International Journal of Information Technology and Knowledge Management
July-December 2009, Volume 2, No. 2, pp. 343-346

ROLE OF TESTING IN PHASES OF SDLC AND QUALITY

Youddha Beer Singh[*] & Shivani Goel[**]

Software testing is an important technique for assessing the quality of a software product. In this paper, various types of testing and various attributes of software quality are explained. Identifying the types of testing that can be applied for checking a particular quality attribute is the aim of this research paper. All types of testing can not be applied in all phases of software development life cycle. Which testing types are applicable in which phases of life cycle of software development is also summarized.

Keywords: Quality Attributes, Phases of SDLC, Testing Technique, Static Attributes, Dynamic Attributes.

1. Introduction to Testing

Software testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item [8]. Software testing is an activity that should be done throughout the whole development process [6] Software testing is one of the "verification and validation," or V&V, software practices. Some other V&V practices, such as inspections and pair programming. Verification (the first V) is the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase [7]. Verification activities include testing and reviews. For example, in the software for the Monopoly game, we can verify that two players cannot own the same house. Validation is the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements [7]. At the end of development validation (the second V) activities are used to evaluate whether the features that have been built into the software satisfy the customer requirements and are traceable to customer requirements.

If we fail to deliver a reliable, good and problem free software solution, we fail in our project and probably we may lose our client. So in order to make it sure, that we provide our client a proper software solution, we go for testing. We check out if there is any problem, any error in the system, which can make software unusable by the client. We make software testers test the system and help in finding out the bugs in the system to fix them on time. We find out the problems and fix them and again try to find out all the potential problems. Software testing consists of several subcategories, each of which is done for different purposes, and often using different techniques.

There are two main purpose of Testing [3]

1. To evaluate quality or acceptability of that which is being tested.
2. To discover problems or errors

2. Objective of Testing

There are four main objective of testing [1] are:

1. **Demonstration:** It show that the system can be used with acceptable risk, demonstrate functions under special conditions and show that products are ready for integration or use.
2. **Detection:** It discover defects, errors, and deficiencies. Determine system capabilities and limitations quality of components, work products, and the system.
3. **Prevention:** It provide information to prevent or reduce the number of errors clarify system specifications and performance. Identify ways to avoid risks and problems in the future.
4. **Improving Quality:** By doing effective testing, we can minimize errors and hence improve the quality of software.

3. Various Types of Testing

There are two basic classes of software testing, black box testing and white box testing. For now, you just need to understand the very basic difference between the two classes, clarified by the definitions below [11]:

1. Black box testing (also called functional testing) is testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions. Examples of black box testing are: Equivalence class partitioning, Boundary value analysis, cause-effect graphs,

[*,**] CSED, Thapar University, Patiala
E-mail: youdhhabeersingh@gmail.com[*]
E-mail: shivani@thapar.edu[**]

comparison testing, acceptance testing, Functional testing.

2. White box testing (also called structural testing and glass box testing) is testing that takes into account the internal mechanism of a system or component. Examples of white box testing are: Basis path testing, structural testing, Statement/Branch/Condition/Loop/Path Coverage testing, Data-flow testing, Domain and boundary testing, Logical based testing, Fault based testing, Unit testing, syntax testing, table testing, desk checking.

4. Applying Various Types of Testing in Software Development Life Cycle Phases

Various phases of software development life cycle are : requirements gathering and analysis, design, coding, integration , implementation and maintenance. The testing phase can be used in all of these life cycle phases as an umbrella activity. *V* model of testing given by Mr. Perry includes only 5 phases of SDLC. Here we extend this model to include more phases of SDLC and select the types of testing that we can apply in each phase:

Table 1
Various Types of Testing in Software Development Life Cycle Phases

Life Cycle Phase	*Types of Testing*	*Person Involved in Testing*
Planning for testing	Exploratory testing, ad hoc testing, Free form testing, risk based testing, structured walkthrough,	Testers
Requirements gathering	Prototyping , Requirement Testing	
Analysis	Decision tables	
Design	JADs, integration testing, system testing	
Coding:	Code and Unit testing, basis path testing, boundary value testing, branch coverage testing, condition coverage testing, desk checking, loop coverage testing, statement coverage testing, syntax testing, table testing,	Developer
Integration:	Integration and system testing, black box testing, white box testing, bottom up testing, CRUD testing, Database testing, end to end testing, equivalence partitioning, incremental integration testing, inspections, positive and negative testing, sandwich testing, thread testing, top down testing,	
Implementation	acceptance testing, alpha testing, beta testing, cause-effect graphing, comparison testing, compatibility testing, Exception testing, load testing, mutation testing, orthogonal array testing, performance testing, stress testing, prior detect history testing, random testing, range testing, recovery testing, state transition testing, robustness testing, penetration testing, security testing, Back to back testing	Users, Developer
Maintenance:	Maintenance and Regression testing histogram testing, pareto testing, run chart, statistical profile testing,	

5. Quality

Quality is defined as "the essential character of omitting, an inherent or distinguishing character". There are two generally accepted meanings of quality [10]. The first is that quality means "meeting requirements." With this definition, to have a quality product, the requirements must be measurable, and the product's requirements will either be met or not met. The second is, the quality definition by the customer "whether the product or service does what the customer needs". Another way of wording it is "fit for use." There should also be a description of the purpose of the product, typically documented in a customer "requirements specification". The requirements are the most important document, and the quality system revolves around it. In addition, quality attributes are described in the customer's requirements specification. Examples include usability, the relative ease with which a user communicates with the application; portability, the capability of the system to be executed across a diverse range of hardware architectures; and reusability, the ability to transfer software components constructed in one software system into another. We can classify all quality features into two categories based on the point where they can be applied i.e. on code or application

5.1 Static Attributes

Static attributes refer to the actual code (maintainable and testable code) and related documentation (Correct and complete documentation).

5.2 Dynamic Attributes

Dynamic attributes refer to the behaviour of the application while in use. Reliability, correctness completeness, consistency, usability, and performance.

5.3 How to Measure Quality ?

In order to measure quality, we need to analyse requirements to design test cases, then design the test cases, document them, implement them and execute these test cases. Then the results are analysed. Before all this, we need to plan for testing, including risk analysis and test management practices. An example is IBM RUP software tools used by testers to execute a software test plan [11]. This all includes communication skill for the effective tester.

5.4 Various Quality Attributes Are:

Understandability: The purpose of the software product is clear. This goes further than just a statement of purpose-all of the design and user documentation must be clearly written so that it is easily understandable. Obviously, the user context must be taken into account, e.g. if the software product is to be used by software engineers it is not required to be understandable to lay users.

Completeness: All parts of the software product are present and each of its parts are fully developed. For example, if the code calls a sub-routine from an external library, the software package must provide reference to that library and all required parameters must be passed. All required input data must be available.

Conciseness: No excessive information is present. This is important where memory capacity is limited, and it is important to reduce lines of code to a minimum. It can be improved by replacing repeated functionality by one sub-routine or function which achieves that functionality. This quality factor also applies to documentation.

Portability: The software product can be easily operated or made to operate on multiple computer configurations. This can be between multiple hardware configurations (such as server hardware and personal computers), multiple operating systems (e.g. Microsoft Windows and Linux-based operating systems), or both.

Consistency: The software contains uniform notation, symbology and terminology within itself.

Maintainability: The product should facilitates updating to satisfy new requirements and software product that is maintainable is simple, well-documented

Testability: The software product facilitates the establishment of acceptance criteria and supports evaluation of its performance. Such a characteristic must be built-in during the design phase if the product is to be easily testable, since a complex design leads to poor testability.

Usability: The product is convenient and practicable to use. The component of the software which has most impact on this is the user interface (UI), which for best usability is usually graphical.

Reliability: The software can be expected to perform its intended functions satisfactorily over a period of time. Reliability also encompasses environmental considerations in that the product is required to perform correctly in whatever conditions it is operated in; this is sometimes termed robustness.

Structure: The software possesses a definite pattern of organization in its constituent parts.

Efficiency: The software product fulfills its purpose without wasting resources, e.g. memory or CPU cycles.

Security: The product is able to protect data against unauthorized access and to withstand malicious interference with its operations. Besides the presence of appropriate security mechanisms such as authentication, access control and encryption, security also implies reliability in the face of malicious, intelligent and adaptive attackers:

6. APPLICATION OF TESTING TYPES TO MEASUREMENT OF QUALITY ATTRIBUTES

We can also categorize various types of testing according to the quality feature it applies to in the given Table:

Table 2
Testing Technique According to Quality Features

Quality Attribute	*Types of Testing*
Functionality	Functional testing
Security	Security testing
Complexity	Unit testing
Performance	Performance testing
Compatibility	Compatibility testing
Reliability	Stress testing, Robustness testing, load testing
Vulnerability	Penetration testing
Usability	Comparison testing
Consistency	Database testing, Table testing
Correctness	Database testing, Table testing
Portability	
Recovery	Recovery testing
Completeness	Boundary/Statement/loop/condition/path coverage testing
Efficiency	Performance testing
Understandability	
Structure	Structural testing
Maintainability	Regression testing

CONCLUSION

Quality is the main focus of any software engineering project. Without measuring, we cannot be sure of the level of quality in software. So the methods of measuring the quality is software testing technique. This paper relates various types of testing that we can apply in measuring various quality attributes. Also which testing are related to various phase of SDLC.

REFERENCES

[1] Software Program Managers Network, *Little Book of Testing*, **1**, (1998).

[2] Kit, Ed, *Software Testing in the Real World*, Addison-Wesley, (1995), 3.

[3] Jorgensen, Paul C., *Software Testing A Craftsman's Approach*, CRC Press, (1995) 3.

[4] Software Program Managers Network, *Little Book of Testing*, **II**, (1998).

[5] Program Manager's Guide for Managing Software, 0.6, 29 (June 2001), Chapter 11.

[6] A. Bertolino, "Chapter 5: Software Testing," in *IEEE SWEBOK Trial Version* 1.00, (May 2001).

[7] IEEE, "IEEE Standard 610. 12-1990, IEEE Standard Glossary of Software Engineering Terminology," (1990).

[8] IEEE, "ANSI/IEEE Standard 1008-1987, IEEE Standard for Software Unit Testing," No. 1986.

[9] Boris Beizer, Software Testing Techniques. Second Edition, 1990.

[10] Boehm, B. W., and Others, "Characteristics of Software Quality," *TRW Software Series*, December 22, 1973.

[11] IEEE Standard for a Software Quality Metrics Methodology (*IEEE Standard* P-1061/D21). New York, N.Y.: Institute of Electrical and Electronic Engineers, Inc.,1990.

[12] Ballista Paper: Multi-Version Comparison Method to Find Silent and Hindering Failures.