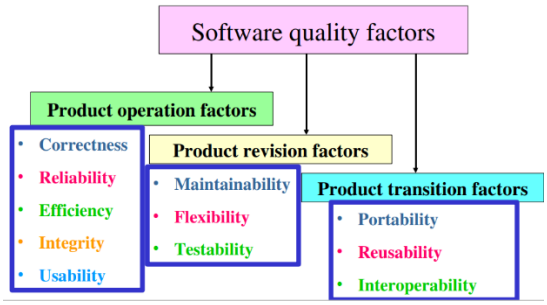
Software Quality: Software Quality is defined as the conformance to explicitly state functional and performance requirements, explicitly documented development standards, and inherent characteristics that are expected of all professionally developed software.

- McCall's software quality factor model



The following factors are used to measure Software Development Quality. Each attribute can be used to measure product performance.

Maintainability

Different versions of the product should be easy to maintain. For development, it should be easy to add code to the existing system, should be easy to upgrade for new features and new technologies from time to time.

Maintenance should be cost-effective and easy. The system is easy to maintain and correct defects or make a change in the software.

Reliability

Measure if the product is reliable enough to sustain in any condition. Should give the correct results consistently. Product reliability is measured in terms of working of the project under different working environments and different conditions.

Portability

This can be measured in terms of Costing issues related to porting, Technical issues related to porting, and Behavioral issues related to porting.

This can be measured in terms of ease of use. The application should be user-friendly. It should be easy to learn. Navigation should be simple.

Usability

The system must be:

- Easy to use for input preparation, operation, and interpretation of the output.
- Provide consistent user interface standards and conventions with our other frequently used systems.
- Easy for new or infrequent users to learn to use the system.

Flexibility

Should be flexible enough to modify. Adaptable to other products with which it needs interaction. Should be easy to interface with other standard 3rd party components.

Understandability

Software understandability not only describes the understanding of the source code, including code readability and code complexity , but also includes non-source code software artifacts generated during the development process, such as documentations, issue summaries, development emails and commit messages .

Efficiency

It is one of the major system quality attributes. It is measured in terms of time required to complete any task given to the system. **For example,** the system should utilize processor capacity, disk space, and memory efficiently. If the system is using all the available resources then the user will get degraded performance failing the system for efficiency. If the system is not efficient, then it cannot be used in real-time applications.

Integrity or Security

Integrity comes with security. System integrity or security should be sufficient to prevent unauthorized access to system functions, prevent information loss, ensure that the software is protected from virus infection, and protect the privacy of data entered into the system.

What Is Cost of Quality?

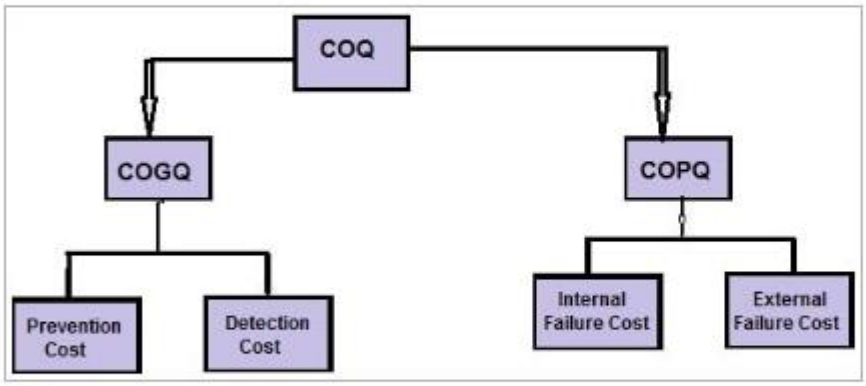
COQ is the cost incurred to deliver a quality product to the client. The cost includes the cost of all activities taken up proactively in a planned way to prevent defects and deliver good quality.

It is not possible to deliver a zero-defect product, the idea is to minimize the expected failure for the committed requirements i.e. both Functional and Non-functional requirements. Expectations can be tangible and intangible attributes. So, setting clarity in expectations is very crucial.

Cost Of Quality Formula

COQ = Cost of control + Cost of failure of control or

COQ = Cost of Good Software Quality + Cost of Poor Software Quality



COQ

Cost of Good Quality (COGQ) = Prevention costs + Detection costs

Cost of Poor Quality (COPQ) = Internal failure costs + External failure costs

While it is most desirable to check for all quality points, every check is a cost overhead, so a balance needs to strike, and clarity of the quality coverage expected should be very clear from the beginning. This coverage should be shared with all stakeholders.

Quality management is about striking a good balance and prioritizing the high-risk quality dimensions achievable within a desirable/optimal budget.

Quality is really crucial, but it need not be attained at the micro-level of all features. The trick is to achieve the **right balance between quality and cost** and achieve the best that is required by the client at an optimal cost.

Multiple Roles of Software Testers – People Relationship

Before going to the People Challenges, I would like to reiterate Few Useful Points on Role of Software Testers:

– Testing is finally recognized as a profession which needs specialized skills set and qualifications. – Also it is convinced testing will not start once the development is completed.

– Testing needs to be start in every stage of the software development Life Cycle.

– Testers need to validate whether the requirements specification documents is developed based on the needs of the organization.

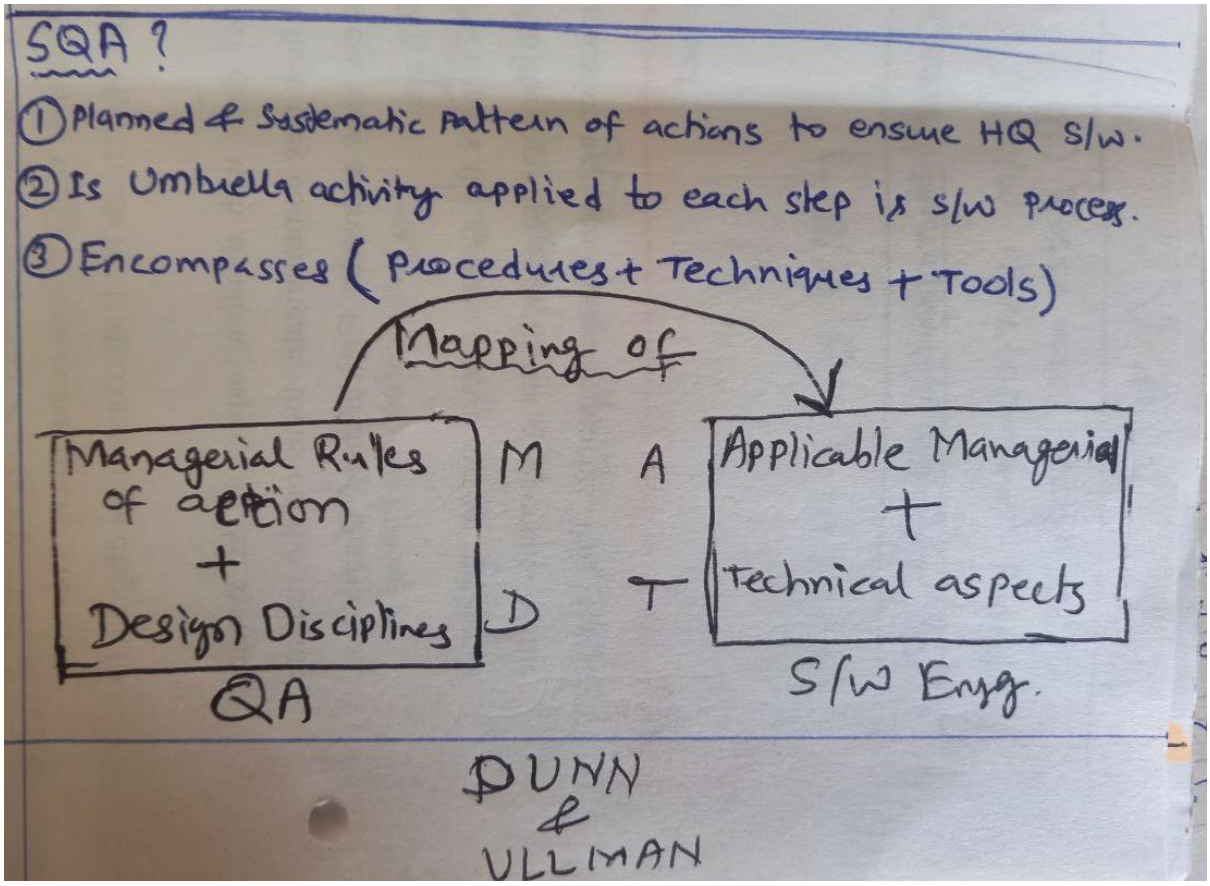
– Testers need to be ensuring that the design documents developed based on the requirements specifications.

– Testers need to be ensure that the test cases and test plan are created based on the requirements specifications.

– Testers need to be ensuring that the defects need to find out before the test cycle starts so that the cost spends will gradually decreases.

– Testers need to be ensuring that the defects can be reproducible by developers.
– Testers need to be ensuring that they are finding defects of the high priority scenarios first.
– If you are in Agile methodology, then testers need to make sure that defects are getting fixed on priority.
– Finding out defects in the later stage of the product will leads the cost spend to rectify the defects is very costly.

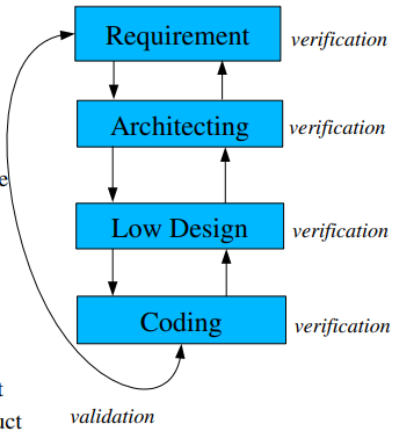
Software Quality Assurance :



SQA

SQE includes

- Verification
  - are we building the product right ?
  - performed at the end of a phase to ensure that requirements established during previous phase have been met
- Validation
  - are we building the right product ?
  - performed at the end of the development process to ensure compliance with product requirements



FAILURE, ERROR, FAULT, AND DEFECT

In the literature on software testing, one can find references to the terms *failure*, *error*, *fault*, and *defect*. Although their meanings are related, there are important distinctions between these four concepts. In the following, we present first three terms as they are understood in the fault-tolerant computing community:

- **Failure:** A failure is said to occur whenever the external behavior of a system does not conform to that prescribed in the system specification.
- **Error:** An error is a *state* of the system. In the absence of any corrective action by the system, an error state could lead to a failure which would not be attributed to any event subsequent to the error.
- **Fault:** A fault is the adjudged cause of an error.

A fault may remain undetected for a long time, until some event activates it. When an event activates a fault, it first brings the program into an intermediate error state. If computation is allowed to proceed from an error state without any corrective action, the program eventually causes a failure. As an aside, in fault-tolerant computing, corrective actions can be taken to take a program out of an error state into a desirable state such that subsequent computation does not eventually lead to a failure. The process of failure manifestation can therefore be succinctly represented as a behavior chain [31] as follows: fault $\rightarrow$ error $\rightarrow$ failure. The behavior chain can iterate for a while, that is, failure of one component can lead to a failure of another interacting component.

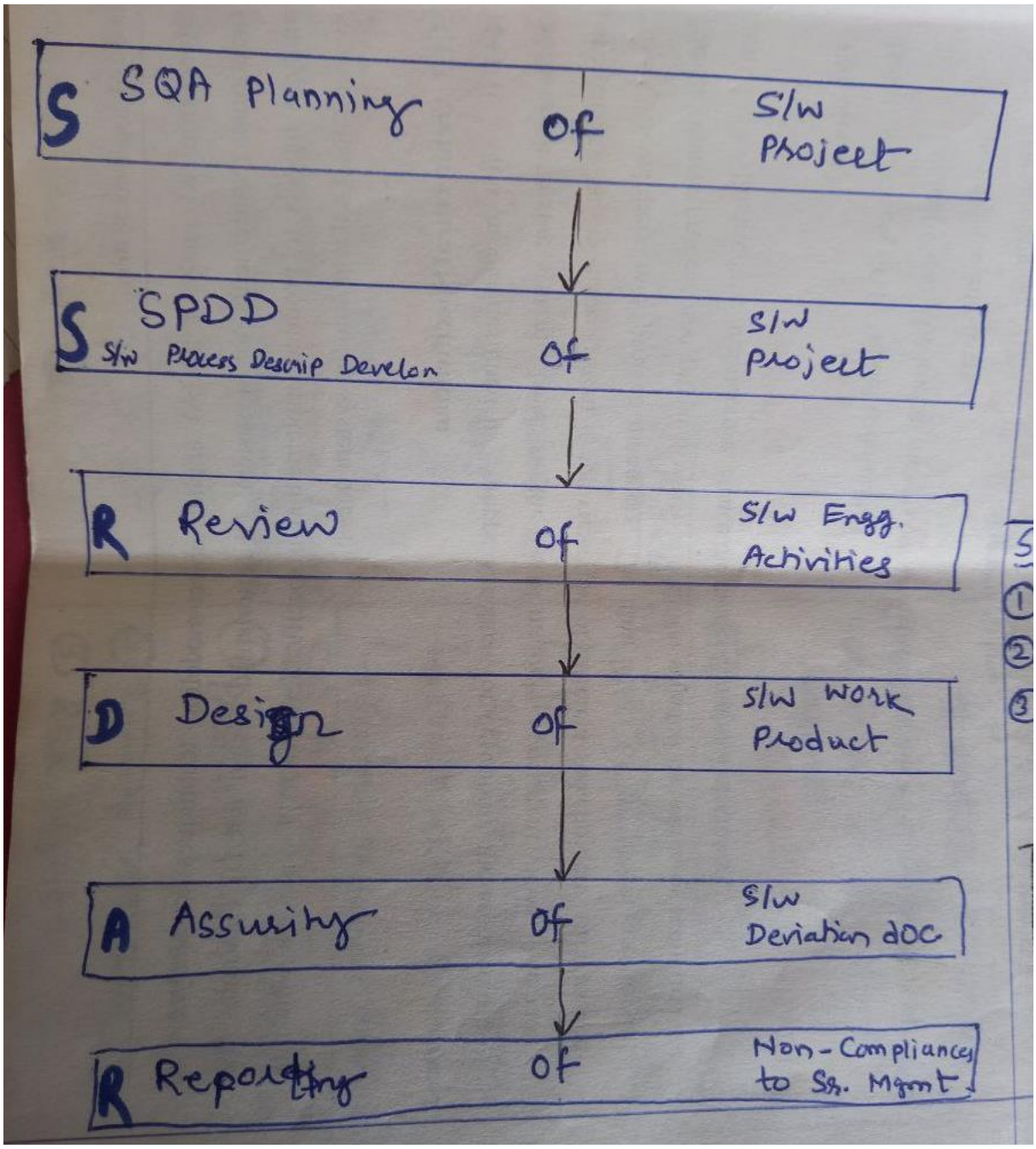
The above definition of failure assumes that the given specification is acceptable to the customer. However, if the specification does not meet the expectations of the customer, then, of course, even a fault-free implementation fails to satisfy the customer. It is a difficult task to give a precise definition of fault, error, or failure of software, because of the "human factor" involved in the overall acceptance of a system. In an article titled "What Is Software Failure" [32], Ram Chillarege commented that in modern software business software failure means "the customer's expectation has not been met and/or the customer is unable to do useful work with product," p. 354.

Roderick Rees [33] extended Chillarege's comments of software failure by pointing out that "failure is a matter of function only [and is thus] related to purpose, not to whether an item is physically intact or not" (p. 163). To substantiate this, Behrooz Parhami [34] provided three interesting examples to show the relevance of such a view point in wider context. One of the examples is quoted here (p. 451):

> Consider a small organization. *Defects* in the organization's staff promotion policies can cause improper promotions, viewed as *faults*. The resulting ineptitudes & dissatisfactions are *errors* in the organization's state. The organization's personnel or departments probably begin to *malfunction* as result of the errors, in turn causing an overall *degradation* of performance. The end result can be the organization's *failure* to achieve its goal.

There is a fine difference between defects and faults in the above example, that is, execution of a defective policy may lead to a faulty promotion. In a software context, a software system may be defective due to design issues; certain system states will expose a defect, resulting in the development of faults defined as incorrect signal values or decisions within the system. In industry, the term defect is widely used, whereas among researchers the term fault is more prevalent. For all practical purpose, the two terms are synonymous. In this book, we use the two terms interchangeably as required.

Process of Software Quality Assurance :



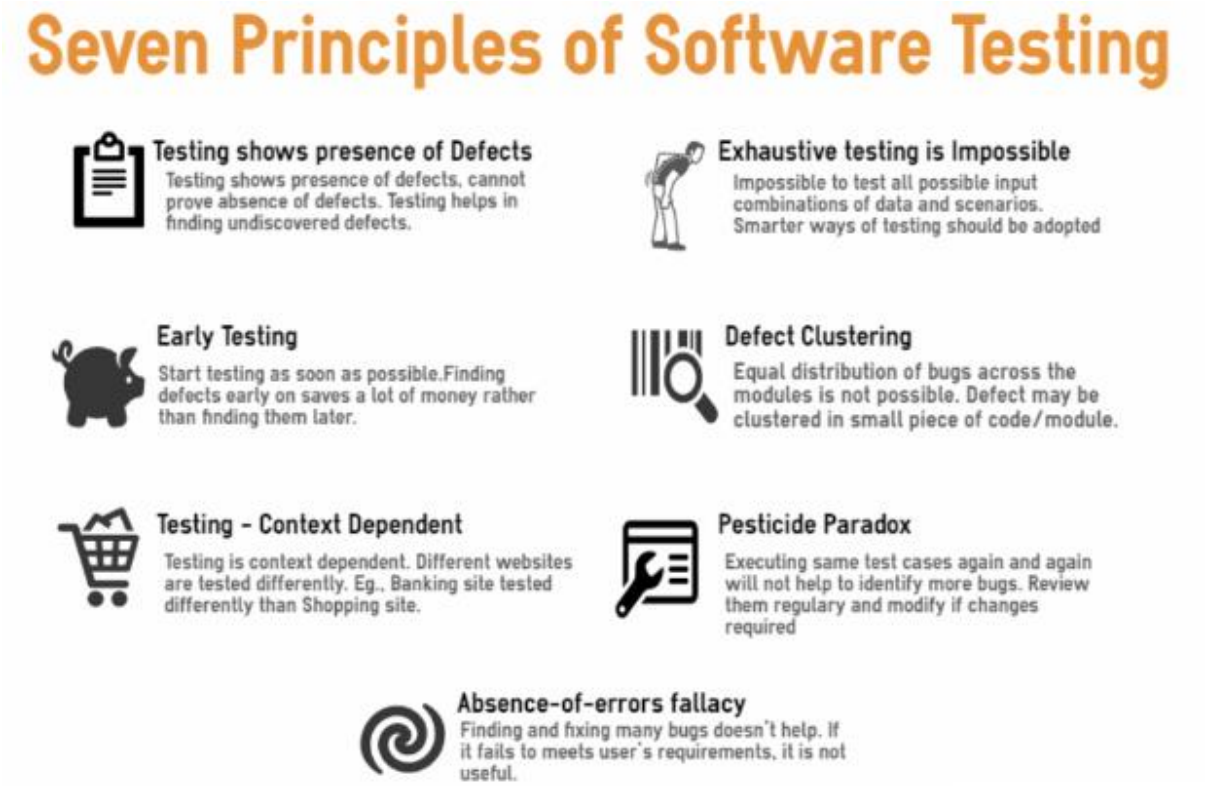
Software Testing:

Testing is a quality assurance activity. It involves the execution of software and observation of its behaviour or outcome with intent of finding errors.

Main purpose of testing:

(1) Demo of proper behaviour (to provide the evidence of quality)
(2) To detect & fix problems.

Principles of testing:



V model is now one of the most widely used software development processes. The introduction of the V model has actually proved the implementation of testing right from the requirement phase. V model is also called a verification and validation model.

Verification and Validation

To understand the V model, let's first understand what is verification and validation in software.

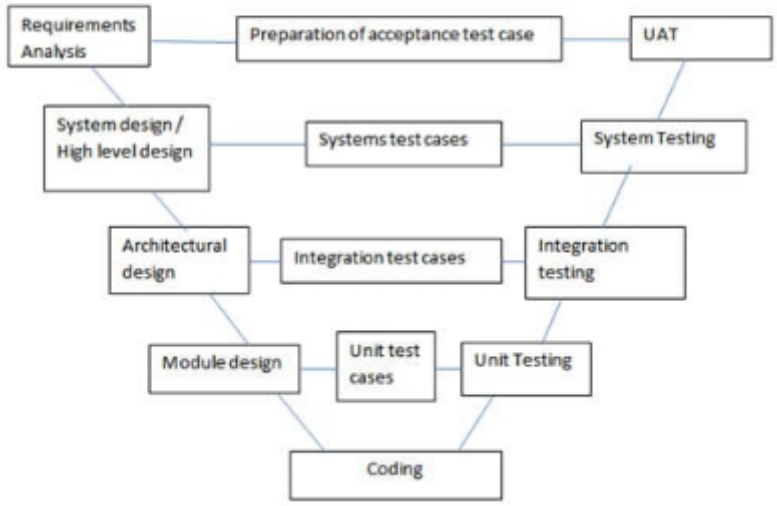
Verification: Verification is a static analysis technique. In this technique, testing is done without executing the code. Examples include – Reviews, Inspection, and walkthroughs.

Validation: Validation is a dynamic analysis technique where testing is done by executing the code. Examples include functional and non-functional testing techniques.

V-Model

In the V model, the development and QA activities are done simultaneously. There is no discrete phase called Testing, rather testing starts right from the requirement phase. The verification and validation activities go hand in hand.

To understand the V model, let's look at the figure below:



In a typical development process, the left-hand side shows the development activities and the right-hand side shows the testing activities. I should not be wrong if I say that in the development phase both verification and validation are performed along with the actual development activities.

When To Use The V Model?

V model is applicable when:

- The requirement is well defined and not ambiguous
- Acceptance criteria are well defined.
- The project is short to medium in size.
- The technology and tools used are not dynamic.

Pros and Cons of using V model

PROS	CONS
- Development and progress is very organized and systematic	-Not suitable for bigger and complex projects
- Works well for smaller to medium sized projects.	- Not suitable if the requirements are not consistent.
- Testing starts from beginning so ambiguities are identified from the beginning.	- No working software is produced in the intermediate stage.
- Easy to manage as each phase has well defined objectives and goals.	- No provision for doing risk analysis so uncertainty and risks are there.

Independent Testing

The term is defined as independent because it's neither connected with the producer nor the user. However, the test is either affiliated with the government, some organization or by an independent testing laboratory. This can be connected and used for all sorts of web applications, iOS, and Android platforms. This also consists of some third-party testing referred to as ***contrast testing*** or ***evaluation facilities***.

Purpose

1. It determines that work is done according to the needs of the specification, regulation, or not.
2. It figures out if the upcoming products/programs are on track.
3. Supplies standard data for scientific engineering and quality assurance functions.
4. Validate suitable for end-user or not.
5. Provides a medium for technical communication.
6. Gives evidence in legal proceedings, product claims, etc.

Benefits of Independent Testing

1. Finds out more defects as compared to other testers working inside the programming team.
2. Unique side assumptions and ideas of independent testers result in identifying hidden defects.
3. The independent testers are unbiased.
4. Cost-effective as it has a separate budget, which helps in tracking money spent on training, testing tools, equipment.
5. Provides improved Software Quality.
6. Supplies more Experienced and skilled power.
7. One can easily switch between manual and automation testing using independent testing due to being more flexible.
8. Reduces time to market by providing access to expert skills in test automation skills ensures faster testing cycles.

Levels of testing

There are four main levels of testing that need to be completed before a program can be cleared for use: unit testing, integration testing, system testing, and acceptance testing.

Unit Testing

During this first round of testing, the program is submitted to assessments that focus on specific units or components of the software to determine whether each one is fully functional. The main aim of this endeavor is to determine whether the application functions as designed. In this phase, **a unit can refer to a function, individual program or even a procedure**, and a White-box Testing method is usually used to get the job done. One of the biggest benefits of this testing phase is that it can be run every time a piece of code is changed, allowing issues to be resolved as quickly as possible. It's quite common for software developers to perform unit tests before delivering software to testers for formal testing.

Integration Testing

Integration testing allows individuals the opportunity to combine all of the units within a program and test them as a group. This testing level is designed to **find interface defects between the modules/functions**. This is particularly beneficial because it determines how efficiently the units are running together. Keep in mind that no matter how efficiently each unit is running, if they aren't properly integrated, it will affect the functionality of the software program. In order to run these types of tests, individuals can make use of various testing methods, but the specific method that will be used to get the job done will depend greatly on the way in which the units are defined.

System Testing

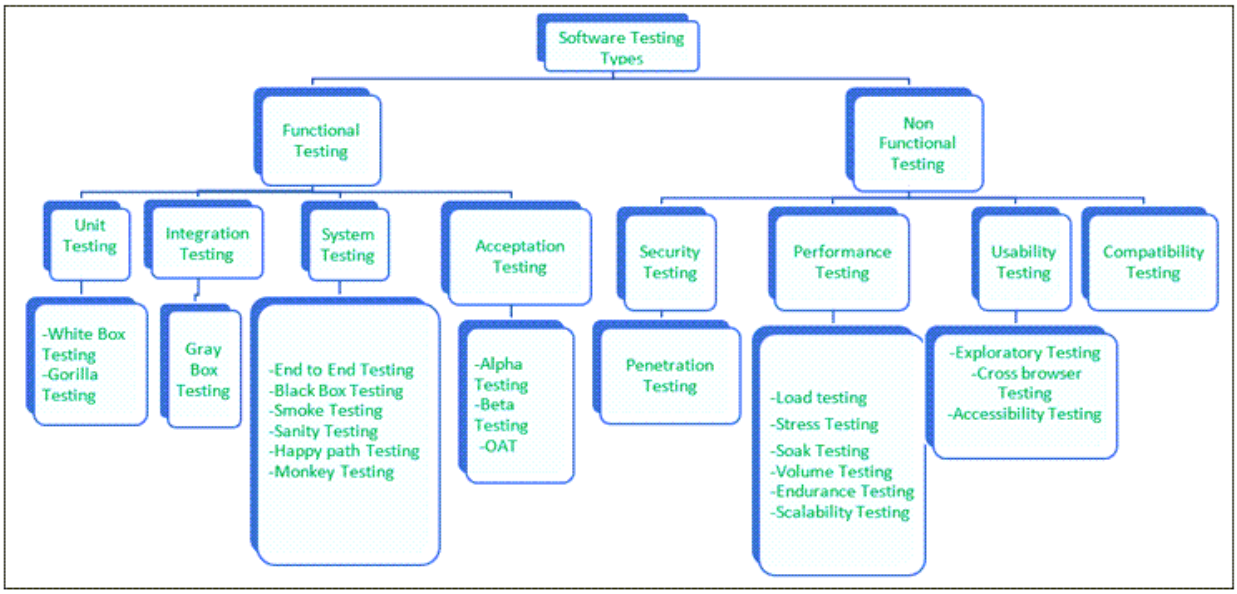
System testing is the first level in which **the complete application is tested as a whole**. The goal at this level is to evaluate whether the system has complied with all of the outlined requirements and to see that it meets Quality Standards. System testing is undertaken by independent testers who haven't played a role in developing the program. This testing is performed in an environment that closely mirrors production. System Testing is very important because it verifies that the application meets the technical, functional, and business requirements that were set by the customer.

Acceptance Testing

The final level, Acceptance testing (or User Acceptance Testing), is conducted to **determine whether the system is ready for release**. During the Software development life cycle, requirements changes can sometimes be misinterpreted in a fashion that does not meet the intended needs of the users. During this final phase, the user will test the system to find out whether the application meets their business' needs. Once this process has been completed and the software has passed, the program will then be delivered to production.

Here is the high-level classification of Software testing types.

We will see each type of testing in detail with examples.



Quality Control: Quality Control involves a series of inspections, reviews, and tests used throughout the software process to ensure each work product meets the requirements place upon it. Quality control includes a feedback loop to the process that created the work product.

Quality Assurance	Quality Control
Quality Assurance (QA) is the set of actions including facilitation, training, measurement, and analysis needed to provide adequate confidence that processes are established and continuously improved to produce products or services that conform to specifications and are fit for use.	**Quality Control (QC)** is described as the processes and methods used to compare product quality to requirements and applicable standards, and the actions are taken when a nonconformance is detected.
QA is an activity that establishes and calculates the processes that produce the product. If there is no process, there is no role for QA.	**QC** is an activity that demonstrates whether or not the product produced met standards.
QA helps establish process	**QC** relates to a particular product or service
QA sets up a measurement program to evaluate processes	**QC** verified whether particular attributes exist, or do not exist, in a explicit product or service.
QA identifies weakness in processes and improves them	**QC** identifies defects for the primary goals of correcting errors.
Quality Assurance is a managerial tool.	Quality Control is a corrective tool.
Verification is an example of QA.	Validation is an example of QC.