Coverage Based Test Suite Minimization using UML Behavioural Diagrams

Rashmi Gupta*, Vivek Jaglan**
Assistant Professor, CSE Department, Amity University Haryana
Associate professor, CSE Department , Amity University Haryana
goyal.rashmi18@gmail.com

Abstract. There is an important role of software systems testing in the software development cycle. The software testing is used to reveal errors in software which enhances overall quality and efficiency of the software. The approach followed for software testing should be such that it can detect maximum number of errors in minimum time which can lower the effort spent in maintenance phase. Coverage criteria are very important to analyze test cases which can be useful in fault coverage and minimizing the test suite. As it takes a lot of time in the software system testing, so minimizing the test cases can help in minimizing the overall effort required to lower the testing time. Test cases should be selected such that they are covering maximum of activities, basic path and conditions present in the graph and taking less time while executing. Keywords: Test Suite Minimization, Path Coverage, Condition Coverage, UML Diagrams

1. Introduction

The Software testing is continuous process and should be done efficiently. UML models mainly intended towards the reduction of complexity of a problem when the product size and complexity is increasing. But, as far as UML models are considered, they are large, complex and there are huge interactions across a lot of objects. Test Model generation is complicated in large programs due to model’s complexity [1].

Test generation is most costly process in terms of time invested during software system testing. Manual testing consists of having test data generation as the most time-consuming task. As the software has been developing with the advent of time, a lot of research work has been done on the problem of automation data generation; a lot of research work has been done on the problem of automation data generation. During system testing, coverage criteria are appropriate measure to adapt if a test case objective is met [2].

The quality of test cases generation is established using the coverage criteria and software testing considers the criteria combination. We can measure coverage criterion on any program during certain phases. The phases include software development, designing model for the requirements. If those requirements are fulfilled, then a coverage criterion is satisfied. It is prominent to define the coverage criterion because the effectiveness of the test cases generation is measured using the coverage criteria [3].

2. Literature Review

Many investigators propounded criteria of test case coverage according to their requirements and demonstration for the UML Models. We are having number of the criteria for the test cases which are
described in the literature and are based on the activity and the conversion paths [4], and simple path [5] for the activity diagrams. Structural element coverage, non-concurrent and concurrent behavior coverage are included in defined coverage criteria. The detailed exploration of concurrent behavior coverage criteria is not present to address the concurrency errors.

Activity diagram path coverage is proposed by Jiesong et al. [4]. In this coverage, he has discussed the traversal of an activity diagram. The traversal is included from beginning node to final node within a depth first search (DFS) along all movements and traversals and acknowledging the curves are traversed mostly. The path traversed represents a potential functional requirement of the software. However, this coverage criterion is not in relation with concurrency testing, and it does not involve or includes any concurrency objectives.

Kundu and Samanta [6] have worked on coverage criteria for activity diagrams which includes basic path and concurrent node coverage criterion for activity diagrams. They have presented basic path coverage criterion, where they have suggested checking loop conditions for true and false paths. They have presented criteria for activity diagrams having concurrent activities, where they have suggested selecting of concurrent path generated from DFS-BFS algorithm.

Swain et al. [7] have discussed the coverage criteria for UML sequence diagrams in which they used control flow-based coverage criteria. The criteria they have used is about generalization of CF such as branch coverage and basic path coverage where branch represents the interaction among objects in the diagram. They generated an IRCFG graph which defines the various types of coverage techniques. They have worked on coverage criteria for all RCFG paths and all RCFG branches present in the sequence diagram.

Mingsong et al. [5] have worked on coverage criteria for concurrent nodes present in the activity diagrams. They have suggested using a fixed representative path from all concurrent paths having same order of execution of simple path from that activity diagram. They have also discussed about the basic activity diagram. The diagram covers all the basic path of the diagram where each basic path is having all activities which are occurring only once in the diagram.

The automatic coverage analysis for state machine is proposed by Faria et al. [8]. With the help of this analysis tool, modified condition/decision coverage (MC/DC) and boundary value guard of machine are analyzed. The elements in this model are represented with different color schemes which aides in graphical indication for coverage achieved.

3. Test Case Generation

Test cases are generated from combinational graph made by integrating activity and sequence diagrams. Activity diagram is an UML Behavioral diagram which is used to represent the complete process flow of the software in terms of activities while sequence diagram helps to understand the interaction between the objects in sequential manner. Fig 1(a) and (b) depicts the activity diagram and graph of railway reservation system.

A case study of railway reservation system is represented by activity and sequence diagrams which are further transformed into activity graph and sequence graph respectively. Then combinational graph is generated by merged the common activity and sequence nodes together. Fig 2(a) and (b) depicts the sequence diagram and graph of railway reservation system.
3.1 Transformation of UML Activity Diagram into Activity Graph:

Fig 1(a): Activity Diagram of Railway Reservation system

Fig 1(b): Activity graph
3.2 Transformation of UML Sequence diagram into Sequence Graph:

Fig 2(a): Sequence diagram of Railway Reservation System

Fig 2(b): Sequence graph
3.3. Formation of Combinational Graph:

Fig 3: Combinational UML Activity Sequence Graph
The combinational graph is generated by integrating activity and sequence diagram.

The test cases for above graph are generated by computing cyclomatic complexity and are used to generate independent paths for the software. The graph shown in fig 3 is traversed using DFS (Depth First Search) traversal algorithm as it gives better results for activity and sequence graphs [9].

The cyclomatic complexity for the graph is computed by formula given as [10]:

\[
\text{Cyclomatic complexity} = 2+E-N
\]

where \( E \) = the count of all edges and \( N \) = count of all nodes in the graph[13].

| Test id | Independent Path                  |
|---------|-----------------------------------|
| Tc1     | 1,2,3,5,6                         |
| Tc2     | 1,2,3,4,7,8,6                     |
| Tc3     | 1,2,3,4,7,9,10,11,12,13,6         |
| Tc4     | 1,2,3,4,7,9,10,11,12,14,15,16,17,18,19,20,21,6 |
| Tc5     | 1,2,3,4,7,9,10,11,12,14,15,16,17,18,19,20,22,23,24,6 |

4. UML Activity Coverage Criterion

To define a quality metrics for the quality of test requirement, we have used coverage criteria. The coverage criteria will be evaluated based on the conditions and rules with respect to the test case. Hitherto, we have observed control flow and dataflow-based coverage metrics. With the help of coverage metric, tests created are more holistic in term of testing software system while it covers the elements in the diagram.

Coverage parameter is a key indicator in overall system software testing robustness. It gives us confidence in the overall testing process. Coverage metric is very helpful metric and parameter in marking the test impact parameters on software systems. Coverage metric is being used as a guiding mechanism for generation of tests, continuation of generation and aborting the generation. The following section contains the different type of coverage metrics such as basic path coverage, condition coverage and activity coverage [9].

Following section describes about the prestige metric for coverage. Here element coverage is the base equation. Coverage percentage is shown in equation (2) [10]. This equation shows the elements, in generated test cases, count present in UML diagram.

\[
\text{Ecov}= \frac{(\text{EtCovS}/\text{EtcovUML}) \times 100}{(2)}
\]

Where Ecov: coverage elements
EtCovS: elements count exercised in the tests
EtcovUML: elements count present in the UML diagram.
4.1 All Activity Coverage: all-activity coverage of the activity graph can be achieved when all the activities of UML activity diagram is visited at least once [11].

4.2 Condition coverage: It is the metric which calculates percentage of all conditions covered by that test case to the total no. of conditions in the graph [12].

4.3 Path Coverage: It is the metric which calculates percentage of path covered by test case to the total path in the graph [13].

For all these coverage criteria, result is calculated for all test cases of the graph shown in fig 3 and Table 2 shows the result analysis of Path coverage and condition coverage.

| Sr. No. | Test id | Activity Covered | Activity Coverage (%) | Total Path Covered | Path Coverage (%) | Condition Covered | Condition Coverage (%) |
|---------|---------|------------------|-----------------------|--------------------|------------------|-------------------|------------------------|
| 1.      | Tc1     | 5                | 20.83                 | 4                  | 15.38            | 1                 | 20                     |
| 2.      | Tc2     | 7                | 29.17                 | 6                  | 23.08            | 2                 | 40                     |
| 3.      | Tc3     | 11               | 45.83                 | 10                 | 38.46            | 3                 | 60                     |
| 4.      | Tc4     | 18               | 75.00                 | 17                 | 65.38            | 4                 | 80                     |
| 5.      | Tc5     | 20               | 83.33                 | 19                 | 73.08            | 4                 | 80                     |

Fig 4 depicts the test case with their coverage which clearly indicates the Tc3 to Tc5 have better coverage as compared to test cases Tc1 and Tc2.

5. Conclusion

This paper emphasizes on practical coverage criteria on combinational graph formed by integration of activity and sequence diagrams. Coverage criteria are very important to analyze test cases which can be useful in fault coverage and minimizing the test suite resulting more efficient software testing process. The paper presents the analysis of path coverage, activity coverage and condition coverage of each test case. From the results, it has been concluded that Tc3, Tc4 and Tc5 are having maximum path coverage and condition coverage. These test cases should be executed first to reveal maximum fault. For future, Fault coverage can also include more test case minimization.
References

[1] Ali, S., 2010 A systematic review of the application and empirical investigation of search-based test case generation. IEEE Transactions on Software Engineering, p 742-762.
[2] Oluwagbemi, O. and Asmuni H., 2015 Automatic Generation of Test Cases from Activity Diagrams for UML Based Testing (UBT) Jurnal Teknologi, p 215-223.
[3] Swain, R.K., Behera P.K. and Mohapatra D. P.,2012 Minimal Test Case Generation for Object-Oriented Software with State Charts. 1208-2265.
[4] Jiesong Y., Linzhang W, Xiaofeng Y., Jun H., Xuandong H., and Guoliang Z. 2004 Generating Test Cases from UML Activity Diagram based on Gray-Box Method. In Proceedings of the 11th Asia-Pacific Software Engineering Conference, p 284-291.
[5] Mingsong C., Xiaokang Q., and Xuandong L. 2006 . Automatic Test Case Generation for UML Activity Diagrams. In Proceedings of the 2006 International Workshop on Automation of Software Test, AST '06,p 2-8. ACM.
[6] Kundu D. and Samanta D 2009. A Novel Approach to Generate Test Cases from UML Activity Diagrams Journal of Object Technology, p 65-83.
[7] Swain J., Rountev A. and Kagan S. 2005 Coverage Criteria for Testing of Object Interactions in Sequence Diagrams. In Fundamental Approaches to Software Engineering, volume 3442 of Lecture Notes in Computer Science, p 289-304, Springer.
[8] Faria J., Ferreira R., and Paiva. A. 2010 Test Coverage Analysis of UML State Machines. In Proceedings of the Third International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), p 284-289. IEEE.
[9] Gupta R and Jaglan V 2019 Test case generation for UML behavioral diagram by traversal algorithm International Journal of Innovative Technology and Exploring Engineering (IJITEE) p 3262-3266.
[10] Dawood Y., Rajeb M.M., Hashim N.L. and Romli R.B. 2017 Coverage criteria for test case generation using UML state chart diagram AIP Conference Proceedings
[11] Panda S and Mohapatra DP 2017 Regression test suite minimization using integer linear programming model Software: Practice And Experience ,Wiley Online Library, p 243-265.
[12] Yoo S. and Harman S. 2012 Regression testing minimization, selection and prioritization : a survey Software Testing, Verification And Reliability , p 67-120.
[13] Bajaj A. and Sangwan O.P. 2019 A Systematic Literature Review of Test Case Prioritization Using Genetic Algorithms, volume 7, p 126355-126375, IEEE Access.