Use of UML Activity Diagram for Generation of Test Cases

1Satinder Kaur, 2Prof. Rasbir Singh
1M.Tech Student, 2Assistant Professor
1,2 Computer Science Department
RIMT-IET, Mandi Gobindgarh, PTU
Satinderkaur988@gmail.com
Rasbir.rai@gmail.com

***Abstract*- Object-Oriented program have changed the scenario of software development industry in terms of software development and its supporting technology. UML, which supports object-oriented technology, is widely used to describe the analysis and design specifications of software development. UML models are an important source of information for test case design. So, new testing techniques and practices including international standards for specifying how systems should work - such as Unified Modeling Language (UML) - have been established. Therefore, a new technique is presented for generating test cases for object oriented system using UML models. This technique is helpful in generating more test cases which will cover maximum aspects of the system. With the help of test cases chances of finding a bug is more. Means, generation of test cases is the most important issue in the software testing. Thus test cases need to be carefully designed. Test cases can be generated manually or automatically. Our focus is to generate test cases automatically by using UML Activity diagram. Activity diagrams are even more challenging for creating test cases. The test cases thus generated are suitable for dynamic testing of system. The proposed approach is to automate the generation of test scenarios from activity diagram using DFS and BFS method. By using these two methods, the useless test paths are eliminated which in turn reduces the time complexity. This approach also helps in synchronization between various activities. This approach was found to be effective in reducing the time complexity also.**

***Keywords*- Software testing, Activity diagram, UML models, UML language and test cases.**

I. INTRODUCTION

Software Testing is the process of executing a program with the intention of finding errors [1]. Day by day with increasing functionality of software has caused increasing the complexity and size of software applications. Due to this reason more emphasis has been sited on object oriented design strategy to cut down software cost and boost software reusability [2]. Every software code has been reviewed and verified through SQA activities but these activities are not sufficient. Every time the software delivered to the client has been thoroughly tested by client before sending it to the production. Thus developer has to test the software before it gets to the client. Testing has been generally performed by three ways: white-box testing, black box testing and gray-box testing [3-4]. The testing techniques discussed so far is good to test software what it was supposed to do but it will not test what is missing in software code. From last few years there has been slow development made to the testing of object-oriented systems. One innovative approach is to use UML models for test cases generation. This approach can lead to find out faults in the design phase prior to the development phase and thus causes correction of faults in early life cycle stages of software. This type of testing comes under the category of gray-box testing [5-9]. UML models are nothing but diagrammatical representation of specification document. UML models can be used as a base to derive test cases and to develop testing environments. However, using UML models to derive test case is not an easy task [10-11]. It is cumbersome for generating test models like control flow graph for an object oriented system. The UML Activity diagram is used for modeling discrete behavior of an object [12]. An activity diagram consists of flowchart for various activities with transition among them. It states the internal behavior of an operation of the system. An activity diagram is used for modeling the dynamic aspects of the system. Activity diagram is a kind of procedural flow chart. It helps in visualizing the sequence of activities involved in a control flow. An activity diagram emphasize on the sequential or concurrent flow path from activity to activity. It discusses the ordering of the activities [13].

II. RELATED WORK

Lot of work has been done on generating test cases from UML diagram. John D. McGregor, David A. Sykes presented a work to generate test cases on the basis of class diagram [14]. UML diagrams are broadly classified into two:

- Structure diagrams
- Behavior diagrams

The structure diagrams mainly focus on the static structural design of the system, so that they are used exclusively in documenting the software architectural design. Structure diagrams contain diagrams like class, component, deployment etc.

The Behavior diagrams are the diagrams which are having the dynamic structure, so that they are dynamic in nature and also it consists of diagrams like sequence, use case, activity etc.
UML has another set of diagrams, interaction diagrams which is an interaction involved. It consists of diagrams like sequence, communication, interaction overview and timing [15].Since UML diagrams are always more abstract and provides ease to generate test cases than control flow graphs so researchers have started using UML diagrams to generate

automated test cases from UML diagrams. Philip Samuel, Rajib Mall and Sandeep Sahoo have presented a novel testing methodology to test object-oriented software based on UML diagrams [16].It is also true that sometimes testing is mistakenly considered not a resource-intensive activity, which does not require any supportive tools. One survey has done on automated test case generation techniques categorized under specification based test case generation and model base test case generation. They also touched on few other approaches for generating test cases as path oriented test case generation and intelligent techniques [17].During the last 10-15 years many studies have been conducted and tools developed, where elements of above mentioned methods are implemented. Thus dynamic methods with elements of formal methods were distinguished: model driven testing, runtime verification and passive testing [18]. The time reviews' effectiveness is highly dependent on experience and motivation, as well as process organization and interaction between participants [19].Again a method is introduced by Korel by using function minimization method in the context of unit testing of procedural programs. He generated test data based on actual execution of the program under test using the function minimization method and dynamic data flow analysis. Test data are developed for the program using actual values of input variables. If during a program execution an undesirable execution flow is observed (e.g. the „actual" path does not correspond to the selected control path), then the function minimization search algorithm is used to automatically locate the values of input variables for which the selected path is traversed. In addition, dynamic data flow analysis is used to determine those input variables responsible for undesirable program behavior, leading to significant speedup of the search process [15]. But this paper shows how test cases are generated from Activity diagram to represent the behavior of the system. If all test cases consider all scenario of the situation then only system can give better results.

III. Implementation of proposed work

The proposed work is to generate test cases from UML activity diagram using BFS and DFS method which will help in eliminating useless test cases and generate the useful test paths with reducing the complexity of time. This approach also helps in synchronization between activities. This approach is implemented on JDK 7 version and used the Net beans framework of 6.7.1version. This approach will consider pre conditions, post conditions, input, output and tesecase ID's which will include all aspects of the system to improve the efficiency and effectiveness of the system.

First step will be implemented with the help of Unified Modeling Language (UML).

Second step will be generated with the help of some pre defined mapping rules. These pre defined mapping rules will help in converting UML activity diagram into an Activity graph.

After that, the Activity graph will help in generating the test paths or test cases. With the help of this java code it is easy to generate test cases of the system.

A Functional Diagram of proposed work

The Flow Chart shown by Fig 1 describes the proper working and easy understandable of the proposed approach. Means, how UML diagrams are helpful in generating test cases which will cover maximum cases of failures. It also shows that how the results are obtained from UML Activity diagrams.

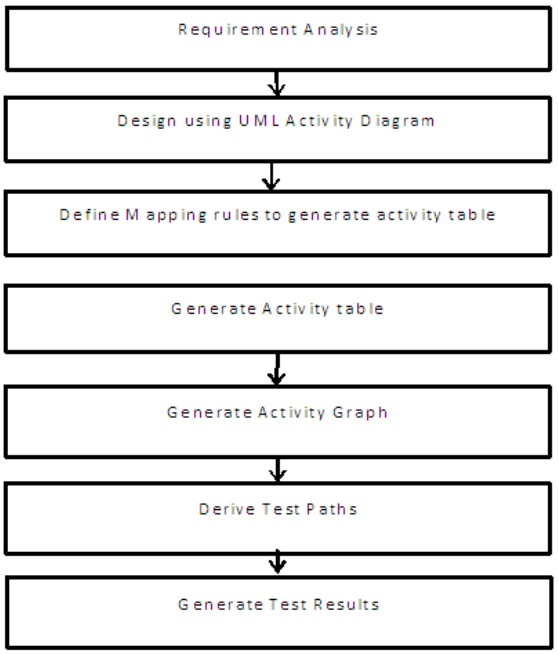

Fig.1 Functional Diagram of each task

B Activity Diagram for online shopping

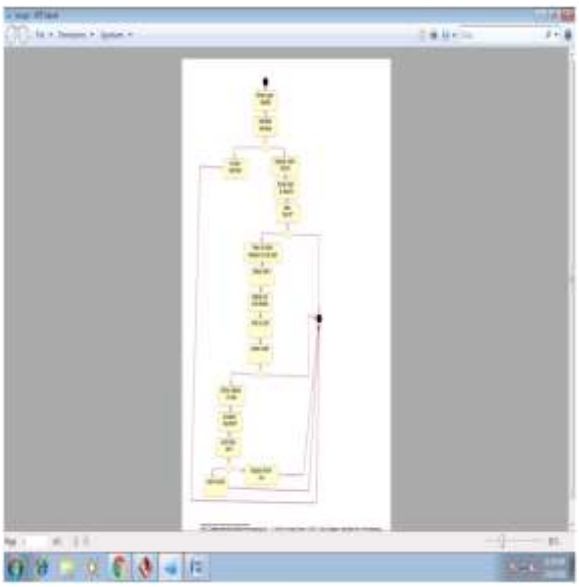

Fig.2 Activity diagram

C Converting Activity diagram to Activity graph

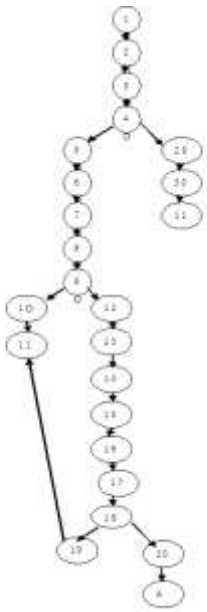

Fig.4 Activity graph

D Algorithm for test case generation

- Begin
- Traverse the graph using BFS and store them in a queue v ith size n
- Set number of paths, p=0 and number of visits=1
- Initialize all the nodes to ready state and set status =1(ready state)
- For i=1 to i=n
- Calculate indegree and outdegree.
- End of for
- Traverse the graph using DFS and store them into the stack
- Set number of visits for the node
- Set the status of the nodes that are stored in stack is 2 (waiting state)
- For visit = min and outdegree=0
- Backtrack to the node of number of visits=max
- Store these nodes into a stack again to calculate its path
- Set path=path + 1
- Set the status of the nodes that are stored in stack is 3 (Processed state)
- Enter current contents of stack into an array and traverse an array to define the paths of activity graph
- Exit.

D Valid Paths

Table 1: Valid Paths

Path Id's	Valid Paths
P1	1->2->3->4->5->6->7->8->9->10->11
P2	1->2->3->4->5->6->7->8->9->12->13->14->15->16->17->18->11
P3	1->2->3->4->5->6->7->8->9->12->13->14->15->16->17->19->20->21->22->23->24->25->11
P4	1->2->3->4->5->6->7->8->9->12->13->14->15->16->17->19->20->21->22->23->26->27->11
P5	1->2->3->4->28->29->11

E Generated Test cases

Table 2: Test cases

Test case ID	Pre-condition	Input	Output	Post-condition
TCID 1	Displaying home page	Enter login details and Enter item you want to search	Display item record and exit (Item not found).	Displays home page
TCID 2	Displaying home page	Enter login details. Enter item you want to search then Place order.	Display all the data related to that item and details of cart (where selected items are stored) and exit (don't want to place order right now).	Displays home page
TCID 3	Displaying home page	Enter login details. Enter item you want to search then Place order and enter required info for	Display invalid card and exit.	Displays home page

IV. CONCLUSION

As the demand of the new software increase in the current field of software engineering, new techniques and methods are needed for fulfilling the needs of the market. In order to make this possible in the case of design model, an approach for using the UML Activity diagram is used. The proposed approach can generate efficient test cases with lesser effort using an UML activity diagram. This helps in saving time and increases the quality of generated test cases. The overall testing process performance can be improved using this

approach. In this activity graph is generated from the Activity diagram by using some mapping rules . Then the proposed technique uses only activity graph as input. It generates the number of possible outcomes as an output which will help in generating the test cases. This algorithm is used for traversing Activity graph in order to extract all the possible test paths. The proposed algorithm is based on DFS and BFS method. It helps in reducing time, effort, and cost consumption.

References

[1] Myers, Glenford J. The art of software testing / Glenford J. Myers ; Revised and updated by Tom Badgett and Todd Thomas, with Corey Sandler.—2nd ed.p.cm. ISBN 0-471-46912-2 pp 6 Vol. 2, No. 2, pp. 79–93/doi: 10.1049/iet-sen: 20060061.

[2] H.-G. Gross. Measuring Evolutionary Testability of Real-Time Software.PhD thesis, University of Glamorgan, Pontypridd, Wales, UK, June2000.

[3] Wang Linzhang, Yuan Jiesong, Yu Xiaofeng, Hu Jun, Li Xuandong and ZhengGuoliang "Generating Test Cases from UML Activity Diagram based on Gray-Box Method" Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC'04) pp 284-291

[4] R. S. Pressman. "Software Engineering: A Practitioner's Approach", 6rd Edition, McGraw Hill, New York, 2005, pp. 424, 434, 449.

[5] A. Abdurazik and J. Offutt. "Using UML collaboration diagrams for static checking and test Generation".In International Conference on the Unified Modeling Language (UML 2000), York, UK, October 2000, pp 383- 395

[6] J. Hartmann, C. Imoberdorf, and M. Meisinger."UML-based integration testing". In International Symposium on Software Testing and Analysis (ISSTA 2000), Portland, USA, August 2000, pp 60 -70

[7] R. Heckel and M. Lohmann."Towards model-driven testing". Electronic Notes In Theoretical Computer Science, 82(6), 2003, pp 33-43

[8] Y.G. Kim, H.S. Hong, D.H. Bae, and S.D. Cha. "Test cases generation from UML state Diagrams". IEEEProceedings Software, 146(4), 1999, pp 187-192

[9] J. Offutt and A. Abdurazik."Generating tests from UML specifications". In International Conference on the Unified Modeling Language (UML 1999), Fort Collins, USA, October 1999, pp 416-429

[10] MonalisaSarmaDebasishKunduRajib Mall, "Automatic Test Case Generation from UML Sequence Diagrams", 15th International Conference on Advanced Computin and Communications. pp 60-64

[11]SupapornKansomkeat and WanchaiRivepiboon "Automated-Generating Test Case Using UML Statechart Diagrams" Proceedings of SAICSIT 2003.

[12] Blanco, R., Fanjul, J.G., Tuya, J.: Test case generation for transition-pair coverage using Scatter Searc. International Journal of Software Engineering and Its Applications 4(4) (October 2010)

[13] A. Abdurazik and J. Offutt, "Generating test cases from UML specifications," *In Proceedings of the Second International Conference Fort Collins, George Mason University*, USA, 1723, 416-429, 1999.

[14] John D. McGregor, David A. Sykes "A Practical Guide to Testing Object-Oriented Software", Addison Wesley, March 05, 2001, pp. 167

[15]http://www.uml-diagrams.org/sequence-diagrams.html, 17 November 2013.

[16] Philip Samuel, Rajib Mall and SandeepSahoo, "UML Sequence diagram Based Testing Using Slicing", IEEE Indicon 2005 Conference, Chennai, India, 11-13 Dec. 2005, pp 176-178

[17] J. Dick and A. Faivre, "Automating the generation and sequencing of test cases from model-based specifications," in *Proceedings of the First International Symposium of For- mal Methods Europe on Industrial-Strength Formal Methods*, 1993, pp. 268 - 284.

[18] S. Helke, T. Neustupny, and T. Santen, "Automating test case generation from z- specifications with isabelle," in *Proceedings of ZUM'97: The Z Formal Specification Notation*, LNCS 1212. Springer-Verlag, 1997, pp. 52 - 71.

[19] R. M. Hierons, S. Sadeghipour, and H. Singh, "Testing a system specified using statecharts and z," *Information and Software Technology*, vol. 43, no. 2, pp. 137 - 149, 2001.

