CHAPTER

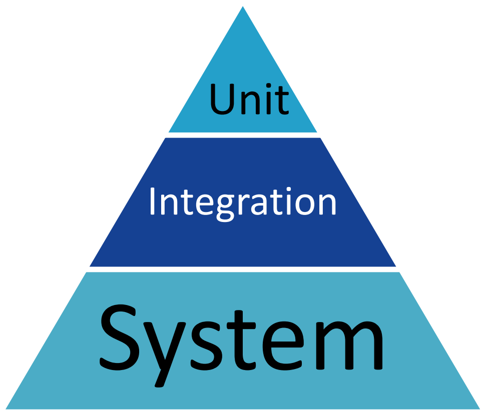
1

Testing Overview

This chapter describes the basic definition and concepts of Testing from Software point of view.

What is testing?

Testing is the process of evaluating a system or its component(s) with the intent to find that whether it satisfies the specified requirements or not. This activity results in the actual, expected and difference between their results. In simple words testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements.



According to ANSI/IEEE 1059 standard, Testing can be defined as "A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item".

Who does testing?

It depends on the process and the associated stakeholders of the project(s). In the IT industry, large companies have a team with responsibilities to evaluate the developed software in the context of the given requirements. Moreover, developers also conduct testing which is called Unit Testing. In most cases, following professionals are involved in testing of a system within their respective capacities:

- Software Tester
- Software Developer
- Project Lead/Manager
- End User

Different companies have difference designations for people who test the software on the basis of their experience and knowledge such as Software Tester, Software Quality Assurance Engineer, and QA Analyst etc.

It is not possible to test the software at any time during its cycle. The next two sections state when testing should be started and when to end it during the SDLC.

i. *When to Start Testing*

An early start to testing reduces the cost, time to rework and error free software that is delivered to the client. However in Software Development Life Cycle (SDLC) testing can be started from the Requirements Gathering phase and lasts till the deployment of the software. However it also depends on the development model that is being used. For example in Water fall model formal testing is conducted in the Testing phase, but in incremental model, testing is performed at the end of every increment/iteration and at the end the whole application is tested.

Testing is done in different forms at every phase of SDLC like during Requirement gathering phase, the analysis and verifications of requirements are also considered testing. Reviewing the design in the design phase with intent to improve the design is also considered as testing. Testing performed by a developer on completion of the code is also categorized as Unit type of testing.

ii. *When to Stop Testing*

Unlike when to start testing it is difficult to determine when to stop testing, as testing is a never ending process and no one can say that any software is 100% tested. Following are the aspects which should be considered to stop the testing:

- Testing Deadlines.
- Completion of test case execution.
- Completion of Functional and code coverage to a certain point.
- Bug rate falls below a certain level and no high priority bugs are identified.
- Management decision.

Difference between Verification & Validation

These two terms are very confusing for people, who use them interchangeably. Let’s discuss about them briefly.

Verification	Validation
Are you building it right?	Are you building the right thing?
Ensure that the software system meets all the functionality.	Ensure that functionalities meet the intended behavior.
Verification takes place first and includes the checking for documentation, code etc.	Validation occurs after verification and mainly involves the checking of the overall product.
Done by developers.	Done by Testers.
Have static activities as it includes the reviews, walkthroughs, and inspections to verify that software is correct or not.	Have dynamic activities as it includes executing the software against the requirements.
It is an objective process and no subjective decision should be needed to verify the Software.	It is a subjective process and involves subjective decisions on how well the Software works.

Difference between Testing, Quality Assurance and Quality Control

Most people are confused with the concepts and difference between Quality Assurance, Quality Control and Testing. Although they are interrelated and at some level they can be considered as the same activities, but there is indeed a difference between them. Mentioned below are the definitions and differences between them:

Quality Assurance	Quality Control	Testing
Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements.	Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements.	Activities which ensure the identification of bugs/error/defects in the Software.
Focuses on processes and procedures rather then conducting actual testing on the system.	Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process.	Focuses on actual testing.
Process oriented activities.	Product oriented activities.	Product oriented activities.
Preventive activities	It is a corrective process.	It is a preventive process
It is a subset of Software Test Life Cycle (STLC)	QC can be considered as the subset of Quality Assurance.	Testing is the subset of Quality Control.

Difference between Audit and Inspection

Audit: A systematic process to determine how the actual testing process is conducted within an organization or a team. Generally, it is an independent examination of processes which are involved during the testing of software. As per IEEE, it is a review of documented processes whether organizations implements and follows the processes or not. Types of Audit include the Legal Compliance Audit, Internal Audit, and System Audit.

Inspection**:** A formal technique which involves the formal or informal technical reviews of any artifact by identifying any error or gap. Inspection includes the formal as well as informal technical reviews. As per IEEE94, Inspection is a formal evaluation technique in which software requirements, design, or code are examined in detail by a person or group other than the author to detect faults, violations of development standards, and other problems.

Formal Inspection meetings may have following process: Planning, Overview Preparation, Inspection Meeting, Rework, and Follow-up.

Difference between Testing and Debugging

Testing: It involves the identification of bug/error/defect in the software without correcting it. Normally professionals with a Quality Assurance background are involved in the identification of bugs. Testing is performed in the testing phase.

Debugging: It involves identifying, isolating and fixing the problems/bug. Developers who code the software conduct debugging upon encountering an error in the code. Debugging is the part of White box or Unit Testing. Debugging can be performed in the development phase while conducting Unit Testing or in phases while fixing the reported bugs.

Testing Myths

Given below are some of the more popular and common myths about Software testing.

Myth: Testing is too expensive.

Reality: There is a saying, pay less for testing during software development or pay more for maintenance or correction later. Early testing saves both time and cost in many aspects however, reducing the cost without testing may result in the improper design of a software application rendering the product useless.

Myth: Testing is time consuming.
Reality: During the SDLC phases testing is never a time consuming process. However diagnosing and fixing the error which is identified during proper testing is a time consuming but productive activity.

Myth: Testing cannot be started if the product is not fully developed.

Reality: No doubt, testing depends on the source code but reviewing requirements and developing test cases is independent from the developed code. However iterative or incremental approach as a development life cycle model may reduce the dependency of testing on the fully developed software.

Myth: Complete Testing is Possible.

Reality: It becomes an issue when a client or tester thinks that complete testing is possible. It is possible that all paths have been tested by the team but occurrence of complete testing is never possible. There might be some scenarios that are never executed by the test team or the client during the software development life cycle and may be executed once the project has been deployed.

Myth: If the software is tested then it must be bug free.

Reality: This is a very common myth which clients, Project Managers and the management team believe in. No one can say with absolute certainty that a software application is 100% bug free even if a tester with superb testing skills has tested the application.

Myth: Missed defects are due to Testers.

Reality: It is not a correct approach to blame testers for bugs that remain in the application even after testing has been performed. This myth relates to Time, Cost, and Requirements changing Constraints. However the test strategy may also result in bugs being missed by the testing team.

Myth: Testers should be responsible for the quality of a product.

Reality: It is a very common misinterpretation that only testers or the testing team should be responsible for product quality. Tester's responsibilities include the identification of bugs to the stakeholders and then it is their decision whether they will fix the bug or release the software. Releasing the software at the time puts more pressure on the testers as they will be blamed for any error.

Myth: Test Automation should be used wherever it is possible to use it and to reduce time.

Reality: Yes it is true that Test Automation reduces the testing time but it is not possible to start Test Automation at any time during Software development. Test Automaton should be started when the software has been manually tested and is stable to some extent. Moreover, Test Automation can never be used if requirements keep changing.

Myth: Any one can test a Software application.

Reality: People outside the IT industry think and even believe that any one can test the software and testing is not a creative job. However testers know very well that this is myth. Thinking alternatives scenarios, try to crash the Software with the intent to explore potential bugs is not possible for the person who developed it.

Myth: A tester's task is only to find bugs.

Reality: Finding bugs in the Software is the task of testers but at the same time they are domain experts of the particular software. Developers are only responsible for the specific component or area that is assigned to them but testers understand the overall workings of the software, what the dependencies are and what the impacts of one module on another module are.

Testing and ISO Standards

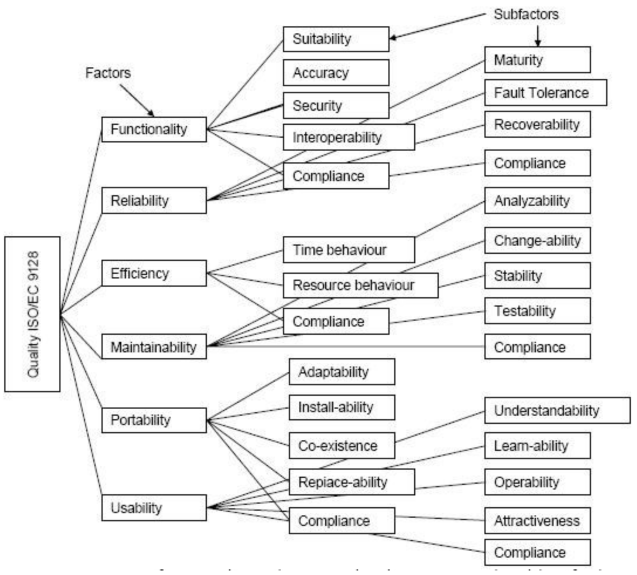
Many organizations around the globe are developing and implementing different Standards to improve the quality needs of their Software. The next section briefly describes some of the widely used standards related to Quality Assurance and Testing. Here is a definition of some of them:

ISO/IEC 9126: This standard deals with the following aspects to determine the quality of a software application:

- Quality model
- External metrics
- Internal metrics
- Quality in use metrics.

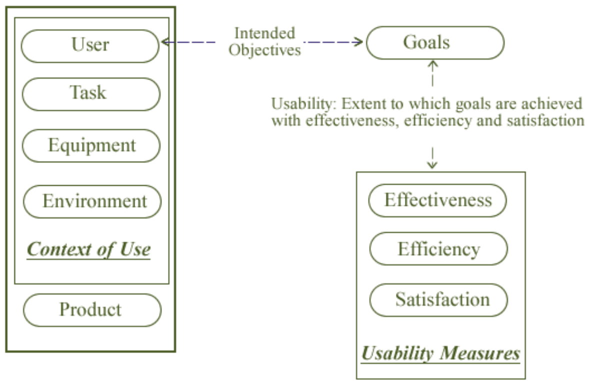
This standard presents some set of quality attributes for any Software such as:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability



The above mentioned quality attributes are further divided into sub-factors. These sub characteristics can be measured by internal or external metrics as shown in the graphical depiction of ISO-9126 model.

ISO/IEC 9241-11: Part 11 of this standard deals with the extent to which a product can be used by specified users to achieve specified goals with Effectiveness, Efficiency and Satisfaction in a specified context of use.
This standard proposed a framework which describes the usability components and relationship between them. In this standard the usability is considered in terms of user performance and satisfaction. According to ISO 9241-11 usability depends on context of use and the level of usability will change as the context changes.



ISO/IEC 25000: ISO/IEC 25000:2005 is commonly known as the standard which gives the guidelines for Software product Quality Requirements and Evaluation (SQuaRE). This standard helps in organizing and enhancing the process related to Software quality requirements and their evaluations. In reality, ISO-25000 replaces the two old ISO standards i.e. ISO-9126 and ISO-14598.

SQuaRE is divided into sub parts such as:

- ISO 2500n - Quality Management Division.
- ISO 2501n - Quality Model Division.
- ISO 2502n - Quality Measurement Division.
- ISO 2503n - Quality Requirements Division.
- ISO 2504n - Quality Evaluation Division.

The main contents of ***SQuaRE*** are:

- Terms and definitions.
- Reference Models.
- General guide.
- Individual division guides.
- Standard related to Requirement Engineering (i.e. specification, planning, measurement and evaluation process

ISO/IEC 12119: This standard deals with Software packages delivered to the client. It does not focus or deal with the client's (the person/organization whom Software is delivered) production process. The main contents are related to the following items:

- Set of Requirements for Software packages.
- Instructions for testing the delivered Software package against the requirements.

Some of the other standards related to QA and Testing processes are:

IEEE 829: A standard for the format of documents used in different stages of software testing.
IEEE 1061: A methodology for establishing quality requirements, identifying, implementing, analyzing, and validating the process and product of software quality metrics is defined.
IEEE 1059: Guide for Software Verification and Validation Plans.
IEEE 1008: A standard for unit testing.
IEEE 1012: A standard for Software Verification and Validation.
IEEE 1028: A standard for software inspections.
IEEE 1044: A standard for the classification of software anomalies.
IEEE 1044-1: A guide to the classification of software anomalies.
IEEE 830: A guide for developing system requirements specifications.

IEEE 730: A standard for software quality assurance plans.
IEEE 1061: A standard for software quality metrics and methodology.
IEEE 12207: A standard for software life cycle processes and life cycle data.
BS 7925-1: A vocabulary of terms used in software testing.
BS 7925-2: A standard for software component testing.