



Automation of Software Testing Process using UML

¹Dr. K.P. Yadav, ²Dr. Saroj Patel, ³Tannu Arora

¹CSE/ Director, ²Applied Science/Professor, ³CSE/Ph.D Scholar

¹MTU, Noida/MIET, Greater Noida, ^{2,3}JNU, Jodhpur
INDIA

Abstract: The software testing community had much less awareness and debate about UML than software design and development communities, and has largely been absent as the modeling standard was developed. This is an important issue, because in many software development organizations, the cost of testing can account for more than 40% of the total development cost for a software system. In software engineering, system modeling is the process of formulating a representation of a real system in an abstract way to understand its behavior. Software testing encourages reusing these models for testing purpose. This expedites the process of test case generation. The objective of this paper is to explore the possibility of using the UML for software testing and how much does development of code cost with and without the models. This paper emphasis on automation of software testing which is widely used in organizations to reduce manual effort and project cost. The testing process that encourages the automation of software testing is Model based testing i.e, MBT. This testing process is based on using the modeling techniques that are defined in Unified modeling language. UML structural and behavioral specification diagrams have been used by testing researchers for generation of test scenarios and test data.

Keywords: UML, MBT, Software testing, Modelling techniques, Behavioral diagrams

I. Introduction

Information technology provides tools for communication and interaction between citizens and governance thus increasing transparency and quality of legislation. The changes to information technology are not restricted on how we use information technology but they also affect how we develop software. Software development has changed considerably over the past decade. As a result, testing has gained more importance and it can be said that the testing phase is one of the most important phases of software development cost wise as well as for measuring quality of the software.

Software testing is an empirical investigation which is conducted to provide information about the software under test. Testing is performed for evaluating product quality, and for improving it, by identifying defects and problems. Software testing consists of the *dynamic* verification of the behavior of a program on a *finite* set of cases, suitably *selected* from the usually infinite executions domain, against the *expected* behavior. Testing is performed for evaluating product quality, and for improving it, by identifying defects and problems.

Software development engineering refers to a range of development approaches that are based on the software modeling as the primary form of expression. These models are sometimes constructed to a certain level of detail, and then code is written by hand in a separate step. Sometimes complete models are built including executable actions. Code can be generated from the models, ranging from system skeletons to complete, deployable products [3].

II. Automation of Software Testing

In general, the workload of software test would increase exponentially with software size. This is because when the number of conditions increases, the number of necessary tests also increases. To overcome this issue, it is natural to adopt the automation of tests with the accompanying quality and efficiency. In software testing, test automation is the use of special model to control the execution of tests and the comparison of actual outcomes with predicted outcomes. Test automation can automate some repetitive but necessary tasks in a formalized testing process already in place, or add additional testing that would be difficult to perform manually [7].

The Unified Modeling Language (UML) has emerged as an industrial standard for modeling software systems, and has been presented to the International Organization for Standardization for consideration as an international standard. UML has received a great deal of attention (both positive and negative) from the software design and development communities, and work is ongoing to enhance and expand its capabilities [2]. However, the software testing community had much less awareness and debate about UML, and has largely been absent as the modeling standard was developed. This is an important issue, because in many software development organizations, the cost of testing can account for more than 40% of the total development cost for a software system.

III. Use of UML

In software engineering, system modeling is the process of formulating a representation of a real system in an abstract way to understand its behavior. The UML is a visual modeling language that can be used to "specify, visualize, construct, and document the artifacts of a software system". The language itself is specified using a 4-layer meta-model architecture, which partitions the UML into three logical subpackages: Foundation, Behavioral Elements, and Model Management. UML provides seven views into a software system. These are the static, use case, state machine, activity, interaction, physical, and model management views.

The Foundation package provides the basic infrastructure for exploring the static structure of systems. This infrastructure includes:

- class diagrams - class structure and associations
 - component diagrams- map classes to software units
 - deployment diagrams - physical system structure
- The Behavioral Elements package provides the linguistic elements for modeling the dynamic behavior of the system and includes:
- use case diagrams - functionality as seen by actors
 - interaction diagrams - object message sequences
 - activity diagrams - business / software processes
 - state diagrams - object state transition behavior

In seeking to explore the issues that arise when using UML in the software testing process, we will focus primarily on the diagrams in the Behavioral Elements package. This is because most of the activities in software testing seek to discover defects that arise during the execution of a software system, and these defects are generally dynamic (behavioral) in nature.

Software testing encourages reusing these models for testing purpose. This expedites the process of test case generation. UML behavioral specification diagrams have been used by testing researchers for generation of test scenarios and test data.

IV. Software Testing and UML

There are many phases in the testing process, including unit, function, system, regression testing. The following table illustrates the differences between these phases, as well as the potential UML diagram for use in the phase.

Table I Testing types vs. UML diagrams

Test Type	Coverage Criteria	Fault Model	UML Diagram
Unit	Code	correctness, error handling pre / post conditions, invariants	class and state diagrams
Function	Functional	functional and API behavior, integration issues	Interaction and class diagrams
System	Operational Scenarios	workload, contention, synchron., recovery	use case, activity, and interaction diagrams
Regression	Functional	Unexpected behavior from new / changed function	Same as Functional testing type i.e. interaction and class diagrams

V. Model Based Testing

Model-based testing is a software testing in which test cases are derived in whole or in part from a model that describes some functional aspects of the system under test (SUT). The model is usually an abstract, partial presentation of the system under test's desired behaviour[7]. The test cases derived from models are functional tests on the same level of abstraction as the model.

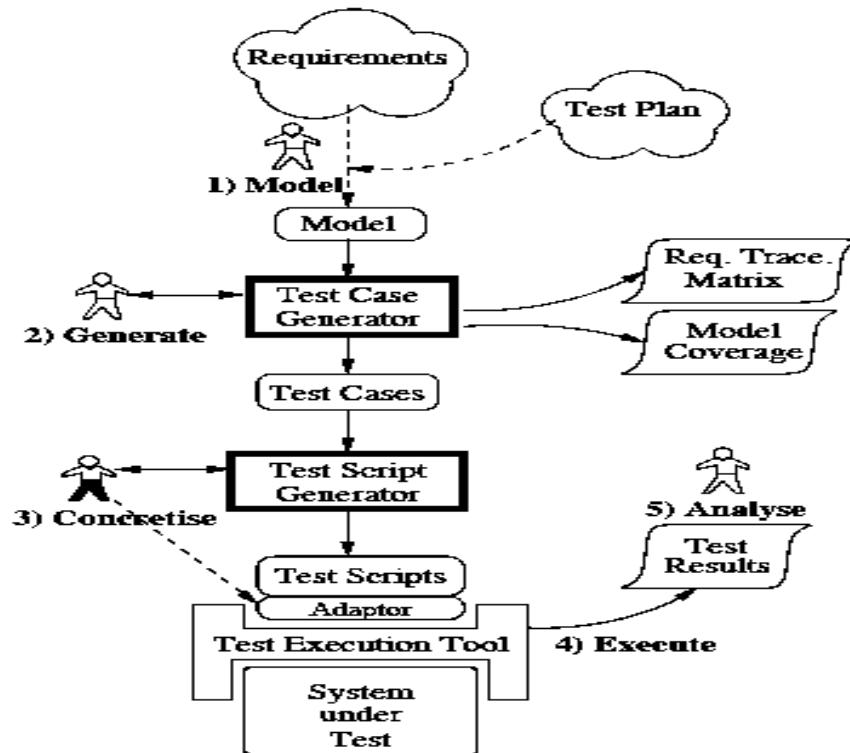
A MBT Process

MBT Process as shown in Fig. 1 includes:

1. Model the SUT and/or its environment
2. Generate abstract tests from the model

3. Concretize the abstract tests to make them executable
4. Execute the tests on the SUT and assign verdicts
5. Analyze the test results (and take corrective action)

Figure 1 MBT Process



B Features of MBT

- automatic execution
- auto regression testing
- auto design of tests
- systematic coverage
- measure coverage of model and requirements

C Benefits of MBT

- **SUT Fault detection**
Increase the possibility of finding errors
- **Reduced testing cost and time**
Less time and effort spent on writing tests and analyzing results
Could generate shortest test sequences
- **Improved test quality**
Possible to measure the "quality" by considering coverage (of model)
- **Requirements defect detection**
Modeling phase exposes requirements issues
- **Traceability**
Between requirements and the model
Between informal requirements and generated test cases
- **Requirements evolution**
Update test suite to reflect new requirements: update model and do it automatically

VI. Conclusion

In this paper use of modeling techniques are considered in software testing process. Software testing is very critical step in software development process in terms of effort and development cost. So there is a need to automate software testing process by using UML that incorporates model based testing.

References

- [1] Jim A., Ila N., "UML 2 AND THE UNIFIED PROCESS Practical Object-Oriented Analysis and Design", Second Edition, Pearson 2006

- [2] Rumbaugh, J., I. Jacobsen, and G. Booch, "The Unified Modelling Language Reference Manual", Second Edition , Addison Wesley, 2005
- [3] Anda, B., Hansen, K., Gullesten, I., Thorsen, H. K., "Experiences from Introducing UML-Based Development in a Large Safety-Critical Project Springer Science", 2006, pp. 558 -576
- [4] Anda, B., Hansen, K., "A Case Study on the Application of UML in Legacy Development", 2006, ACM
- [5] Dobing, B., Parsons, J., "How UML is Used", May 2006, Communications of the ACM, pp. 109, 110
- [6] Shivkumar Hasmukhrai Trivedi, "Software Testing Techniques", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10, October 2012
- [7] Howard Foster, Antonia Bertolino, J. Jenny Li, "Automation of Software Test", 6th International Workshop, 2011