INTERNATIONAL JOURNAL OF TECHNOLOGY AND COMPUTING (IJTC)
ISSN-2455-099X,
Volume 2, Issue 5 May 2016.

Functional Test Cases Generation Based on Automated Generated Use Case Diagram using DFS algorithm

Ramandeep Kaur [a], Rasbir Singh [b]

[a] *Research Scholar, M.Tech CSE*
[a] *Assistant Professor, CSE Department*
[a,b] *Department of CSE,RIMT-IET, Mandigobindgarh , Punjab, India*

ABSTRACT
The Use case situations are made amid the examination stage to determine programming framework's necessities and can likewise be utilized for making framework level experiments. Utilizing use cases to get framework tests has a few advantages including test plan at early phases of programming improvement life cycle that decreases over all advancement expense of the framework. Current methodologies for framework testing utilizing use cases include useful subtle elements and does exclude watches as passing criteria i.e. utilization of class chart that appear to be troublesome at exceptionally starting level which lead the need of particular based testing without including utilitarian subtle elements. A procedure for framework testing specifically got from the determination without including practical points of interest has also been developed. This framework utilizes both types of conditions namely pre and post connected as watchmen at every level of the utilization cases that empowers era of formalized experiments and makes it conceivable to produce test cases for every stream of the framework. We utilized use case situations to create framework level experiments, though framework arrangement outline is being utilized to cross over any barrier between the test target and experiments, got from the detail of the framework. Since, a state graph got from the mix of succession charts can display the whole conduct of the framework. Created test cases can be utilized and executed to state diagram to catch conduct of the framework with the state change. Every one of these strides empowers deliberate refining of the detail to accomplish the objectives of framework testing at initial advancement stage**.** This paper presents a review of the use of use case diagrams in software testing.
Keywords: Software testing, UML, Use case diagrams, SDLC, DFS algorithm

I. INTRODUCTION

In the field of software engineering, the most well-known meaning of an experiment is an arrangement of conditions or variables under which an analyzer will figure out whether a prerequisite or use case upon an application is halfway or completely fulfilled. Making test cases includes an additional level of accuracy to an utilization case. Test cases are ordinarily composed in light of project source code. This puts forth test defense era troublesome particularly to test at group levels. Further this methodology turns out to be deficient in segment based programming advancement, where the source code may not be accessible to the engineers. It is in this manner alluring to produce test cases consequently from the product plan reports, instead of code or code-based particulars. Test era from outline reports has the additional point of preference of permitting experiments to be accessible ahead of schedule in the product improvement cycle, in this manner making test arranging more powerful. Further, in outline based tests the created test information is free of a specific usage of the configuration. The advantage of making an experiment is that it helps with distinguishing the vital ideas being used case.

Software testing is a procedure of executing a project or application for finding the errors or defects in it or it can be said as for locating the bugs in the executed program. Testing of software also ensures that software is valid and verified and satisfies the requirements that are required to fulfill product demand. The software developed should be appropriate enough to meet the demands be it technical or business for which it is designed and it should be able to work as per the user expectations. The testing of software occurs in the form a process which is generally mistook by individual activity. And this process takes place for the whole SDLC. SDLC is the development life cycle that involves all the activities that occur throughout the complete cycle of software.

II. RELATED WORK

Arjinder Singh et al. 2015, In this paper the author has highlighted the benefits of software testing. Software testing is a technique that checks the system for errors and defects. A new developed system or software will have some limitations and may have some defects too, so these are found by software testing. Software development is complete after undergoing three stages namely coding, design and its specification. Testing is done to check whether the software works properly and to ensure that it is error free and do not have any defects. The author proposes a framework in the paper that will automatically generate a use case diagram. It is automatic software testing system that works in certain steps for testing the developed software but is an efficient method and saves time and effort that is otherwise utilized for testing software.

U.Senthil Kumaran, et al. 2011, In this paper the author has proposed a new scheme to generate test cases. The quality of test cases can be measured by the accuracy of the updations, the problems in the updates should be least and it should be complete so that issues that come while updating change requests can be avoided. The author suggests generation of some test cases at initial stages and these cases will have knowledge of almost all required cases. Use cases will capture the requirements so these will be used for generation of test cases in the proposed work.

Cle´mentine Nebut et al. 2006, The authors in this paper has proposed a use case for software testing. The author suggests bridging gap between two cases namely the high-level use case and the second one is concrete test case and this bridging can help in automating the test generation. The author in the paper used software of embedded that will be object-oriented for designing automatic system and while designing this system the problems of the two cases will be considered so that the gap can be bridged. The test cases in the proposed technique are synthesized using automatic transition system. The objective of the author in the paper is to produce desired results with the generated tests.

Ashish Verma et al. 2014, In this paper the author has utilized the UML diagrams for application based on object orientation for designing the new approach. The author considers that the utilization of object oriented application is at initial stage and this method has not been used to large extent so the author uses it for developing new technique. Test cases are generated from UML diagrams in the paper and these are then used for discovering pattern and retrieving information.

N. K. Sharma et al. 2013, The author has clearly described the importance of software testing in SDLC in the paper. The author states that testing shows the difference between the obtained results and the expected results. The programs that are executed for finding the difference are run on test cases. The efficiency of the software testing system is improved by increasing the reliability and decreasing the time consumption and cost of the system. The author mentions test case generation as one of the best ways to test software. In this paper, the author has talked about various approaches using UML diagrams for automated software testing.

Anjali Sharma et al. 2013, In this paper the author stated that importance of test case generation for testing any software automatically. It is unknown fact that errors could not be detected from written test cases and this result in the failure of software. In the paper, the author has proposed generation of test case on the basis of design; it is different from conventional approaches a earlier the code was considered while testing. In the proposed work UML modeling is employed for generating the test cases and these diagrams will also help in compute cost of each test case in server hitting. The information about the entity is obtained from the use case diagram. The static information is got from the sequence diagram and the time information about messages is taken from the class diagrams. This paper proposes an optimized technique for generating test cases and detecting errors in the software.

Monika et al. 2014, In this paper the author has reviewed the techniques used for testing. It is said that software is approved only after certifying its quality by testing. Software is tested each time a change is made in it and it is done to validate the change and check the working of the software.

III. UML (UNIFIED MARKUP LANGUAGE)

UML known as Unified Markup Language. This language is used to model the structure and the behavior of the system. How the user and the system interact is defined with the help of this language. This language was developed with the purpose of helping IT professionals to work together with model computer applications. UML diagrams are basically defined in two categories:

1. Structural Diagrams
2. Behavioral Diagrams

Structural diagrams:

Structural diagrams are those types of UML diagrams that tell about the system structure and also depict the relation between the levels that are abstracted and implemented in the system. The concepts that are included in the structural diagrams are meaningful and these are those concepts that are abstracted and implemented in the system. The structural diagrams are further of many types that are described below:

i. Class Diagram: These are those diagrams that are basic building block for any object oriented solution. The main things that are included in the class diagrams are the classes, their attributes and their operations.

ii. Component Diagram: These types of diagrams are used in the complex system where the whole system is divided into components. Interfaces are present in these diagrams that help components to communicate with each other.

iii. Deployment Diagram: These are those diagrams that help depict the hardware of the system. These diagrams find their applications in the areas where software is deployed across various machine.

iv. Object Diagram: It is clear that objects are instances of a class. So, object diagrams can be called as instance diagrams of a class. The basic function of these object diagrams is to show relationship between the objects present in a system.

v. Package Diagram: These are type of structural diagrams that shows the dependencies between different packages in the system.

vi. Profile Diagram: These are the latest types of diagrams in UML. And these types of structural diagrams describe the lightweight extension mechanism by defining stereotypes, tagged values and constraints in a system.

vii. Composite Diagrams: These types of diagrams are used to show the internal structure of the class. Consider the example of a real class, the teachers and the number of students present in class will represent the internal structure of a class and the number of classes will be the part of internal structure of a school. So, these all things are depicted in these diagrams.

Behavioral Diagrams:

These types of UML diagrams show the internal behavior of the system. The changes that are observed in the system in particular time are shown with the help of these diagrams.

i. Use case Diagrams: these diagrams uses actors that interacts with the system. It is the graphic representation of the actors involved in the system. It is believed that different actors in the system perform different functions, and this concept is utilized in the use case diagrams. At first, we must define the main actors and the main processes of the system. Actors are defined as the users and the function performed defined in the boundary. The example of use case diagram can be taken for ATM machine system.

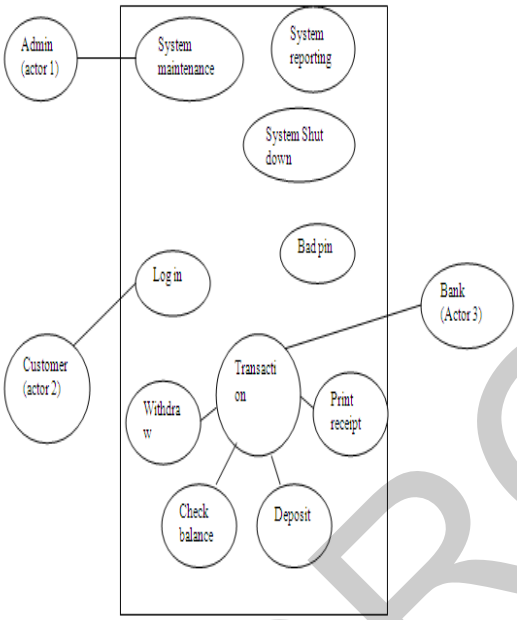


Fig. 1. Use case Diagram for ATM machine system

ii. Activity Diagram: Activity Diagram is used to represent the flow of system or component. It is an alternative to the State Machine Diagram.

iii. State Machine Diagram: State machine Diagrams are similar to Activity Diagram. But these diagrams consist of state diagrams or state chart diagrams as well. The state of the objects is represented in these diagrams. It is shown whether the diagrams are in the running state or waiting state etc. Two important terms are used in these diagrams that are state and event. The State shows the current position of an object. Event shows the change of the state. These diagrams visualize the reaction of system by internal/ external factors.

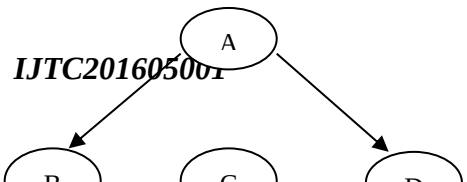
iv. Sequence Diagrams: These diagrams show the sequence of the processes. It shows how the objects interact with each other and the order in which those interactions occur. The processes are represented vertically and the interactions are shown as arrows in these diagrams.

v. Communication Diagram: These diagrams are also known as collaboration diagrams in UML. These are somewhat similar to sequence diagrams but in these the main focus is on the messages passed between objects.

vi. Interaction Diagrams: Interaction Diagrams are used to describe some type of interactions among different elements in the model. The basic objective of these diagrams is to capture the dynamic nature of the objects.

IV. DFS ALGORITHM

DFS stands for Depth First Search. In this each and every node or vertex of the graph or tree is traversed at once. The training starts from the root element and from that it move towards the adjacent vertex of the root node. If the adjacent vertex further have any adjacent vertex to it, the element then switches to it. This process will continue until the leaf vertex is traversed and then backtracking is performed to react towards another adjacent of root node. Stack is used in DFS for performing operations of push and pop.



A B E H F C G D

In this technique initially all the vertex are in 'Not Visited State'. But when the node is pushed into the stack the state changes to 'Waiting'. And when the adjacent of the node is found and the node is poped up from the stock, their state changes to the 'Processed'.
The algorithm works in some steps that are described below:

Step I. All the vertexes of 'G' are initially in not visited state.
Step II. Push the Starting vertex A into the stack and change its state to the 'Waiting State'.
Step III. Repeat step IV and VI while stack is not empty.
Step IV. Store the vertex on the top of the stack in a variable W.
Step V. Pop 'W' from stack and change its state to 'Visited'.
Step VI. For each neighbor N of W
If (vertex N is 'Not Visited') then
Insert Vertex N onto the stock & change the State to 'Waiting'.
End of if.
End of loop
End of Step III.
Step VII. Return

V. Conclusion & Future Scope:

Use case diagrams have been used in conventional techniques for generation of test cases and testing software. The test cases are needed to be generated automatically so that the time and effort could be minimized. To increase the efficiency of automatic generation of test cases, DFS algorithm can be applied. This efficient algorithm can efficiently generate use cases for AOP systems. Earlier the works are done on the basis of OOP system whereas in future the work can be done for AOP systems using use case diagrams for automated generation of test cases.

References

[1] Arjinder Singh, "Functional Test Cases Generation Based on Automated Generated Use Case Diagram", IJIRAE, Issue 8, Volume 2 (August 2015), pp 105-110
[2] U.Senthil Kumaran, "An Approach to Automatic Generation of Test Cases Based on Use Cases in the Requirements Phase" International Journal on Computer Science and Engineering, Vol. 3 No. 1 Jan 2011,pp 102-113
[3] Clementine Nebut, "Automatic Test Generation: A Use Case Driven Approach", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 32, NO. 3, MARCH 2006,pp 140- 155

[4] Ashish Verma, "Automated Test case generation using UML diagrams based on behavior" , International Journal of Innovations in Engineering and Technology ,Vol. 4 Issue 1 June 2014, pp 31-39
[5] N. K. Sharma, "Study Of Approaches For Generating Automated Test Cases By UML Diagrams", International Journal of Engineering Research & Technology, Vol.2 - Issue 6, June 2013 .
[6] Anjali Sharma, "Generation Of Automated Test Cases Using UML Modeling", International Journal of Engineering Research & Technology, Vol.2 - Issue 4, April – 2013.
[7] Monika, "Test Case Prioritization: A Review**",** International Journal of Engineering Research & Technology, Vol. 3 - Issue 5, May – 2014
[8]Muhammad Touseef, "A USE CASE DRIVEN APPROACH FOR SYSTEM LEVEL TESTING", pp 1-9