverage of the system behavior. Then we design a test model which represents the expected behavior of the system under test (SUT), standard modelling language such as UML are used to formalize the control points and observation points of the system , expected dynamic behavior of the system. Next, for automated test suite creation we apply all the above collected details to our next proposed enhanced ASMT based interaction algorithm for path oriented test generation.

In this 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. Later in the work accomplishes our test suite results with other existing approaches and tools. After all the activities of automated test generation we analyses the result through a real time system. The key concern will be on determining which combination of UML diagrams, and their associated constraints, may be used to automatically, or semi-automatically, generate test cases for pairwise & combinatorial testing.

5.1. ASMT Architecture

A typical deployment of ASMT goes through five stages

Step-I: Setting Up Test Criteria
Usually 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 behavior. One common kind of test generation criteria is based on structural model coverage, using well known test design strategy of path based testing. Another useful kind of test generation criteria ensures that the generated test cases cover all the requirements, perhaps with more tests for requirements that have a higher level of risk so in this paper we are combining the pathwise approach with new architecture through UML Navigational approach.

Step-*II*: Test Model Designing
The model, generally called the test model, represents the expected behavior of the system under test (SUT). Standard modelling languages, such as the Unified Modelling Language (UML) are used to formalize the control points and observation points of the system, the expected dynamic behavior of the system.

Step-III: Test Suite Creation
This is an automated process that generates the required number of high-level (abstract) test cases from the test model. 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.

Step-IV: Perform Tests
Generated concrete tests are typically executed within a standard automated test execution environment, such as pathwise 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.

Step-V: Analyzing Result
Analyze 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

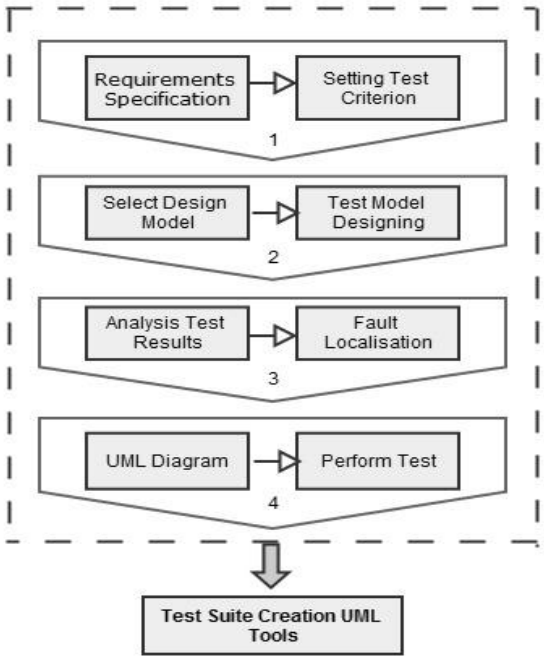

Figure 2: Design architecture of Proposed ASMT Model

ASMT with enhanced test generation 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.

In our later research results we prove that the given design architecture ASMT is well defined for improving efficiency and performance through multiple parameters (Size, Time, Complexity, Cost etc.). It is a well defined dynamic approach for quality improvements because it provides effective error detection at very low cost

6. EXPECTED BENEFITS

Expected benefits of ASMT 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 may be summarized as follows:

Contribution to the quality of functional requirements:

- Modelling for test generation is a powerful means for the detection of "holes" in the specification (undefined or ambiguous behavior).
- Independence from the test execution robot.

Contribution to test generation and testing coverage:

- Automated generation of test cases;
- Systematic coverage of functional behavior;
- Automated generation and maintenance of the requirement coverage matrix;
- Continuity of methodology (from requirements analysis to test generation).

Contribution to test automation:

- Definition of action words (UML model operations) used in different scripts;
- Test script generation;
- Generation of skeleton code for a library of automation functions;
- Independence from the test execution robot.

7. CONCLUSION

The idea of UML-based pairwise testing is to use an explicit abstract model of a SUT and its environment to automatically derive tests for the SUT: the behavior of the model of the SUT is interpreted as the intended behavior of the SUT. The technology of the ASMT test case generation has matured to the point where large-scale deployments of this technology are becoming commonplace. The prerequisites for success, such as qualification of the test team, integrated tool chain availability and methods, are now identified, and a wide range of commercial and open-source tools are available. Although ASMT will not solve all testing problems, it is an important and useful technique, which brings significant progress over the state of the practice for functional software testing effectiveness, and can increase productivity and improve functional coverage.

8. REFERENCES

[1] 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.

[2] 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.

[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.