ISSN:0975-9646

An Automated Sequence Model Testing (ASMT) For Improved Test Case Generation Using Cloud Integration

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

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

Abstract - **Software quality assurance is primarily done by means of testing an activity that faces constraints of both time & resources. Traditional testing strategies obliged the overwhelming resources and infrastructure and it can be carried out after the completion code is created. So to apply some changes after the development obliges loads of effort & cost. It likewise affects the time of deployment. So model based testing is a generally accepted & dynamic approach for quality improvements because it gives affecting mistake detection easily. It gives scalable & systematic solution to the test case reduction problem. To better understand the need of this early test case generation general scenario of collaboration & development strategies is to be considered. The most ideal approach to improve learning of internal structure design models is the best choice. In this manner, in this work we are focusing our research on the path based testing by which internal structure of complete design can be confirmed & tested. Model based testing on path coverage is connected with standard coverage criteria for the expansive software's & generates bigger 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 begins.**

This paper proposes a novel automated sequence model based testing (ASMT) integrated with the cloud; this approach selects the test criteria based on UML diagram like activity, etc... 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 & analysing the result. Accordingly, 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 demonstrate its effectiveness in future implementation.

Keywords— 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 [2]. The primary issue we face amid testing is dealing with the huge number of test cases we have to make and execute. A standout amongst the most imperative components in a testing environment is an automatic test data generator, a system that automatically produces test data for a given program. 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 [1]. During the time a few endeavours 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. We proposed another strategy, how the modelling of determination of system could be tested proficiently utilizing automated sequence model based testing (ASMT) integrated with the cloud. 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.

UML is a capable modelling language used to speak to the research problems outwardly. A ton of literature is accessible for modelling problems by the utilization of UML; however constrained research papers are accounted for in literature on applications of UML for the association path utilizing sequence model problems [3]. By the utilization of UML, path based software testing problems can be fathomed and performance can be judged in the wake of modelling of the issue. We show the proposed UML based design process architecture which is a five phase model development utilized for automated sequence based path test suite creation approach. The schemas will evaluate the diagrams in a model for sufficient test related information. The aim, of the proposed investigation, is not to force restrictions upon the modelling process; notwithstanding, it is planned that our procedures will convey to a designer, the amount of information is sufficient to empower an automatic generation of test cases in order to reduce the quantity of test suites.

A target of this exploration is to present a designer with confirmation that the diagrams in a system model to incorporate sufficient information for automatically creating a suite of test cases by sequence based test criteria generation [4]. There are two noteworthy aspects of the proposed study, which will be investigated in five phase process. The primary perspective identifies with the testable information contained in an UML model; while the second angle identifies with the development of a technique to create a test suite from the procured graph data. It can be considered as a test oracle issue which can be tackled by distributing testing [5]. In any case a performance factor is by all accounts unaffected. At first, as a component of the first phase, we must figure out what information is generally collected by requirement social occasion phase then select a fitting design model for test case extraction. Once the taxonomy of this generic information is created, we must figure out which diagrams can give the necessary information.

In the second phase the navigational diagrams that offer this information are recognized after that we will apply the test data on approval apparatus to clear up the maturity of test cases. The templates will structure some piece of an application which creates a report on the amount and quality of the test related information contained in a model's outline. At that point we will explore which techniques may be suitable for the test data extraction process. With this approach a percentage of the accessible information will originate from distinctive outline sorts. In a percentage of the exceptionally viable approaches for keyword driven & path oriented testing approach, utilization of backtracking main focus. It explores the alternative flows, for robotizing manual tests in the connection of keyword-driven automation [6]. The moment, all the information is generally assembled, we evaluate the result through injecting some deficiency in software & analyse our ASMT integrated with cloud , architecture for these flaw detection Once the information is concentrated we will create a test suite organize, that will empower the execution of test cases against the SUT. At long last, we then evaluate our technique for effectiveness and efficiency, against other random test case generation systems.

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. Industry the essential concern for the practitioners is to concentrate on finding automatic cost-effective software testing and debugging techniques. Keeping up high fault detection and localization capacity will ensure top notch software productions. Nowadays software testing, research principally 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]. 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.

(A) The Purpose of the Study

The focus of study will be to investigate and develop strategies and techniques to derive effective test cases from system-level, Automated Sequence Model Based Testing (ASMT) along with the execution time detection and effort to reduce the time involved in calculation of test suites. 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:

a) Determine what information is necessary to test the integration of components in the process of system composition;
b) 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
c) Develop a strategy that reports on the amount of testable information contained in a model.
d) Develop a UML based technique for information extraction using power and efficient cloud infrastructure based on the information required for component integration, from single and multiple UML diagram types;
e) Evaluate our overall strategy and techniques.

(B) 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.

(C) Why cloud integration?

In software engineering test case generation and evaluation is the main key to quality assurance. Analysis of test cases may give high quality softwares. 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.

- **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.

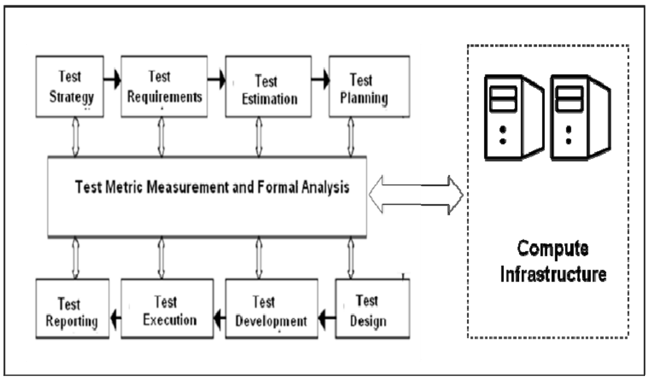

Figure 1: Test Development and Execution Process scheme.

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 analysed & 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.

- **Understanding Aneka Cloud and Map Reduce**

Aneka permits servers and desktop PCs to be connected together to structure an effective computing infrastructure. Aneka is a workload distribution and administration stage that quickens applications in Microsoft .NET framework environments. It disentangles the improvement of distributed applications by giving: an accumulation of diverse routes for communicating the rationale of distributed applications, a solid infrastructure that deals with the distributed execution of applications, and a set of cutting edge peculiarities, for example, the capacity to hold and value reckoning hubs and to incorporate with existing cloud infrastructures, for example, Amazon Ec2[23].Aneka gives a flexible and extensible environment which runs numerous applications all the while and backings complex models and conditions inside those applications.

- **Map Reduce Overview**

Map Reduce is triggered by map and reduce operations in functional languages, such as Lisp. This model abstracts computation problems through two functions: map and reduce. All problems formulated in this way can be parallelized automatically. All data processed by Map Reduce are in the form of key/value pairs. The execution happens in two phases. In the first phase, a map function is invoked once for each input key/value pair and it can generate output key/value pairs as intermediate results. In the second one, all the intermediate results are merged and grouped by keys. The reduce function is called once for each key with associated values and produces output values as final results [22].

- **Map and Reduce**

A map function takes a key/value pair as input and produces a list of key/value pairs as output[22]. The type of output key and value can be different from input key and value:

map::(key1,value1) => list(key2,value2)

A reduce function takes a key and associated value list as input and generates a list of new values as output:

reduce::list(key2,value2) => list(value3)

- **Map Reduce Execution**

A Map Reduce application is executed in a parallel manner through two phases. In the first phase, all map operations can be executed independently with each other. In the second phase, each reduce operation may depend on the outputs generated by any number of map operations. However, similar to map operations, all reduce operations can be executed independently. From the perspective of dataflow, Map Reduce execution consists of m independent map tasks and r independent reduce tasks, each of which may be dependent on m map tasks. Generally the intermediate results are partitioned into r pieces for r reduce tasks. The Map Reduce runtime system schedules map and reduce tasks to distributed resources. It manages many technical problems: parallelization, concurrency control, network communication, and fault tolerance. Furthermore, it performs several optimizations to decrease overhead involved in scheduling, network communication and intermediate grouping of results.[22]

II. RELATED STUDY

Literature gives prior views of code behaviour after complete connections are established. Use of program paths to capture underlying program behaviour 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 behaviour 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 behaviours 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, 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].

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, 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.

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.

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 Tabular 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 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.

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.

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.

III. 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, behavioural models are necessary.

Therefore, in MBT, behavioural 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 behavioural 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.

However, there is bunches of modelling language, for example, UML, SDL, Z- Specification, we will utilize UML as a test model, which is a semi formal modelling language. 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. 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.
In summary, the research aims:

1. To understand the automatic Test Case Generation Process
2. To develop early test case generation strategy
3. To use Aneka cloud infrastructure to evaluate the best test case set from the given set of parents
4. To reduce the Test Suite size, complexity & cost
5. 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. 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 use 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.

IV. PROPOSED APPROACH

The main problem with testing is about managing the expansive number of automated test suite creation with smaller size & less complexity. Consequently, 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 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 modelling 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.

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
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 behaviour. 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 path wise approach with new architecture through UML Navigational approach.
Step-II: Test Model Designing

The model, generally called the test model, represents the expected behaviour 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 behaviour of the system.
Step-III: Test Suite Creation 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.

Step-IV: Perform Tests

Generated concrete tests 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.

Step-V: Analysing Result

Analyse 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.

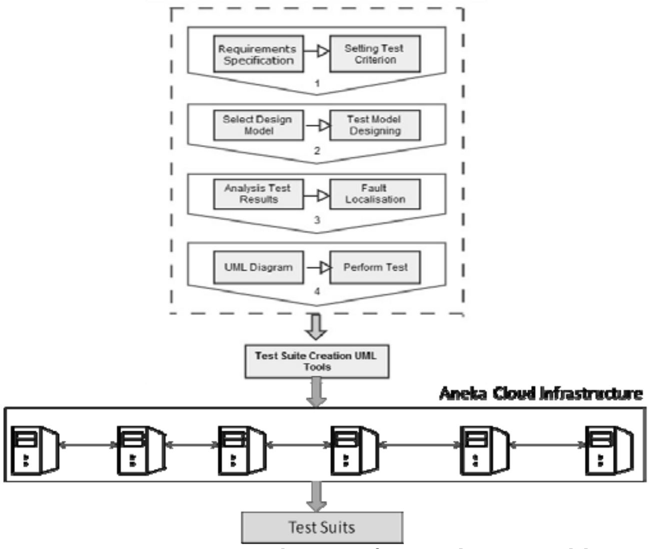

Figure 2: Design architecture of Proposed ASMT Model

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.) 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. Expected Benefits

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:

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 behaviour).
- Independence from the test execution robot.

Contribution to test generation and testing coverage:

- Powerful Map Reduce based cloud infrastructure tremendously reduces the time required to produce test suits.
- Automated generation of test cases;
- Systematic coverage of functional behaviour;
- 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;
- 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.