RESEARCH ARTICLE OPEN ACCESS

A Literature Survey for Test Case Generation Using UML Model

Md Khalid Hussain [1], Dr. Krishna Nandan Prasad [2]
Research Scholar [1]
Inst. of Information Sciences & IT M. U. Bodh-Gaya, Gaya
Associate Professor [2]
A.N. College, Patna
Bihar - India

ABSTRACT
Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. The success of software can be improved by modeling .Modeling is one way to visualize the design and check it against the requirement before the developer begins the coding phase. One such modeling language widely used is unified modeling language (UML). UML diagrams produce a graphical representation of the systemunder design.

Keywords :- Introduction UML diagram, Activity diagram, sequence diagram, different test case generation.

I. INTRODUCTION

Software engineering is also about communication on a team and with internal and external stakeholders. Teams do not consist only of developers, but also of quality assurance testers, systems architects, system/platform engineers, customers, project managers, and other stakeholders. Implementation is no longer just writing code, but it is also following guidelines, writing documentation, and also writing unit tests. But unit tests alone are not enough. The different pieces have to fit together. In addition, we have to be able to spot problematic areas using metrics and improve the quality in those areas. The code should follow certain standards to make it easier for a team to work together. Once we are finished coding, that does not mean that we are finished with the project: for large projects, maintaining software and other maintenance can keep many people busy for a long time. Software engineers speak a funny language called Unified Modeling Language, or UML for short. As if a musician has to learn musical notation before being able to play piano, we need to learn UML before we are able to engineer software. UML is useful in many parts of the software engineering process, for instance: planning, architecture, documentation, or reverse engineering. Therefore, it is worth our efforts to know it. Designing software, is a little like writing a screenplay for a Hollywood movie? The characteristics, actions, and interactions of the characters are carefully planned, as is the relevant components of their environment.

As an introductory example, consider our friend Bill, a customer, who is at a restaurant for dinner. His waiter is Linux, who takes the orders and brings the food. In the kitchen is Larry, the cook. Steve is the cashier. In this way, we have provided useful and easily accessed information about the operation of a restaurant in the screenplay.

1.1 Case Diagram

The use case model [1] is representation of the systems intended functions and its environment. The first thing a software engineer does is to draw a Use Case diagram all the actions are represented by ovals and are called use cases. Lines connect the actors and use cases. Very often, there is also one or more system boundaries. Actors, which usually are not part of the system, are drawn outside the system area. Use Case diagrams are very simple, so even managers can understand them. However, they are very helpful in understanding the system to be designed. They should list all parties involved in the system and all major actions that the system should be able to perform. The important thing about them is that you do not forget anything, it is less important that they are super-detailed, for this, we have other diagram types.

1.2 Activity Diagram

The Activity diagram gives more detail to a given use case and it often depicts the flow of information, hence it is also

called a Flowchart. Where the Use Case diagrams have no timely order, the Activity diagram has a beginning and an end, and it depicts decisions and repetitions.
If the Use Case diagram names the actors and gives us the headings for each scene (use case) of our play, the Activity diagram tells the detailed story behind each scene. Some managers may be able to understand Activity diagrams, but do not count on it.

1.3 Sequence Diagram

Once we are done drawing our Activity diagrams, the next step of refinement is the Sequence diagram. In this diagram we list the actors or objects horizontally and then we depict the messages going back and forth between the objects by horizontal lines. Time is always progressing downwards in this diagram.
The Sequence diagram is a very important step in what is called the process of object-oriented analysis and design. This diagram is so important, because on the one hand, it identifies our objects/classes and on the other hand, it gives us the methods for each of those classes, because each message turns into a method. Sequence diagrams can become very large, since they describe the whole program. Make sure, you cover every path in your Sequence diagrams, but try to avoid unnecessary repetition. Managers will most likely not understand Sequence diagrams.

1.4 Collaboration Diagram

The Collaboration diagram is an intermediate step to get us from the Sequence diagram to the Class diagram. It is similar to the Sequence diagram, but it has a different layout. Instead of worrying about the timeline, we worry about the interactions between the objects. Each object is represented by a box, and arrows show interactions between the objects.This diagram shows the responsibility of objects. If an object has too much responsibility, meaning there are too many lines going in and out of a box, probably something is wrong in your design. Usually you would want to split the box into two or more smaller boxes. At this stage in your design, this can still be done easily. Try to do that once you started coding, or even later, it will become a nightmare.

1.5 Class Diagram

The Class diagram is the most important one. A Class diagram consists of classes and lines between them. The classes themselves are drawn as boxes, having two compartments, one for methods, and one for attributes.
You start with the Collaboration diagram, and the first thing you do is take all the boxes and call them classes now. Next, instead of having many lines going between the objects, you replace them by one line. However, for every line you remove, you must add a method entry to the class's method compartment. Therefore, at this stage the Class diagram looks quite similar to the Collaboration diagram.

II. TEST CASE GENERATION APPROACH

This is the main part of the testing process. The Approaches involved in generating test cases can be categorized in these three parts : Scenario Based Test Case Generation, Model based test case Generation and Genetic Based test Case Generation .Even though variety of approaches have been proposed yet for a decade there has been constant research on generating test cases based on specification and design models.

2.1 Scenario based Test Case Generation: In Scenario based test case generation test scenarios are used for generating test cases .Baikunt Narayan Biswal has presented a paper. A Novel Approach for Scenario based test case Generation [2]. This paper deals with Test adequacy Criteria for complex transactions or Events, Scenario based testing gives best results. Test case generation UML Activity diagrams presented by Kim are also based on concurrency in Activity Diagram where multiple systems interact with each other.

2.2 Model based Test Case Generation: In model based testing, the testing begins at design phase. So, early detection of faults can be achieved by using this approach further reducing time, cost and efforts of the developer to a large extent. Automatic Test case generation using Unified Modeling Language (UML) state diagram by P.samuel and A.K.Bothra and Rajib Mall published on the basis of Model Based Test Case generation.Test Case Generation by UML Sequence Diagram and labeled Transition System. The procedure is based on the model based testing techniques with test cases generated from UML Sequence diagram converted into Labeled Transition System(LTS).Test Case Generation Based on Use Case and Sequence Diagram by Santosh Kumar Swain, Durga Prasad Mohapatra and Rajib Mall . Test cases are derived from a model based on systemGraph integrating UDG and SDG.

2.3 Genetic based Test Case Generation: In Genetic based test case generation technique, the test cases are generated using Genetic Algorithm. Improving GA based Automated Test Data Generation Technique For Object Oriented Software [3] by Nirmal Kumar Gupta, Mukesh Kumar Rohil,. The proposed strategy shows that genetic algorithms are useful in reducing the number of unfeasible test cases by generating test cases for object-oriented software. A Hybrid Genetic Algorithm Based Test Case Generation Using

Sequence Diagrams [4] by Mahesh Shirole, Rajeev Kumar. Test cases generated using genetic algorithm improves the method coverage as well as exception coverage. Object Oriented Test Case Generation Technique using Genetic Algorithms [5] by V.Mary Sumanlatha, G.S.V.P.Raju,. Test cases are generated using sequence Diagram and optimized using Genetic Algorithm.

III. LITERATURE SURVEY

Software developers cannot test everything, but they can use combinatorial test design to identify the minimum number of tests needed to get the coverage they want. Combinatorial test design enables users to get greater test coverage with fewer tests. Whether they are looking for speed or test depth, they can use combinatorial test design methods to build structured variation into their test cases. In survey it has been found that these test cases can be generated with different techniques like test case generation using UML models like Activity Diagrams, Sequence Diagrams etc. The process of generating test cases from design will help to discover problems early in the development process and thus it save time and resources during development of the system. However, it is very difficult to select test cases from UML models. In UML, the behavior of a use case can be represented by using interaction, activity and state machine diagrams. Sequence diagrams capture the exchange of messages between objects during execution of a use case. It focuses on the order in which the messages are sent. Activity diagrams, on the other hand, focus upon control flow as well as the activity-based relationships among objects. These are very useful for visualizing the way several objects collaborate to get a job done. Ranjita Kumari Swain, Vikas Panthi and Praful Kumar Behera, "Generation of test cases using Activity Diagram" have generated the test cases using activity diagrams [6]. In that approach, first an activity flow graph is derived from activity diagram. Then, all the required information is extracted from the activity flow graph (AFG). The activity flow graph (AFG) for the activity diagram is created by traversing the activity diagram from beginning to end, showing choices, conditions, concurrent executions, loop statements. From the graph different control flow sequence are identified by traversing the AFG by depth first traversal technique. Next, an algorithm is proposed to generate all activity paths. Finally, test cases are generated using activity path coverage criteria. Then Case study is being presented on Soft drink Vending Machine (SVM). Abinash Tripathy and Anirban Mitra "Test Case Generation Using Activity Diagram and Sequence Diagram" , presented an approach to generate test cases by using together UML Activity diagram and Sequence Diagram [7]. In this approach first the activity diagram is being converted into activity graph (AG) and the sequence diagram is being converted into sequence graph (SG) and then the two graphs SG & AG are integrated to form system Graph (SYG). Then the System Graph (SYG) is being traversed to form the test cases by using an Graph optimization technique known as Depth First Search Method (DFS).This approach is also applied on an example of ATM card validation. It has been shown that the test cases obtained in this method are not only exhaustive but also optimal but how it is not clear. Also whenever two UML methods are combined it will cover all the possibilities. Activity diagram also solves the problem of concurrent execution problem which leads to state explosion problem. Supaporn Kansomkeat, Phachayanee Thiket and Jeff Offutt "Generating Test Cases from UML Activity Diagrams using the Condition-Classification Tree Method" have focused on a UML diagram called an activity diagram [8]. They have used Condition Classification Tree method to generate test cases from activity diagram. In their paper, they provided a method to automatically gather control flow information from decision points and guard conditions in activity diagrams. This information is used to construct condition-classification trees. These trees are then used to generate a test case table and test cases. Experimental data show that tests generated by the CCTM Method have strong ability to detect faults at reasonable cost and also early in development. The case study is being done on SALES example. Xiajiong Shen and Qian Wang PeipeiWang and Bo Zhoupropose, "A Novel Technique Proposed for Testing of Object Oriented Software Systems", have proposed a novel technique [9] for testing of object oriented software systems that use the profile and UML state diagrams that all methods in a system are divided into different grades according to integrated values (frequency and significance) and then the methods that obtain the highest integrated value generate test cases from UML state diagrams. To prove the efficiency of the approach, the technique is compared with testing all methods only using UML state diagrams. Finally results prove that the approach is efficient for object-oriented software systems, and find faults that are difficult to find in other ways and reduce the cost of testing dramatically. Fanping Zeng, Zhide Chen, Qing Cao, Liangliang Mao, "Research on Method of Object-Oriented Test Cases Generation Based on UML and LTS"[10] have presented a new technique for object oriented test case generation based on UML state diagrams and Label Transition System(LTS). Test cases are generated from UML state diagrams model that represent state transition. UML state diagram can be a model of modeling software system. It shows all kinds of possible states of a specific object and all the possible changes between states which cause by all kinds of events. Labeled Transitions System (LTS) model is an internal model to precisely represent the state transition. The procedure is based on model-based testing techniques with test cases generated from UML state diagrams translated into LTS. Nirmal Kumar Gupta, Mukesh Kumar Rohil, "Improving GA based Automated Test Data Generation Technique For Object Oriented Software" [3]. This paper put forward a strategy for evaluating the fitness of both feasible and unfeasible test cases leading to the improvement of evolutionary search by achieving higher coverage and evolving more number of unfeasible test cases into feasible ones. The proposed strategy shows that genetic algorithms are useful in reducing the number of unfeasible test cases by generating test cases for object oriented software. Furthermore, we build our Genetic Algorithm for structural testing for generating more suitable test cases. In path testing weight reevaluation strategy is employed to develop unfeasible test cases into feasible test cases at the later generations. Esmaeil Mirzaeian, Samad Ghaderi Mojaveri, Homayun Motameni, Ahmad farahi, "An optimized approach to generate object oriented software test case by Colored Petri Nets"[11]. This paper put forward a technique for generating object oriented test cases using Colored Petri Nets extended version of Petri Nets usually used to system modeling and International Journal of Computer Applications (0975 – 8887) Volume 105 – No. 15, November 2014 32 simulation. They have introduced a new algorithm for to convert UML State chart into CPNs. This method considers net explosion problem and also generated net covers all instances of objects from different classes in the same hierarchy. At last a case study is also shown by a Banking account Example to show the benefits of the approach. Rajiv mall, "Automatic Test Case Generation From UML models" [12]. This paper proposed an algorithm, to generate test case from a combination of use case diagram and sequence diagram. First, they convert the use case diagram into use case graph and then sequence diagram into sequence graph. After that the two graphs are integrated and a system graph is generated. That system graph is traversed but not clearly mentioned. Mahesh Shirole, Rajeev Kumar, "A Hybrid Genetic Algorithm Based Test Case Generation Using Sequence Diagrams"[8]. This paper presented a hybrid approach of generating test cases using sequence diagram with genetic algorithm. Sequence diagram shows the method call dependencies that exist among the methods that potentially appear in a method call sequence, which is good for integration testing. Test cases generated using genetic algorithm improves the method coverage as well as exception coverage. Mahesh Shirole, Rajeev Kumar, "UML behavioral model based test case generation: a survey" [13] The objective of this paper is to improve the understanding of UML based testing techniques. It has focused on test case generation from the behavioral specification diagrams, namely sequence, state chart and activity diagrams. Also classified the various research approaches that are based on formal specifications, graph theoretic, heuristic testing, and direct UML specification processing and discussed the issues of test coverage associated with these approaches. Yamina Mohamed ben Ali, Fatma Benmaiza, "Generating Test Cases for Object-Oriented Software Using Genetic Algorithm and Mutation Testing Method."[14] This paper presented an automatic creation of software test cases based on the use of a genetic algorithm and a mutation testing technique. Philip Samuel, Rajib Mall, Pratush kant, "Automatic test case generation from UML communication diagrams"[15]. This paper presented a method to generate cluster level test cases based on UML communication diagrams. In this approach, A tree representation of communication diagrams is being constructed. Then a post-order traversal of the constructed tree for selecting conditional predicates from the communication diagram is being carried out. The generated test cases achieve message paths coverage as well as boundary coverage. The technique is being tested on several example problems. Ranjita Kumari, Vikas Panthi, Prafulla Kumar Behera, "Generation of test cases using Activity Diagram"[6] In this paper, test cases are generated using activity path coverage criteria. Here, a case study on Soft drink Vending Machine (SVM) has been presented to illustrate our approach. "Test Cases Generation from UML Activity Diagrams"[16] This paper proposed a method to generate test cases from UML activity diagrams that minimizes the number of test cases generated while deriving all practically useful test cases. Boghdady, P. , Badr, N. L., Hashem, M. A., Tolba, M. F., "An enhanced technique for generating hybrid coverage test cases using activity diagrams" [17]. This paper put forward an enhanced approach for automatically generating test cases from activity diagrams. Category partition method is applied to generate the final set of reduced test cases. The proposed model validates the generated test paths during the generation process to ensure that they meet a hybrid coverage criterion. The proposed model is automated and applied to around forty different case studies in different domains. Experimental evaluation is demonstrated to prove that the proposed model saves time and cost, thus increases the performance of the testing process. Yvan Labiche "Integration Testing Object-Oriented Software Systems: An Experiment-Driven Research

Approach”[18] has discussed about the questions : What integration testing process, indicating in which order classes are (Integration) tested, should be selected? Which test design techniques should be applied to unit and integration test classes when following an integration test order? G.Suganya and S.Neduncheliyan has given the idea about trouble markers of object-oriented software and object-oriented testing techniques and specialized techniques for OO Environment in “A Study of Object Oriented Testing Techniques: Survey and Challenges”[19]. They also discussed about how Unit Testing, Integration Testing and System Testing are being carried out in the Object Oriented environment. Nagendra Pratap Singh, Mrinal Kanti Debbarma has explained about the Life Cycle of Object-Oriented Testing in “The Review: Lifecycle of Object-Oriented Software Testing” [20]. This cycle provide us a big point of view to test object-oriented software. Although it is not much, differ from conventional testing but helpful for the thorough study of various approaches.

activity diagrams” [17]. This paper put forward an enhanced approach for automatically generating test cases from activity diagrams. Category partition method is applied to generate the final set of reduced test cases. The proposed model validates the generated test paths during the generation process to ensure that they meet a hybrid coverage criterion. The proposed model is automated and applied to around forty different case studies in different domains. Experimental evaluation is demonstrated to prove that the proposed model saves time and cost, thus increases the performance of the testing process. Yvan Labiche “Integration Testing Object-Oriented Software Systems: An Experiment-Driven Research Approach”[18] has discussed about the questions : What integration testing process, indicating in which order classes are (Integration) tested, should be selected? Which test design techniques should be applied to unit and integration test classes when following an integration test order? G.Suganya and S.Neduncheliyan has given the idea about trouble markers of object-oriented software and object-oriented testing techniques and specialized techniques for OO Environment in “A Study of Object Oriented Testing Techniques: Survey and Challenges”[19]. They also discussed about how Unit Testing, Integration Testing and System Testing are being carried out in the Object Oriented environment. Nagendra Pratap Singh, Mrinal Kanti Debbarma has explained about the Life Cycle of Object-Oriented Testing in “The Review: Lifecycle of Object-Oriented Software Testing” [20]. This cycle provide us a big point of view to test object-oriented software. Although it is not much, differ from conventional testing but helpful for the thorough study of various approaches.

IV. CONCLUSION

This literature paper Unified Modeling Language has now become a model in the field of software testing, particularly in the industry sectors. New technique for the generation of test case from these UML diagrams needs to be recognized. Furthermore, the study shows the different method mostly concentrates on the behavioural diagrams, and covers different techniques for software testing. In future, we are planning to develop an automated tool for generating test cases through class diagram and use different approach.

REFERENCES

[1] .https://en.wikipedia.org/wiki/Use_Case_Diagram

[2] Baikuntha Narayan Biswal, Pragyan Nanda, Durga Prasad Mohapatra “A Novel Approach for Scenario-Based Test Case Generation” International Conference On Information Technology © 2008 IEEE Computer Society

[3] Nirmal Kumar Gupta, Mukesh Kumar Rohil “Improving GA based Automated Test Data Generation Technique For Object Oriented Software” 2013 3rd IEEE International Advance Computing Conference (IACC) .

[4] Supaporn Kansomkeat, Phachayanee Thiket and Jeff Offutt, “Generating Test Cases from UML Activity Diagrams using the Condition-Classification Tree Method” 2nd International Conference on Software Technology and Engineering(ICSTE) @ 2010 IEEE.

[5] V.Mary Sumalatha, G.S.V.P.Raju, Object Oriented Test Case Generation Technique using Genetic lgorithms”International Journal of Computer Applications (0975 – 8887) Volume 61– No.20, January ,2013

[6] Ranjita Kumari,Vikas Panthi, Prafulla Kumar Behera,“Generation of test cases using Activity Diagram” International Journal of Computer Science and Informatics, ISSN (PRINT): 2231 –5292, Volume- 3, Issue-2, 2013

[7] Abinash Tripathy and Anirban Mitra, “Test Case Generation Using Activity Diagram and Sequence Diagram” Proceedings of ICAdC, AISC 174, pp. 121-129. springerlink.com © Springer India 2013

[8] Mahesh Shirole, Rajeev Kumar, “A Hybrid Genetic Algorithm Based Test Case Generation Using Sequence Diagrams” Contemporary Computing Communications in Computer and Information Science , Springer Verlag,Volume 94, 2010, pp 53-63

[9] Xiajiong Shen and Qian Wang PeipeiWang and Bo Zhou, "A Novel Technique Proposed for Testing of Object Oriented Software Systems" @2009 IEEE.

[10] Fanping Zeng, Zhide Chen, Qing Cao, Liangliang "Research on Method of Object-Oriented Test Cases Generation Based on UML and LTS" .The 1st International Conference on Information Science and Engineering (ICISE2009) ©2009 IEEE computer Society

[11] Esmaeil Mirzaeian, Samad Ghaderi Mojaveri, Homayun Motameni, Ahmad farahi, "An optimized approach to generate object oriented software test case by Colored Petri Nets", 2010 2nd International Conference on Software Technology and Engineering (ICSTE) @ IEEE

[12] Monalisa Sharma,Rajib Mall, " Automatic Test Case Generation From UML models" The 10th International Conference on Information Technology@2007IEEE.

[13] Mahesh Shirole, Rajeev Kumar, "UML behavioral model based test case generation: a survey" ACM(DL), July 2013

[14] Yamina Mohamed ben Ali, Fatma Benmaiza, "Generating Test Case for Object-Oriented Software Using Genetic Algorithm and Mutation Testing Method" ACM(DL),2012

[15] Sujata khatri, R.S.Chillar, "Generating Test Cases for Object Oriented Programs Using Specification based Testing Techniques" Sujata Khatri et al / Indian Journal of Computer Science and Engineering (IJCSE), Feb-Mar 2012

[16] Hyungchoul Kim, Sungwon Kang, Jongmoon Baik, Inyoung Ko, "Test Cases Generation from UML Activity Diagrams" © 2007 IEEE computer society

[17] Boghdady, P. , Badr, N. L., Hashem, M. A., Tolba, M. F., "An enhanced technique for generating hybrid coverage test cases using activity diagrams" Informatics and Systems (INFOS), 8th International Conference, 2012 in IEEE.

[18] Yvan Labiche "Integration testing object-oriented software systems: an Experiment-driven Research Approach", IEEE 24th Canadian Conference on Electrical and Computer Engineering,2011.

[19] G.Suganya and S.Neduncheliyan " A Study of Object Oriented Testing Techniques: Survey and Challenges", IEEE International Conference on Innovative Computing Technologies(ICICT)2010.

[20] Nagendra Pratap Singh and Mrinal Kanti Debbarma " The Review: Lifecycle of Object-Oriented Software Testing", IEEE 3rd International Conference on Electronic Computer Technology (ICECT), 2011 (Volume:3).