www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 4 Issue 9 Sep 2015, Page No. 14163-14171

Automating the test case generation for Object Oriented Systems using Activity Diagrams

Rajvir Singh[1], Preeti[2]

[1]Assistant Professor, CSE Department,
Deenbandhu Chhotu Ram University, Murthal.
Rajvirsingh.cse@dcrustm.org

[2]Student, M.Tech(CSE),
Deenbandhu Chhotu Ram University, Murthal.
Preeti.lohchab32@gmail.com

ABSTRACT: Generation of test cases is the most important issue in the software testing. Thus test cases need to be carefully designed. An activity diagram is used for modelling the dynamic aspects of the system. It helps in visualizing the sequence of activities involved in a control flow. 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. It generates an Activity Graph from the activity diagram by specifying some mapping rules. Then a proposed approach will generate the number of valid paths by using activity graph as an input and these valid paths will help in generating the test cases which will cover all aspects of the system and results in increased system efficiency and performance.

KEYWORDS: Object-oriented Software, Model Based Testing, Activity Diagram, Testing, Design based approach

1. INTRODUCTION

Software testing is one of the main methods to increase the confidence of the programmers in the correctness and reliability of software. Sometimes, programs that are poorly tested perform correctly for months and even years before some input sets reveal the presence of serious errors. Incorrect software that is released to market without being fully tested could result in customer dissatisfaction and moreover it is vitally important for software in critical applications that it is free of software faults which might lead to heavy financial loss or even endanger lives. In the past decades, systematic approaches to software testing procedures and tools have been developed to avoid many difficulties that existed in ad-hoc techniques. Nevertheless, software testing is the most usual technique for error detection today's software industry. The main goal of software testing is to increase one's confidence in the correctness of the program being tested [15].

In the last few years, object-oriented analysis and design (OOAD) has come into existence, it has found widespread acceptance in the industry as well as in academics. The main reason for the popularity of OOAD is that it holds the following promises:

- Code and design reuse
- Increased productivity
- Ease of testing and maintenance
- Better code and design understandability

Software Testing is a time consuming and costly process in software development life cycle. Even after testing, if the system does not respond correctly then a great problem will arise. This may result in performance degradation and reduce the efficiency of the system. So, the main focus is to generate the test cases as maximum as possible by considering most of the aspects of the system. By using UML diagrams it is easy to cover most aspects of the system. And for this UML Activity diagram is used because it comes under Behavioral diagrams which emphasize on the behavior of the software system i.e. how various objects interact with each other [16, 17]. Also, an Activity diagram consists of flowchart of various activities with transition among them. An activity states the internal behaviour of an operation of the system. An activity diagram is used for modeling the dynamic aspects of the system. It emphasizes the sequential or concurrent flow path from activity to activity. Both conditional and parallel activities can be represented by an activity diagram [18].

2. RELATED WORK

Testing plays an important part in software development process to ensure the quality and reliability of the developed product. For Object-oriented systems, model based testing has recently become very popular. A lot of work has been done in this field by the researchers.

Software Testing involves three processes: Test Case Generation, Test Case Execution and Test Case Evaluation. Manual testing is time-consuming, labor-intensive and error-prone. Therefore, it is required to automate the testing effort. Automation, which automates a part of testing process, reduces the human effort in finding bugs and errors.Most of the work in the field of automated test case generation for object-oriented systems focus on the model based approach. A number of researchers have used Models for testing the object oriented systems. This approach has increasingly become popular.

C.D. Turner and D.J. Robson used the concept of FSA (Finite State Automata) for generation of test cases and validation of object-oriented programs. It emphasizes on the validation of interaction between the features of a class[1].

Wang Linzhang, Yuan Jiesong, Yu Xiaofeng, Hu Jun, Li Xuandong and Zheng Guoliang, in their paper [2], derive test scenarios directly from the activity diagram modeling an operation. Then, all the information for test case generation, i.e. input/output sequence and parameters, the constraint conditions and expected object method sequence, is extracted from each test scenario. At last, the possible values of all the input/output parameters could be generated by applying category-partition method, and test suite could be systematically generated to find the inconsistency between the implementation and the design.

Debasish Kundu and Debasis Samanta used UML Activity Diagrams to generate test cases in.They used UML 2.0 syntax for generating test cases from activity diagrams with use case scope. They considered a coverage criterion called activity path coverage criterion with the aim to cover faults like synchronization faults, faults in a loop [3].

Santosh Kumar Swain, Durga Prasad Mohapatra and Rajib Mall in [4], proposed a novel technique by combining state and activity models of the system to construct an intermediate representation which they named as state-activity diagram. This technique is very effective in detecting integration faults.
Monalisa Sarma,Rajib Mall in 'Automatic Test Case Generation from UML Models' use the combination of sequence diagrams and use case diagrams for test case generation and uses sequence diagram message path coverage and use case dependency coverage as metrics for measuring the effectiveness of generated test cases[5].

A use case driven approach has been used by Cle´mentine Nebut, Franck Fleurey, Yves Le Traon, Jean-Marc Je´ze´ quel for automated test case generation for embedded systems [6].
Philip Samuel, Rajib Mall and Sandeep Sahoo have used UML sequence diagrams and dynamic slicing technique in their paper published in 2005. It uses slice coverage as the test coverage criteria, [7].
Sequence diagrams, use case template and class diagrams have been used by Monalisa Sarma and Rajib Mall for system testing of the software, in [8]. This approach ensures sequence diagram message path sequence coverage.

Fanping Zeng, Zhide Chen, Qing Cao and Liangliang Mao used UML state diagram model that represent state transition to generate test cases, in [9]. A variation of this approach has also been proposed which uses state chart diagrams for generation and optimization of test cases, [10].

A requirement prioritization method approach has been proposed by Nicha Kosindrdecha and
Jirapun Daengdej which is based on use case diagram which ensures domain specific requirement coverage, in [11].

A.V.K.Shanthi and DR.G.Mohankumar applied the concept of data mining to generate optimal test cases from UML class diagram, in [12].

Combinational UML models (sequence diagram and Activity diagram, [13] and class diagram, sequence diagram and state chart diagram, [14] have also been used by researchers for automating the test case generation for object-oriented systems.

Some researchers have also applied genetic algorithm to UML sequence diagrams for test case generation, [15], [16]. A multi objective genetic algorithm has been used by Kirandeep Kaur and Vinay Chopra for generating test cases from UML sequence diagram, [16].

3. PROBLEM STATEMENT

In the previous researches, test paths have been generated either by using DFS approach or by using BFS approach. The test paths generated by BFS algorithm are exponential including useful and useless test paths. The test paths generated by DFS algorithm help in eliminating useless test cases and reduce the time complexity. The proposed work is to generate test cases from 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 6.7.1 version.

And the objectives to achieve goal of this research work are:

- To increase the performance of the system by considering maximum cases of failure so that system cannot stop its working during the execution of important task.
- To increase the reusability by collecting lot of information related to that system.
- To analyze various techniques used for test generation in activity diagram, communication diagram, use case diagram and sequence diagram.
- To propose an approach for test case generation of activity diagram using depth first search method and breadth first search.
- To implement the proposed approach by generating test paths from the activity flow graph using breadth first search and depth first search method. An algorithm is used by proposed approach that uses the concept of indegree and outdegree.

The proposed approach reduces the total time required for executing test paths. Thus, it improves performance of test case generation.

4. IMPLEMENTATION

4.1 Functional Diagram of proposed work

The Flow Chart shown by Fig 5.1 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.

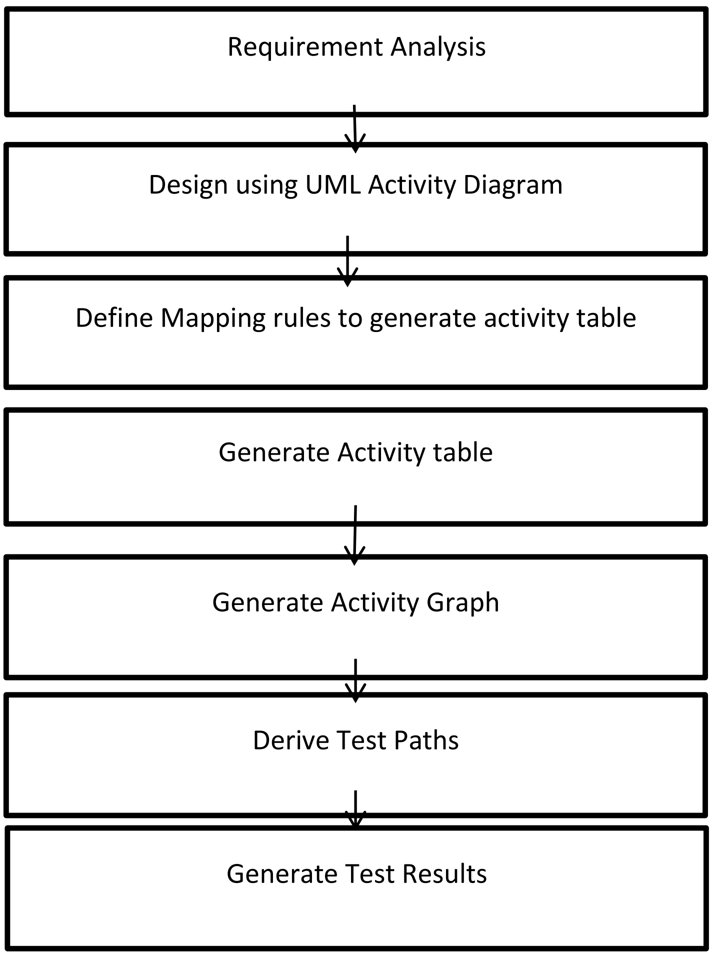

Fig 5.1: Functional Diagram of each task

4.2 Result Analysis

Most of the work has been done by using UML activity diagram. But there is a lot of difference between the proposed work and the previous work. As, in the previous works, test paths have been generated either by using DFS approach or by using BFS approach. The test paths generated by BFS approach are exponential including useful and useless test paths. The test paths generated by DFS approach help in eliminating useless test cases and reduce the time complexity. Also, if there is a use of any one of the method (BFS or DFS) then it will not consider all aspects of the system which results in generating less number of test cases as compared to the proposed approach which is helpful in generating test cases from activity diagram using both the BFS and DFS method. Because, these both methods 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.

5. CASE STUDY

5.1 Issuing a book in library

Here, a library management system is used to show the working of proposed technique, the module that is used for this purpose is issuing a book.

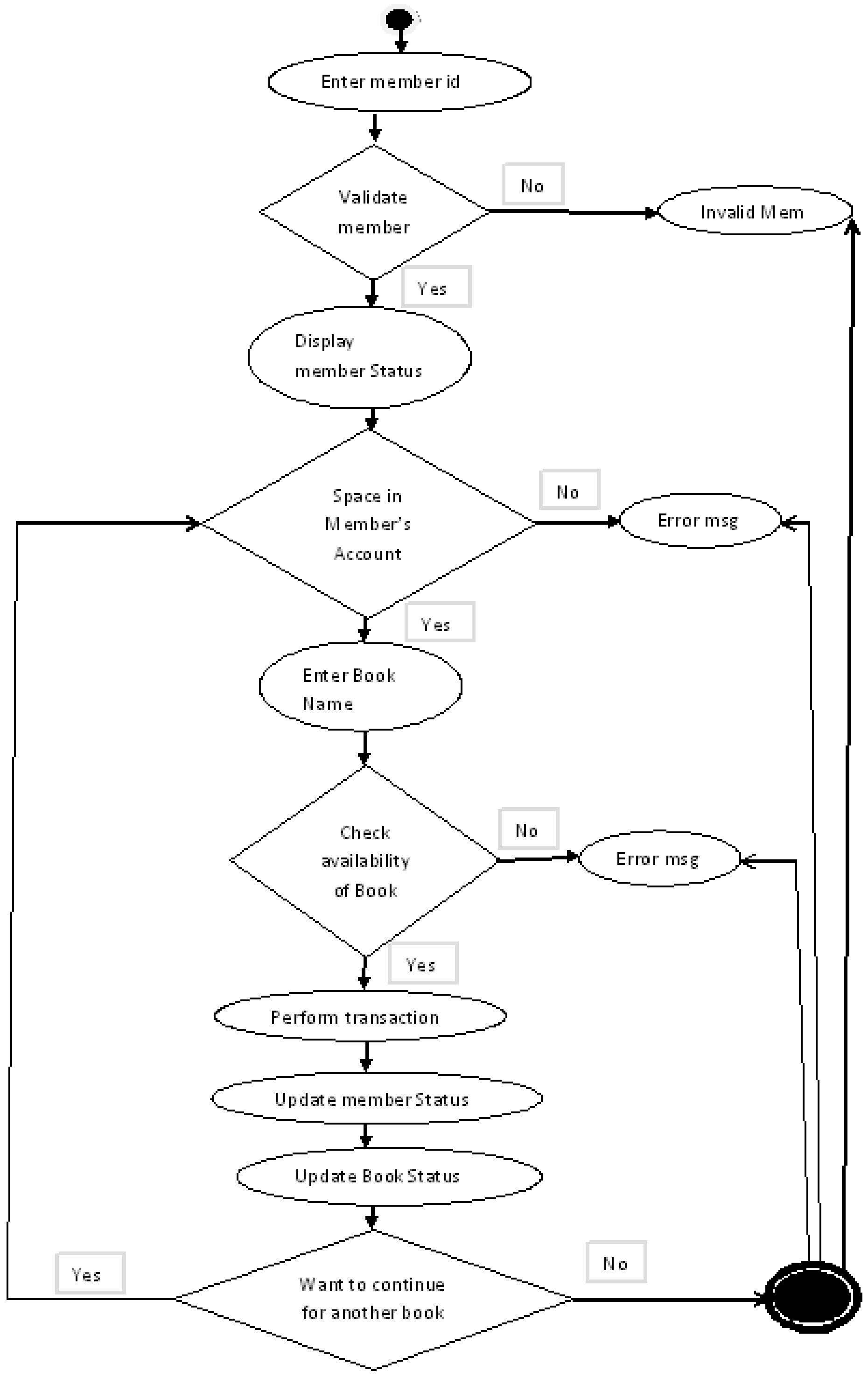

Fig 5.2: UML Activity diagram

DOI: 10.18535/ijecs/v4i9.17

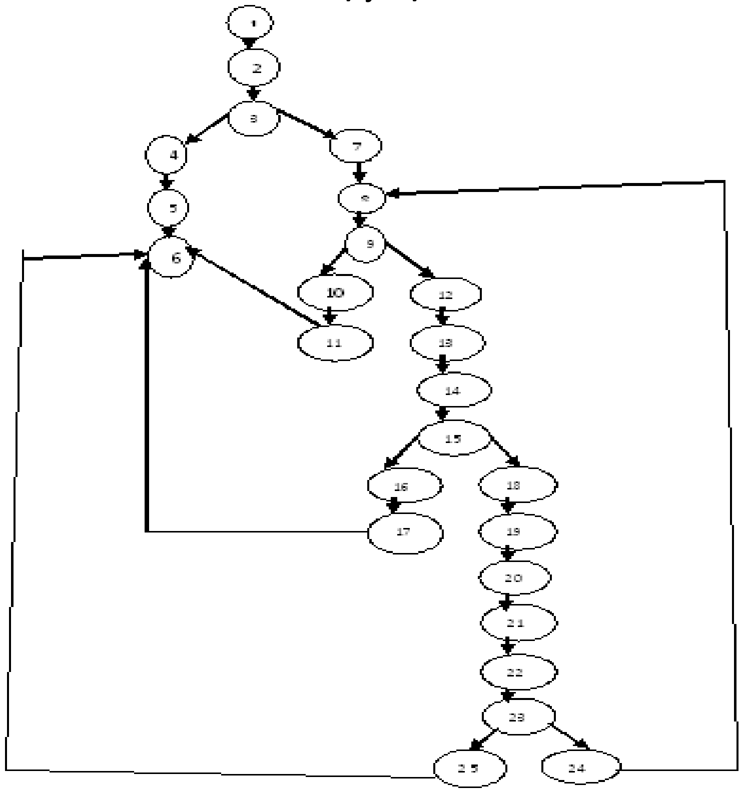

Fig 5.3: Activity graph obtained from activity diagram

5.2 Algorithm: Generate Activity Paths

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

5.3 Test cases

DOI: 10.18535/ijecs/v4i9.17

Table 5.4 Test cases generated by using UML Activity diagram

Test case ID	Pre-condition	Input	Output	Post-condition
TCID1	System is idle and displaying a welcome screen	Enter member card number	Not the valid member.	Displays a welcome screen
TCID2	system is idle and displaying a welcome screen	Enter member card number	Member verified, display member status with message, account is FULL	Displays a welcome screen
TCID3	system is idle and displaying a welcome screen	Enter member card number and author name	Member verified, display member status and an error message, BOOK is not available	Displays a welcome screen
TCID4	system is idle and displaying a welcome screen	Enter member card number, author name and choice	Member verified, display member status, book available, Book issued, update member and book status and enter no for issuing another book	Displays a welcome screen

6. CONCLUSION

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.

7. FUTURE SCOPE

The proposed technique has focused on automated generation of test paths from activity diagram but still there are the following points that can be explored further.
The execution time for generation of test paths can further be reduced.
The proposed approach can be applied to other algorithms like binary search, bubble sort etc. which can further reduce the execution time for test paths in these algorithms.
In the present work, activity diagram is used for a single use case at a time. However, activity diagrams of multiple use cases which are related to each other by various relationships such as, include, extend, generalization /specialization can be considered, which can plan to take up in the future work.

REFERENCES

[1] C.D. Turner, DJ. Robson. The State-based Testing of Object-Oriented Programs
[2] Wang Linzhang, Yuan Jiesong, Yu Xiaofeng, Hu Jun, Li Xuandong and Zheng Guoliang. Generating Test Cases from UML Activity Diagram based on Gray-Box Method
[3] Debasish Kundu, Debasis Samanta. A Novel Approach to Generate Test Cases from UML Activity Diagrams. Chair of Software Engineering, 2008.
[4] Santosh Kumar Swain, Durga Prasad Mohapatra, Rajib Mall. Test case generation based on state and activity models.
[5] Cle´mentine Nebut, Franck Fleurey, Yves Le Traon, Jean-Marc Je´ze´ quell. Automatic Test Generation: A Use Case Driven Approach, 2005.
[6] Philip Samuel, Rajib Mall and Sandeep Sahoo. UML Sequence Diagram Based Testing Using Slicing, 2005.
[7] Monalisa Sarma,Rajib Mall. Automatic Test Case Generation from UML Sequence
diagrams, 2007.
[8] Fanping Zeng, Zhide Chen, Qing Cao, Liangliang Mao. Research on Method of Object-Oriented Test Cases Generation Based on UML and LTS, 2009
[9] Ranjita Kumari Swain, Prafulla Kumar Behera, Durga Prasad Mohapatra. Minimal TestCase Generation for Object-Oriented Software with State Charts, 2012

DOI: 10.18535/ijecs/v4i9.17

[10] Nicha Kosindrdecha, Jirapun Daengdej. A Test Case Generation Technique and Process, 2010.
[11] A.V.K.Shanthi, DR.G.Mohankumar. Automated Test cases generation for Object Oriented Software,2011.
[12] Rohin Verma, Rajesh Bhatia. Behavior based Automated Test Case Generation for Object Oriented Systems, 2012.
[13] V.Mary Sumalatha, G.S.V.P.Raju. Object Oriented Test Case Generation Technique using Genetic Algorithms, 2013.
[14] Kirandeep Kaur, Vinay Chopra. Review of Automatic Test Case Generation from UML Diagram using Evolutionary Algorithm, 2014.
[15] R. Ferguson and B. Korel. The chaning approach for software test data generation. ACM Transactions on Software Engineering Methodology, 5(1):63-86, 1996.
[16] B. N. Biswal and D. P. Mohapatra, Test Case Generation and Optimization of
Object-Oriented Software using UML Behavioral Model, 2010.
[17] R. Singh and V. Arora, A practical approach for model based slicing, 2013.
[18] R. S. Pressman, Software Engineering - A Practitioner's Approach, 2005.