"Analysis of Test Case Generation Using Activity Diagram"

Latharani T R

Research Scholar

Abstract – Software testing phase is a critical and most important phases in the software development life cycle (SDLC). In general, the software testing phase takes around 40-70% of the effort, time, and cost. This area is well researched over a long period of time [5] and this paper is based on test case generation using activity diagram.

Keyword: Test Case Generation, Activity Diagram

----------♦----------

INTRODUCTION

A test automation program should be considered when key program factors indicate the overall development program is not meeting expectations and there are no cost-effective alternatives to test automation. The key indicators from a program objective would be:

- quality objectives are not being met
- defect escape velocity into production is deemed unacceptable
- target deployment velocity is not being met because testing is perceived as a bottleneck
- testing is not being completed within the assigned timeframe

These factors are often experienced by the project team at the same time -- basically the test group has reached its testing capacity. When this happens, the leadership team has two alternatives: grow the manual testing team or launch a test automation proof-of-concept (POC) to determine if automation can address the testing deficit.

It should be noted that testing is often not the "true" bottleneck, but as long as it is perceived as the bottleneck other systemic quality issues will not be addressed. Test automation is one way to remove the perception of a testing bottleneck empowering the testing team while allowing the program to focus on other systemic quality issues for example the initial quality of the code.

REVIEW OF LITERATURE:

UML activity diagrams as intend stipulation and consider the automatic approach to test case generation by expanding [1]. UML diagrams are classified on the basis of the structural or behavior aspects of systems, i.e. whether they are planned to describe the structural or behavior features of systems. UML activity diagrams [2-3] describe the chronological or concurrent control flow of behavior. They can be used to model the dynamic aspect of a group of objects, or the organize flow of an operation. Our move toward first constructs the activity drawing for the given problem and then arbitrarily generates the initial test cases for a PUT [4]. Randomly generated initial test case helps to produce best test case using heuristic rule by satisfying the path exposure criteria. Our approach is concerned to develop a technique that will automatically generate test cases with most path coverage.

A large part of current testing research aim at improving the degree of attainable automation, either by developing the advanced techniques for generating the test inputs, or, beyond test generation, by finding innovative support procedures to automate the testing process [6].

According to Ben-Gurion, Software testing activities are usually planned by human experts, while test automation tools are limited to execution of pre-planned tests only [7]. If for example the types of errors are defined and classified in each phase of the developing life cycle, they could be used in future predictions in similar types of projects, hence they could be avoided in different similar projects [8].

ACTIVITY DIAGRAM

UML gives a number of diagrams to explain particular aspects of software artifacts. These diagrams can be classified depending on whether they are proposed to describe structural or behavioral features of systems. Activity diagrams also explain the sequence of activities among the objects concerned in the control flow during put into practice.

Activity diagrams are alike to practical flow charts. But the major difference between them is that activity diagrams support explanation of parallel activities and organization aspects involved in different actions. Before presenting the detailed procedure to generate test cases using UML *activity diagram*, we need to define the *activity diagram*.

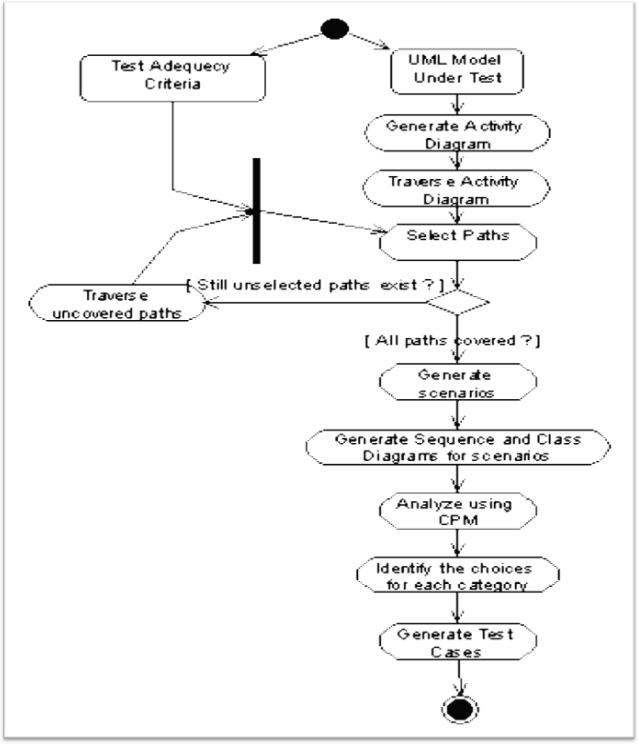

Fig 1: Test Case Generation using Design Models

TEST CASE GENERATION :

After the test scenarios are generated, we examine the equivalent sequence diagram for each preferred scenario. Each sequence diagram is collected of objects and the messages they exchange. The objects concerned in sequence diagram realize and carry out the functionalities explained in the scenario through embellishment and message exchanges. In this phase, class diagrams are also measured, as class diagram describe operations and attributes required for the interactions of their objects.

CONCLUSION:

Testing allows developers to deliver software that meets expectations, prevents unexpected results, and improves the long term maintenance of the application. Depending upon the purpose of testing and the software requirements, the appropriate methodologies are applied, where possible, testing can be automated.

REFERENCES:

[1] A. Abdurazik and J. Offutt, “Using uml collaboration diagrams for static checking and test generation,” in Proceedings of the third International Conference on the UML. York, UK: Lecture Notes in Computer Science, Springer-Verlag GmbH, 2000, pp. 383 – 395.

[2] “Object management group,” available at http://www.omg.org/uml, 2003.

[3] R. E. Prather and J. P. M. Jr., “The path prefix software testing strategy,” IEEE Transactions on Software Engineering, vol. 13, no. 7, pp. 761–766, July 1987.

[4] G. Booch, J. Rumbaugh, and I. Jacobson,The Unified Modeling Language User Guide. Addison-Wesley, 2001.

[5] Nicha Kosindrdecha and Jirapun Daengdej, A test case generation technique and process, Autonomous System Research Laboratory Faculty of Science and Technology Assumption University, Thailand.

[6] Antonia Bertolino“Software Testing Research: Achievements, Challenges, Dreams.” IEEE, Future Of Software Engineering (FOSE’07). 0-7695-2829-5/07-2007.

[7] Oded Maimon, Lior Rokach, “Data Mining and Knowledge Discovery Handbook”, Springer (2006).

[8] Ali Ilkhani1, Golnoosh Abaee2 1 Department, Extracting Test Cases by Using Data Mining; Reducing the Cost of Testing, International Journal of Computer Information Systems and Industrial Management Applications. ISSN 2150-7988 Volume 3 (2011) pp. 730-737