A Survey on Model Based Test Case Prioritization Using UML Sequence and Activity Diagram

[1]Lokesh Kumar Rathore, [2]Neelabh Sao
[1]M.Tech Scholor (Software Engineering), [2]Assistant Professor,
[1]Department Of Computer Science & Engineering,
[1]Rungta College of Engineering & Technology, Bhilai, India

***Abstract*— Software Testing and Maintenance is the crucial phase of software development life cycle (SDLC). More than half of the total development cost of a software system is associated with this phase. As the size and complexity of software get increases testing of software also becomes complex and takes more time and cost. Therefore it is required to detect faults as earlier as possible to maintain the software quality. Test cases or scenario prioritization is one of the effective techniques to find error earlier. In this paper, we proposed a combined approach to generate and prioritize test cases using UML Sequence and Activity diagrams. The objective of the project is effective use of UML model for test case prioritization with high fault detection rate and higher testing coverage criteria to reduce time and cost of software testing.** *(Abstract)*

***Index Terms*— Test Case Prioritization, Activity Diagram, Sequence Diagram, Software Testing.**

I. INTRODUCTION

Software Testing is the process of determining whether the system fulfill the specified requirements or not. Testing is exercising a system to find out errors, gaps, or requirements that are omitted in opposition of real requirements [1]. Modeling is the designing of software applications prior to coding and its necessary activity of large software projects.

The UML (Unified Modeling Language) of OMG (Object Management Group) is visualization, specification, and documentation of software systems models, along with their design and structure. UML models are an important information source for test case formation and it is independent of methodology [2]. UML can be used to express the results of design and analysis, regardless of the methodology used. The transformation of UML model from one tool into another tool or into a repository, or for refinement another OMG standard XMI (XML Metadata Interchange) can be used.

A. SOFTWARE TESTING TECHNIQUEs:

i. Black Box Testing:

Black Box testing, a software testing strategy evaluates the functionality of an application without worrying of internal structures or workings. The Black Box method of testing can be applied to almost every level of software testing i.e. unit, integration, system and acceptance. It is based on the analysis of the software specifications without reference to its internal activity. It ensures how well the software component fulfills the specified requirement for the system component. It is shown in fig.1.

The honesty of exterior information is maintained in black box testing, The example of black box testing techniques in which user is not get involved are functional testing, ad-hoc testing, stress testing, load testing, recovery testing exploratory testing, usability testing, smoke testing, and volume testing, and the black box testing methods such as user acceptance testing, alpha testing and beta testing in which user involvement is required. Some other types of Black box testing strategy like graph based testing method, equivalence partitioning, boundary value analysis, comparison testing, orthogonal array testing, specialized testing, fuzz testing, and traceability metrics.

Advantages of black box testing include:

- Since the tester and the designer are independent of one another hence test is unbiased.
- The examiners need not to be aware of any particular programming languages.
- The testing is performed as per the user perspective, not the designer.
- As soon as the specifications are completed test cases can be designed.

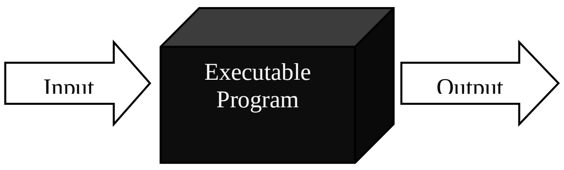

Fig.1 black box testing

ii. White Box Testing:

A White Box testing is a technique of evaluating the software that examines internal structures or workings of an application, rather than its functionality unlike the black box testing. It is also called as transparent box testing, clear box testing, glass box testing, and structural testing. To design test cases in white box testing an inner view of the system along with programming skills are used. White box testing based on an examination of internal activity and design of a component of software. Basically it is the process of providing the input to the system and testing how the system processes that input to produce the requisite output.

White Box Testing Techniques:

- Statement Coverage: Objective of this technique is to exercising all statements of program with minimal tests.
- Branch Coverage: This method ensures that every branch is evaluated at least once by executing a succession of tests.
- Path Coverage: In this method all possible paths i.e. each statement and branch is covered.

Advantages of White Box Testing:

- Developers need to reason carefully about implementation.
- Uncovers errors in hidden code.
- Fix the Dead Code or other issues.

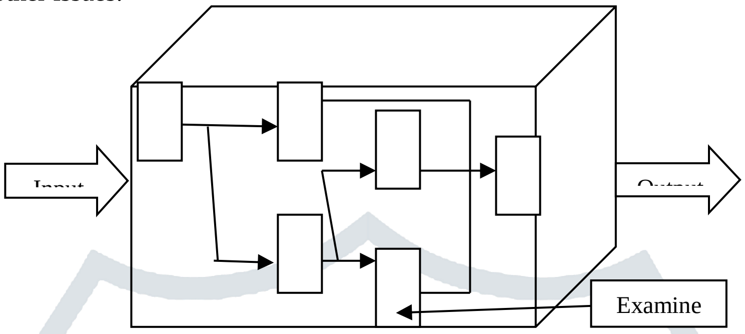

Fig.2 white box testing

iii. Gray Box Testing:

Gray box testing is a mixture of black box testing and white box testing. The objective is to find the error if any due to incorrect structure or improper usage of applications.

Advantages of gray box testing:

- Suitable for large code segments.
- For testing the application one need not to know programming language or methods.
- Roles of developers and users are clearly defined for testing. This testing is based users point of view, rather than the designer.
- Reverse engineering can be incorporated with Grey box to identify boundary values or error messages.

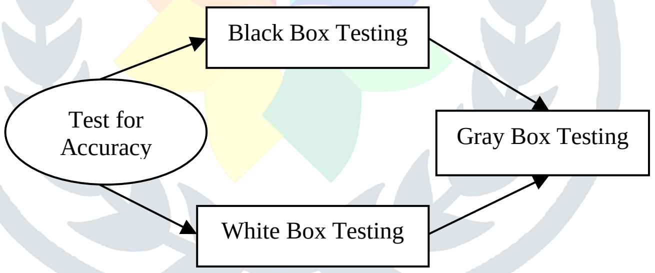

Fig.3 gray box testing

B. RELEVANT UML DIAGRAMs:

UML was made by Object Management Group and UML 1.0 detail draft was proposed to the OMG in January 1997. UML is a "dialect" for pointing out and not a system or method. The UML is utilized to characterize a software system; to detail the curios in the system, to report and develop it is the dialect that the blueprint is composed in. The UML is a widely used general-purpose modeling language in software engineering, which is intended for defining, envisioning, developing, and archiving the software system artifacts. Furthermore, utilizing XMI (XML Metadata Interchange, an alternate OMG standard), you can exchange your UML model from one device into a storehouse, or into an alternate tool for refinement or the following step in process of development.

What would you be able to Model with UML? UML 2.0 characterizes diverse sorts of diagrams, divided into three categories:

i. Structure Diagrams incorporate the Object Diagram, Class Diagram, Composite Structure Diagram, Component Diagram, Package Diagram, and Deployment Diagram.

ii. Behavior Diagrams incorporate the Use Case Diagram (utilized by a few procedures while gathering requirements); Activity Diagram, and State Machine Diagram.

iii. Interaction diagram all got from the more general Behavior Diagram, incorporate Communication Diagram, Sequence Diagram, Timing Diagram, and Interaction Overview Diagram.

Sequence diagram or chart is the most widely recognized sort of interaction diagram, which concentrates on the message exchange between quantities of life savers.

An UML sequence graph is an association outline that catches time subordinate (temporal) successions of collaborations down between items. They demonstrate the sequential grouping of the messages, their names and reactions and their possible contentions.

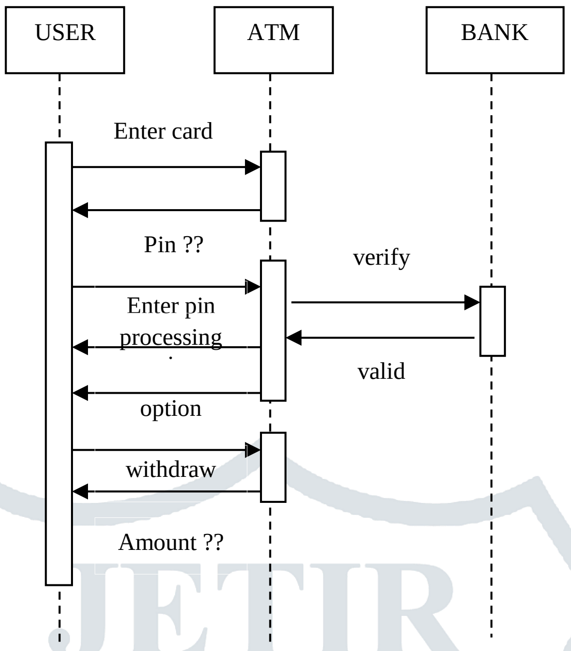

Fig.4 Sequence Diagram for ATM withdrawal

Activity diagram is an alternate essential chart in UML to define dynamic parts of the system. Activity chart is essentially a flow diagram to show the stream structure one action to an alternate movement. Action graphs depict the work process conduct of the framework. These are like state diagram on the grounds that exercises are the condition of doing something. The simplest approach to imagine an activity chart is to think about a flowchart of a code.

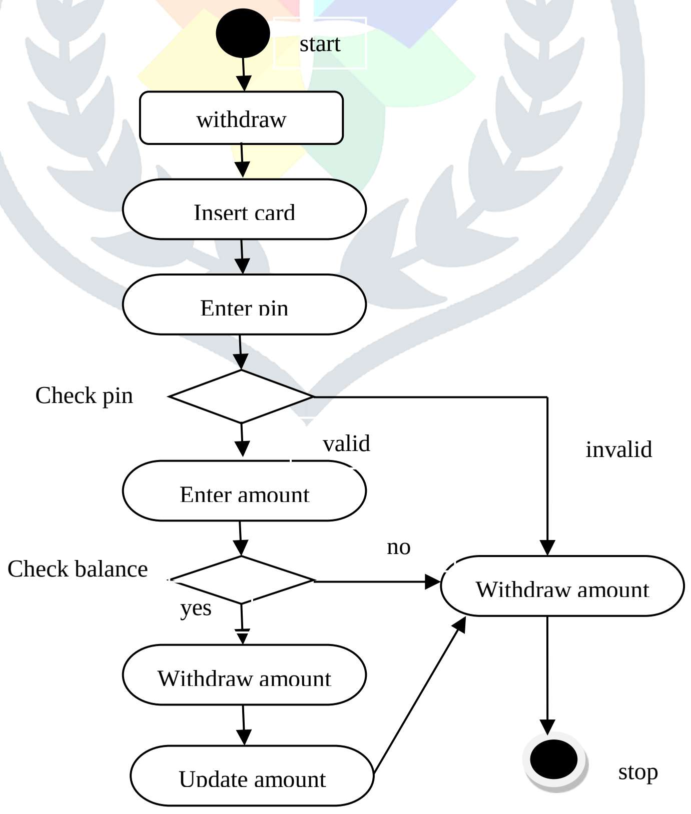

Fig.5 Activity Diagram for ATM withdrawal

C. MODEL BASED TEST CASE PRIORITIZATION:

In this test case prioritization technique model of a framework is utilized to schedule the test cases. Framework models are utilized to catch a few parts of the framework conduct. The prioritization of test cases with model may enhance the early error discovery when contrasted with the code based test case prioritization. Model based test prioritization may be a modest option to the current code based test prioritization systems. Be that as it may, model based test case prioritization may be touchy to the right/wrong data gave by the analyzers/developer. Hence forth demonstrate based prioritization of test case is the best one contrasted with code based prioritization.

II. RELATED WORK

In [1] Swain R. K. et al (2013) has exhibited an incorporated methodology and a prioritization procedure to produce test scenarios from UML activity diagrams and communication at cluster level. In this methodology, first they build a tree representation of communication diagrams, and activity diagrams and convert them into an intermediate tree. At that point complete a post order traversal of the built tree for selecting restrictive predicates from the intermediate tree. After this they use an algorithm to generate test scenarios and then prioritize those test scenarios. This strategy does not create redundant test scenarios and hence saves the time the time which can be utilized for testing the more basic components. This methodology is totally in light of the structural parts of the communication and activity diagrams. This method is in view of prioritization on the unpredictability of constructs in the activity diagrams and communication and it gives approx 60% average percentage fault detection rate.

In [3], Mohanty S., et al (2011) has proposed another prioritization system to organize the experiments to perform regression testing for Component Based Software System. The parts and the state changes for a segment based programming system are shown by UML state diagram which are then changed over into Component Interaction Graph (CIG) to define the interrelation among segments. The prioritization calculation takes this CIG as data along with the old test cases and produces an organized test suit considering aggregate number of state changes and aggregate number of database access.

In [4], Misra S. K and Mohapatra D. P, (2014) has presented a novel methodology for test scenario construction using UML sequence diagram focusing the pieces, nesting of fragments and control flow primitives display in sequence diagram. The strategy first creates a intermediate diagram called Sequence Control Flow Graph (SCFG) from the XMI representation of UML sequence graph. At that point they examine the control flow data, message flow and the fragment structure, it produces test scenarios, for different use case display in a system.

This approach is a completely methodical understanding of control flow data for different fragments. In this manner this methodology utilizes primitives of these control flow for test scenario creation in automated way. The test scenarios therefore created are suitable for useful testing an interaction and scenario faults.

In [5], Khandai M. and Acharya A. A., Mohapatra D. P., (2011) has introduced new way for test cases creation for parallel system with the assistance of UML Sequence Diagram. The Sequence Diagram is changed into a Concurrent Composite Graph (CCG) and then CCG is navigated by DFS (Depth-First-scan) and BFS (Breath-First-Technique) to produce test cases for concurrent systems. This mechanism keeps away from test blast by taking care of the deadlock and synchronization issues of concurrency. The test cases are valuable for recognizing situation, operational and communication faults in case of concurrent systems.

In [6], Gantait A. (2011), a way to producing test cases from UML activity diagrams and prioritizing those test cases utilizing model data furthermore proposes a methodology for selecting test data taking into account investigation of the branch conditions of the decision nodes in the activity diagrams. This methodology would help in arranging the prioritized testing sequence accordingly enhancing the testing adequacy of software applications modeled using UML notations. This prioritization methodology is taking into account probability of use of a path at runtime.

In [7], Mohanty H and Sapna P.G. (2009), has proposed a prioritization procedure taking into account UML action charts. This methodology is totally in light of the structural parts of the activity diagram The develops of an action graph are utilized to organize situations. Experiment prioritization includes systems intended to discover the best organized test suite.

The model based test case prioritization is developing system in test prioritization for testing the product in less time with upgraded shortcoming discovery rate.

From above writing survey it is observed that there are a few issues with respect to UML model that can be utilized for creating and prioritizing test cases to reduce fault detection time and expense of testing the product application.

III. PROPOSED METHOD

The proposed work is about to provide the effective test case prioritization mechanism with minimum time and cost of testing the software application. We proposed a combined approach to generate and prioritize test cases using UML Sequence and Activity diagrams. The objective of the project is effective use of UML model for test case prioritization with high fault detection rate and higher testing coverage criteria to reduce time and cost of software testing. This is shown in fig 6.

Major steps of our approach are:

- Parsing the XMI representation of UML Sequence Diagram and Activity Diagram.
- Creating the Sequence Control Flow Graph (SCFG).
- Generating Test Cases.

- Prioritizing Test Cases.
- Performance Evaluation using metric like Average Percentage of Fault Detected (APFD).

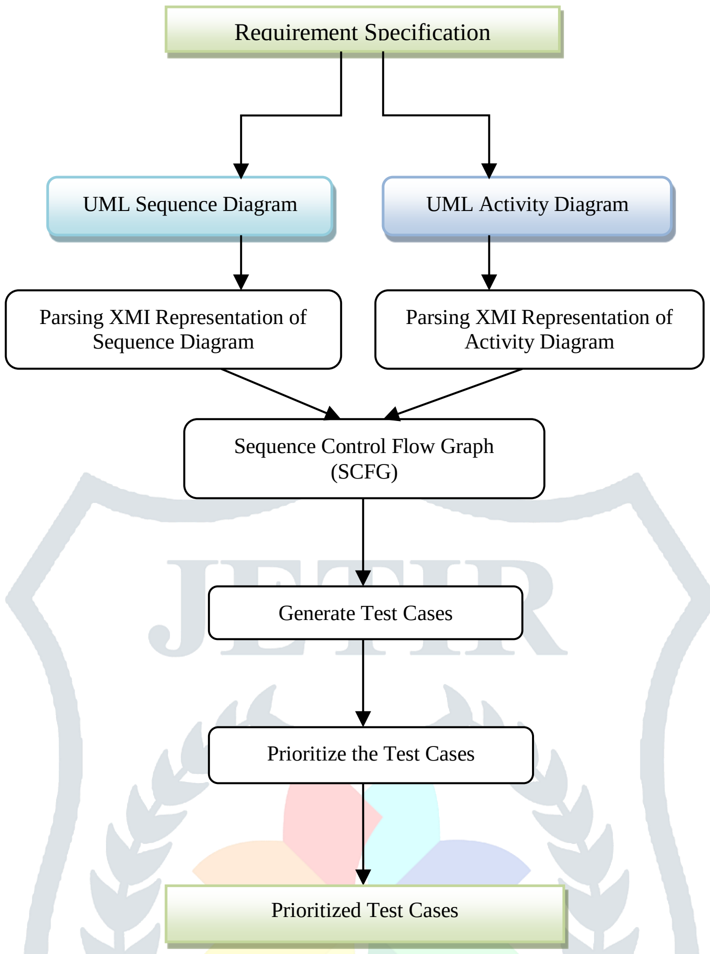
Requirement Specification
UML Sequence Diagram
UML Activity Diagram
Parsing XMI Representation of Sequence Diagram
Parsing XMI Representation of Activity Diagram
Sequence Control Flow Graph (SCFG)
Generate Test Cases
Prioritize the Test Cases
Prioritized Test Cases

Fig.6 proposed methodology

IV. CONCLUSION

The model based test case prioritization is the emerging technique in test case prioritization for testing the software in less time with optimized fault detection rate.

In our investigation of leaving testing strategies exploration is centered on strategies in light of codes, models and UML diagrams. For deciding the adequacy of prioritization strategies measurement (APFD) is utilized. There is great scope regarding research in understanding the ideas of distinctive code based strategies and conduct of segments, associations and similarity of parts.

From above literature review it is found that there are several research is done on model based test case prioritization in recent year but none of them used this two UML model together. The proposed work is about to provide the effective prioritized test cases with minimum time and cost of testing the software application.

REFERENCES

[1]. Ranjita Kumari Swain, Vikas Panthi, Durga Prasad Mohapatra, Prafulla Kumar Behera, “Prioritizing test scenarios from UML communication And activity diagrams”, Innovations Syst Softw Eng, Springer-Verlag London 2013.

[2]. http://www.uml.org,http://www.wikipedia.org, http://tutorialspoint.com

[3]. Sanjukta Mohanty, Arup Abhinna Acharya, Durga Prasad Mohapatra, “A Model Based Prioritization Technique For Component Based Software Retesting Using Uml State Chart Diagram”, IEEE Third Int’l Conf. on Electronics Computer Technology, 2011.

[4]. Saroj Kanta Misra "Test Scenario Generation and Prioritization from UML 2.x Sequence Diagrams".

[5]. Monalisha Khandai, Arup Abhinna Acharya, Durga Prasad Mohapatra, “A Novel Approach of Test Case Generation for Concurrent Systems Using UML Sequence Diagram”, 2011 IEEE.

[6]. Amitranjan Gantait, “Test case Generation and Prioritization from UML Models”, IEEE Second International Conference on Emerging Applications of Information Technology, 2011.

[7]. Sapna P.G., Hrushikesha Mohanty, “Prioritization of Scenarios based on UML Activity Diagrams,” IEEE First International Conference on Computational Intelligence, Communication Systems and Networks, 2009.

[8]. Sangeeta Sabharwal, Ritu Sibal, Chayanika Sharma, “Prioritization Of Test Case Scenarios Derived From Activity Diagram Using Genetic Algorithm”, IEEE Int’l conf. on Computer & Communication Technology, 2010.

[9]. Joao Pascoal Faria, Ana C. R. Paiva, Zhuanli Yang,” Test Generation from UML Sequence Diagrams,” IEEE Eighth International Conference on the Quality of Information and Communications Technology, 2012.

[10]. Beilei Liang ,Pan Liu, Huaikou Miao, “Scenario Specification based Testing Model Generation,” 2013 IEEE.

[11]. Prateeva Mahali, Arup Abhinna Acharya, “Model Based Test Case Prioritization Using Uml Activity Diagram And Evolutionary Algorithm” International Journal of Computer Science and Informatics, ISSN, 2013.

[12]. Arup Abhinna Acharya, Durga Prasad Mohapatra, Namita Panda, “Model Based Test Case Prioritization for Testing Component Dependency in CBSD Using UML SequenceDiagram”.(IJACSA) International Journal of Advanced Computer Science and Applications, December 2010.

[13]. S. Shanmuga Priya, Dr. P. D. Sheba “Test Case Generation From Uml Models – A Survey”, International Conference on Information Systems and Computing (ICISC-2013).

[14]. Chhabi Rani Panigrahi, Rajib Mall, “Model-Based Regression Test Case Prioritization” ACM SIGSOFT Software Engineering, November 2010.

[15]. Deepak Garg, Amitava Datta and Tim French, “A Two-Level Prioritization Approach for Regression Testing of Web Applications” 2012 19th Asia-Pacific Software Engineering Conference, IEEE.

[16]. Deepak Garg, Amitava Datta "Test Case Prioritization due to Database Changes in Web Applications" Fifth International Conference on Software Testing, Verification and Validation, IEEE 2012.

[17]. Santosh Kumar Swain, Durga Prasad Mohapatra,and Rajib Mall, "Test Case Generation Based on Use case and Sequence Diagram" Int.J. of Software Engineering, IJSE Vol.3 No.2 July 2010.

[18]. Ranjita Kumari Swain, Prafulla Kumar Behera and Durga Prasad Mohapatra “Minimal TestCase Generation for Object-Oriented Software with State Charts”.

[19]. Cagatay Catal, Deepti Mishra,”Test case prioritization: a systematic mapping study” Springer Science+Business Media, LLC 26 july 2012.