

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

This tutorial will give you a basic understanding on software testing, its types, methods, levels, and other related terminologies.

Why to Learn Software Testing?

In the IT industry, large companies have a team with responsibilities to evaluate the developed software in context of the given requirements. Moreover, developers also conduct testing which is called **Unit Testing**. In most cases, the following professionals are involved in testing a system within their respective capacities –

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

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

Applications of Software Testing

- **Cost Effective Development** - Early testing saves both time and cost in many aspects, however reducing the cost without testing may result in improper design of a software application rendering the product useless.
- **Product Improvement** - During the SDLC phases, testing is never a time-consuming process. However diagnosing and fixing the errors identified during proper testing is a time-consuming but productive activity.
- **Test Automation** - Test Automation reduces the testing time, but it is not possible to start test automation at any time during software development. Test automation 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.
- **Quality Check** - Software testing helps in determining following set of properties of any software such as
 - Functionality
 - Reliability
 - Usability
 - Efficiency
 - Maintainability
 - Portability

Testing, Quality Assurance, and Quality Control

Most people get confused when it comes to pin down the differences among Quality Assurance, Quality Control, and Testing. Although they are interrelated and to some extent, they can be considered as same activities, but there exist distinguishing points that set them apart. The following table lists the points that differentiate QA, QC, and Testing.

Quality Assurance	Quality Control	Testing
QA includes activities that ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements.	It includes activities that ensure the verification of a developed software with respect to documented (or not in some cases) requirements.	It includes activities that ensure the identification of bugs/error/defects in a software.
Focuses on processes and procedures rather than conducting actual testing on the system.	Focuses on actual testing by executing the software with an aim 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.

Audit and Inspection

Audit – It is 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 involved during the testing of a software. As per IEEE, it is a review of documented processes that organizations implement and follow. Types of audit include Legal Compliance Audit, Internal Audit, and System Audit.

Inspection – It is a formal technique that involves formal or informal technical reviews of any artifact by identifying any error or gap. As per IEEE94, inspection is a formal evaluation technique in which software requirements, designs, or codes are examined in detail by a person or a group other than the author to detect faults, violations of development standards, and other problems.

Formal inspection meetings may include the following processes: Planning, Overview Preparation, Inspection Meeting, Rework, and Follow-up.

Testing and Debugging

Testing – It involves identifying bug/error/defect in a software without correcting it. Normally professionals with a quality assurance background are involved in bugs identification. Testing is performed in the testing phase.

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

Software Testing - ISO Standards

Many organizations around the globe develop and implement different standards to improve the quality needs of their software. This chapter briefly describes some of the widely used standards related to Quality Assurance and Testing.

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, which you can study when you study the standard in detail.

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 that describes the usability components and the 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 25000:2005

ISO/IEC 25000:2005 is commonly known as the standard that provides the guidelines for Software 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 clients' production process. The main contents are related to the following items –

- Set of requirements for software packages.
- Instructions for testing a delivered software package against the specified requirements.

Miscellaneous

Some of the other standards related to QA and Testing processes are mentioned below –

Sr.No Standard & Description

1 **IEEE 829**

- A standard for the format of documents used in different stages of software testing.
IEEE 1061
- 2 A methodology for establishing quality requirements, identifying, implementing, analyzing, and validating the process, and product of software quality metrics.
IEEE 1059
- 3 Guide for Software Verification and Validation Plans.
IEEE 1008
- 4 A standard for unit testing.
IEEE 1012
- 5 A standard for Software Verification and Validation.
IEEE 1028
- 6 A standard for software inspections.
IEEE 1044
- 7 A standard for the classification of software anomalies.
IEEE 1044-1
- 8 A guide for the classification of software anomalies.
IEEE 830
- 9 A guide for developing system requirements specifications.
IEEE 730
- 10 A standard for software quality assurance plans.
IEEE 1061
- 11 A standard for software quality metrics and methodology.
IEEE 12207
- 12 A standard for software life cycle processes and life cycle data.
BS 7925-1
- 13 A vocabulary of terms used in software testing.
BS 7925-2
- 14 A standard for software component testing.

Software Testing - Types of Testing

This section describes the different types of testing that may be used to test a software during SDLC.

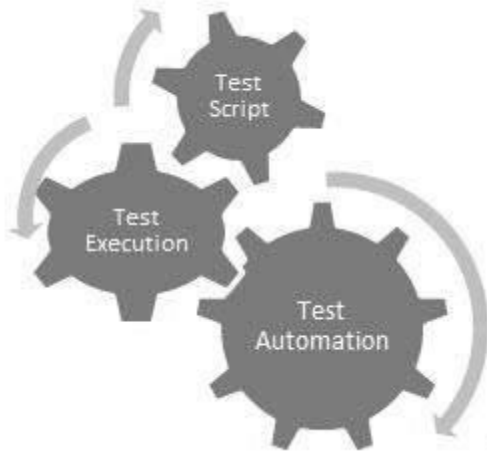
Manual Testing

Manual testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

Testers use test plans, test cases, or test scenarios to test a software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it.

Automation Testing

Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.



Apart from regression testing, automation testing is also used to test the application from load, performance, and stress point of view. It increases the test coverage, improves accuracy, and saves time and money in comparison to manual testing.

What to Automate?

It is not possible to automate everything in a software. The areas at which a user can make transactions such as the login form or registration forms, any area where large number of users can access the software simultaneously should be automated.

Furthermore, all GUI items, connections with databases, field validations, etc. can be efficiently tested by automating the manual process.

When to Automate?

Test Automation should be used by considering the following aspects of a software –

- Large and critical projects
- Projects that require testing the same areas frequently
- Requirements not changing frequently
- Accessing the application for load and performance with many virtual users
- Stable software with respect to manual testing
- Availability of time

How to Automate?

Automation is done by using a supportive computer language like VB scripting and an automated software application. There are many tools available that can be used to write automation scripts. Before mentioning the tools, let us identify the process that can be used to automate the testing process –

- Identifying areas within a software for automation
- Selection of appropriate tool for test automation
- Writing test scripts
- Development of test suits
- Execution of scripts
- Create result reports
- Identify any potential bug or performance issues

Software Testing Tools

The following tools can be used for automation testing –

- HP Quick Test Professional
- Selenium
- IBM Rational Functional Tester
- SilkTest
- TestComplete
- Testing Anywhere
- WinRunner
- LoadRunner

- Visual Studio Test Professional
- WATIR

Software Testing – Methods

There are different methods that can be used for software testing. This content briefly describes the methods available.

Black-Box Testing

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

The following table lists the advantages and disadvantages of black-box testing.

Advantages	Disadvantages
Well suited and efficient for large code segments.	Limited coverage, since only a selected number of test scenarios is actually performed.
Code access is not required.	Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
Clearly separates user's perspective from the developer's perspective through visibly defined roles.	Blind coverage, since the tester cannot target specific code segments or errorprone areas.
Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems.	The test cases are difficult to design.

White-Box Testing

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called **glass testing** or **open-box testing**. In order to perform **white-box** testing on an application, a tester needs to know the internal workings of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

The following table lists the advantages and disadvantages of white-box testing.

Advantages	Disadvantages
As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.	Due to the fact that a skilled tester is needed to perform white-box testing, the costs are increased.
It helps in optimizing the code.	Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems, as many paths will go untested.
Extra lines of code can be removed which can bring in hidden defects.	It is difficult to maintain white-box testing, as it requires specialized tools like code analyzers and debugging tools.
Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.	

Grey-Box Testing

Grey-box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase the more you know, the better carries a lot of weight while testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black-box testing, where the tester only tests the application's user interface; in grey-box testing, the tester has access to design documents and the database. Having this knowledge, a tester can prepare better test data and test scenarios while making a test plan.

Advantages	Disadvantages
Offers combined benefits of black-box and white-box testing wherever possible.	Since the access to source code is not available, the ability to go over the code and test coverage is limited.
Grey box testers don't rely on the source code; instead they rely on interface definition and functional specifications.	The tests can be redundant if the software designer has already run a test case.
Based on the limited information available, a grey-box tester can design excellent test scenarios especially around communication protocols and data type handling.	Testing every possible input stream is unrealistic because it would take an unreasonable amount of time; therefore, many program paths will go untested.
The test is done from the point of view of the user and not the designer.	

A Comparison of Testing Methods

The following table lists the points that differentiate black-box testing, grey-box testing, and white-box testing.

Black-Box Testing	Grey-Box Testing	White-Box Testing
The internal workings of an application need not be known.	The tester has limited knowledge of the internal workings of the application.	Tester has full knowledge of the internal workings of the application.
Also known as closed-box testing, data-driven testing, or functional testing.	Also known as translucent testing, as the tester has limited knowledge of the insides of the application.	Also known as clear-box testing, structural testing, or code-based testing.
Performed by end-users and also by testers and developers.	Performed by end-users and also by testers and developers.	Normally done by testers and developers.
Testing is based on external expectations - Internal behavior of the application is unknown.	Testing is done on the basis of high-level database diagrams and data flow diagrams.	Internal workings are fully known and the tester can design test data accordingly.
It is exhaustive and the least time-consuming.	Partly time-consuming and exhaustive.	The most exhaustive and time-consuming type of testing.
Not suited for algorithm testing.	Not suited for algorithm testing.	Suited for algorithm testing.
This can only be done by trial-and-error method.	Data domains and internal boundaries can be tested, if known.	Data domains and internal boundaries can be better tested.

Software Testing - Levels

There are different levels during the process of testing. In this section, a brief description is provided about these levels.

Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are –

- Functional Testing
- Non-functional Testing

Functional Testing

This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of a software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

There are five steps that are involved while testing an application for functionality.

Steps	Description
I	The determination of the functionality that the intended application is meant to perform.
II	The creation of test data based on the specifications of the application.
III	The output based on the test data and the specifications of the application.
IV	The writing of test scenarios and the execution of test cases.
V	The comparison of actual and expected results based on the executed test cases.

An effective testing practice will see the above steps applied to the testing policies of every organization and hence it will make sure that the organization maintains the strictest of standards when it comes to software quality.

Unit Testing

This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is different from the test data of the quality assurance team.

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

Limitations of Unit Testing

Testing cannot catch each and every bug in an application. It is impossible to evaluate every execution path in every software application. The same is the case with unit testing.

There is a limit to the number of scenarios and test data that a developer can use to verify a source code. After having exhausted all the options, there is no choice but to stop unit testing and merge the code segment with other units.

Integration Testing

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing.

Sr. No.	Integration Testing Method
1	Bottom-up integration This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.
2	Top-down integration In this testing, the highest-level modules are tested first and progressively, lower-level modules are tested thereafter.

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations.

System Testing

System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialized testing team.

System testing is important because of the following reasons –

- System testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.
- The application is tested thoroughly to verify that it meets the functional and technical specifications.
- The application is tested in an environment that is very close to the production environment where the application will be deployed.
- System testing enables us to test, verify, and validate both the business requirements as well as the application architecture.

Regression Testing

Whenever a change in a software application is made, it is quite possible that other areas within the application have been affected by this change. Regression testing is performed to verify that a fixed bug hasn't resulted in another functionality or business rule violation. The intent of regression testing is to ensure that a change, such as a bug fix should not result in another fault being uncovered in the application.

Regression testing is important because of the following reasons –

- Minimize the gaps in testing when an application with changes made has to be tested.
- Testing the new changes to verify that the changes made did not affect any other area of the application.
- Mitigates risks when regression testing is performed on the application.
- Test coverage is increased without compromising timelines.
- Increase speed to market the product.

Acceptance Testing

This is arguably the most important type of testing, as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirement. The QA team will have a set of pre-written scenarios and test cases that will be used to test the application.

More ideas will be shared about the application and more tests can be performed on it to gauge its accuracy and the reasons why the project was initiated. Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors, or interface gaps, but also to point out any bugs in the application that will result in system crashes or major errors in the application.

By performing acceptance tests on an application, the testing team will reduce how the application will perform in production. There are also legal and contractual requirements for acceptance of the system.

Alpha Testing

This test is the first stage of testing and will be performed amongst the teams (developer and QA teams). Unit testing, integration testing and system testing when combined together is known as alpha testing. During this phase, the following aspects will be tested in the application –

- Spelling Mistakes
- Broken Links
- Cloudy Directions
- The Application will be tested on machines with the lowest specification to test loading times and any latency problems.

Beta Testing

This test is performed after alpha testing has been successfully performed. In beta testing, a sample of the intended audience tests the application. Beta testing is also known as **pre-release testing**. Beta test versions of software are ideally distributed to a wide audience on the Web, partly to give the program a "real-world" test and partly to provide a preview of the next release. In this phase, the audience will be testing the following –

- Users will install, run the application and send their feedback to the project team.
- Typographical errors, confusing application flow, and even crashes.
- Getting the feedback, the project team can fix the problems before releasing the software to the actual users.
- The more issues you fix that solve real user problems, the higher the quality of your application will be.
- Having a higher-quality application when you release it to the general public will increase customer satisfaction.

Non-Functional Testing

This section is based upon testing an application from its non-functional attributes. Non-functional testing involves testing a software from the requirements which are nonfunctional in nature but important such as performance, security, user interface, etc.

Some of the important and commonly used non-functional testing types are discussed below.

Performance Testing

It is mostly used to identify any bottlenecks or performance issues rather than finding bugs in a software. There are different causes that contribute in lowering the performance of a software –

- Network delay
- Client-side processing
- Database transaction processing
- Load balancing between servers
- Data rendering

Performance testing is considered as one of the important and mandatory testing type in terms of the following aspects –

- Speed (i.e. Response Time, data rendering and accessing)
- Capacity
- Stability
- Scalability

Performance testing can be either qualitative or quantitative and can be divided into different sub-types such as **Load testing** and **Stress testing**.

Load Testing

It is a process of testing the behavior of a software by applying maximum load in terms of software accessing and manipulating large input data. It can be done at both normal and peak load conditions. This type of testing identifies the maximum capacity of software and its behavior at peak time.

Most of the time, load testing is performed with the help of automated tools such as Load Runner, AppLoader, IBM Rational Performance Tester, Apache JMeter, Silk Performer, Visual Studio Load Test, etc.

Virtual users (VUsers) are defined in the automated testing tool and the script is executed to verify the load testing for the software. The number of users can be increased or decreased concurrently or incrementally based upon the requirements.

Stress Testing

Stress testing includes testing the behavior of a software under abnormal conditions. For example, it may include taking away some resources or applying a load beyond the actual load limit.

The aim of stress testing is to test the software by applying the load to the system and taking over the resources used by the software to identify the breaking point. This testing can be performed by testing different scenarios such as –

- Shutdown or restart of network ports randomly
- Turning the database on or off
- Running different processes that consume resources such as CPU, memory, server, etc.

Usability Testing

Usability testing is a black-box technique and is used to identify any error(s) and improvements in the software by observing the users through their usage and operation.

According to Nielsen, usability can be defined in terms of five factors, i.e. efficiency of use, learn-ability, memory-ability, errors/safety, and satisfaction. According to him, the usability of a product will be good and the system is usable if it possesses the above factors.

Nigel Bevan and Macleod considered that usability is the quality requirement that can be measured as the outcome of interactions with a computer system. This requirement can be fulfilled and the end-user will be satisfied if the intended goals are achieved effectively with the use of proper resources.

Molich in 2000 stated that a user-friendly system should fulfill the following five goals, i.e., easy to Learn, easy to remember, efficient to use, satisfactory to use, and easy to understand.

In addition to the different definitions of usability, there are some standards and quality models and methods that define usability in the form of attributes and sub-attributes such as ISO-9126, ISO-9241-11, ISO-13407, and IEEE std.610.12, etc.

UI vs Usability Testing

UI testing involves testing the Graphical User Interface of the Software. UI testing ensures that the GUI functions according to the requirements and tested in terms of color, alignment, size, and other properties.

On the other hand, usability testing ensures a good and user-friendly GUI that can be easily handled. UI testing can be considered as a sub-part of usability testing.

Security Testing

Security testing involves testing a software in order to identify any flaws and gaps from security and vulnerability point of view. Listed below are the main aspects that security testing should ensure –

- Confidentiality
- Integrity
- Authentication
- Availability
- Authorization
- Non-repudiation
- Software is secure against known and unknown vulnerabilities
- Software data is secure
- Software is according to all security regulations
- Input checking and validation
- SQL insertion attacks
- Injection flaws
- Session management issues
- Cross-site scripting attacks
- Buffer overflows vulnerabilities
- Directory traversal attacks

Portability Testing

Portability testing includes testing a software with the aim to ensure its reusability and that it can be moved from another software as well. Following are the strategies that can be used for portability testing –

- Transferring an installed software from one computer to another.
- Building executable (.exe) to run the software on different platforms.

Portability testing can be considered as one of the sub-parts of system testing, as this testing type includes overall testing of a software with respect to its usage over different environments. Computer hardware, operating systems, and browsers are the major focus of portability testing. Some of the pre-conditions for portability testing are as follows –

- Software should be designed and coded, keeping in mind the portability requirements.
- Unit testing has been performed on the associated components.
- Integration testing has been performed.
- Test environment has been established.

Software Engineering | Verification and Validation

Verification and Validation is the process of investigating that a software system satisfies specifications and standards and it fulfills the required purpose. **Barry Boehm** described verification and validation as the following:

Verification: Are we building the product right?

Validation: Are we building the right product?

Verification:

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have.

Verification is **Static Testing**.

Activities involved in verification:

1. Inspections
2. Reviews
3. Walkthroughs
4. Desk-checking

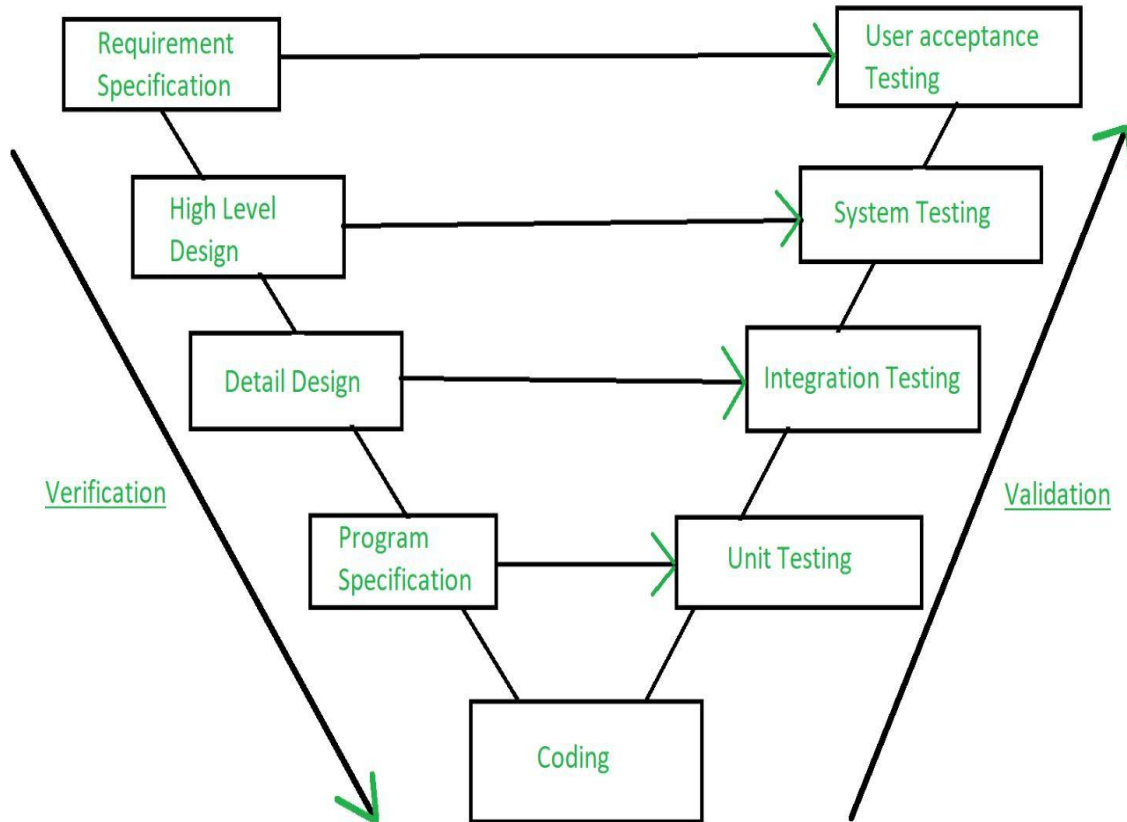
Validation:

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product.

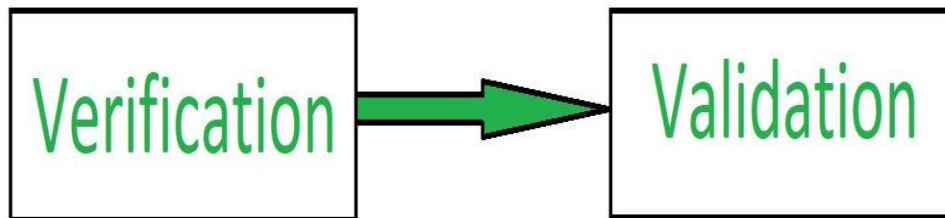
Validation is the **Dynamic Testing**.

Activities involved in validation:

1. Black box testing
2. White box testing
3. Unit testing
4. Integration testing



Note: Verification is followed by Validation.



Quality Assurance & Quality Control

Quality assurance (QA) and quality control (QC) are two terms that are often used interchangeably. Although similar, there are distinct differences between the two concepts. This content will explain the differences between quality control and quality management, and provide definitions and examples of each.

Differences Between QA and QC

Quality assurance and quality control are two aspects of quality management. While some quality assurance and quality control activities are interrelated, the two are defined differently. Typically, QA activities and responsibilities cover virtually all of the quality system in one fashion or another, while QC is a subset of the QA activities. Also, elements in the quality system might not be specifically covered by QA/QC activities and responsibilities but may involve QA and QC. Below are [ISO 9000](#) definitions from [ISO 9000:2015](#):



Quality Assurance

Quality assurance can be defined as "part of *quality management* focused on providing confidence that *quality requirements* will be fulfilled." The confidence provided by quality assurance is twofold—internally to management and externally to customers, government agencies, regulators, certifiers, and third parties. An alternate definition is "all the planned and systematic activities implemented within the quality system that can be demonstrated to provide confidence that a product or service will fulfill requirements for quality."

Quality Control

Quality control can be defined as "part of *quality management* focused on fulfilling *quality requirements*." While quality assurance relates to how a process is performed or how a product is made, quality control is more the inspection aspect of quality management. An alternate definition is "the operational techniques and activities used to fulfill requirements for quality."

Industry Perspectives on QA and QC

For some service organizations, the concept of quality control may be foreign because there is no tangible product to inspect and control. The quality assurance function in a service organization may not include quality control of the service but may include quality control of any products involved in providing the service.

A service may include products that are documents (such as a report, contract, or design) or tangible products (such as a rental car or units of blood). It may be necessary to control product quality in a service organization to ensure that the service meets customer requirements.

QA, QC, and Inspection

Inspection is the process of measuring, examining, and testing to gauge one or more characteristics of a product or service and the comparison of these with specified requirements to determine conformity. Products, processes, and various other results can be inspected to make sure that the object coming off a production line, or the service being provided, is correct and meets specifications.

Quality Assurance and Audit Functions

Auditing is part of the quality assurance function. It is important to ensure quality because it is used to compare actual conditions with requirements and to report those results to management.

In *The Quality Audit: A Management Evaluation Tool* (McGraw-Hill, 1988), Charles Mill wrote that auditing and inspection are not interchangeable: “The auditor may use inspection techniques as an evaluation tool, but the audit should not be involved in carrying out any verification activities leading to the actual acceptance or rejection of a product or service. An audit should be involved with the evaluation of the process and controls covering the production and verification activities.”

Formal management systems have evolved to direct and control organizations. There are quality management systems (QMSs) as well as environmental or other management systems, and each of these systems may be audited.

History of QA and QC

Quality has been defined as fitness for use, conformance to requirements, and the pursuit of excellence. Even though the concept of quality has existed from early times, the study and definition of quality have been given prominence only in the last century.

1920s: Quality Control

Following the Industrial Revolution and the rise of mass production, it became important to better define and control the quality of products. Originally, the goal of quality was to ensure that engineering requirements were met in final products. Later, as manufacturing processes became more complex, quality developed into a discipline for controlling process variation as a means of producing quality products.

1950s: Quality Assurance and Auditing

The quality profession expanded to include the quality assurance and quality audit functions. The drivers of independent verification of quality were primarily industries in which public health and safety were paramount.

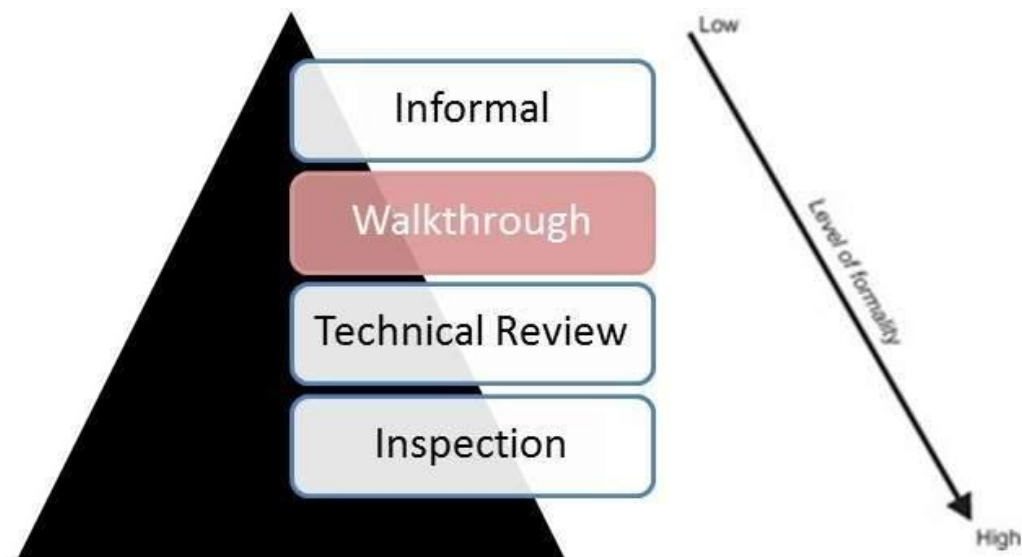
Code Walkthrough

What is Code Walkthrough?

Code Walkthrough is a form of peer review in which a programmer leads the review process and the other team members ask questions and spot possible errors against development standards and other issues.

- The meeting is usually led by the author of the document under review and attended by other members of the team.
- Review sessions may be formal or informal.
- Before the walkthrough meeting, the preparation by reviewers and then a review report with a list of findings.
- The scribe, who is not the author, marks the minutes of meeting and note down all the defects/issues so that it can be tracked to closure.
- The main purpose of walkthrough is to enable learning about the content of the document under review to help team members gain an understanding of the content of the document and also to find defects.

Where Code Walkthrough fits in ?



Software Maintenance

Software maintenance is widely accepted part of SDLC now a days. It stands for all the modifications and updations done after the delivery of software product. There are number of reasons, why modifications are required, some of them are briefly mentioned below:

- **Market Conditions** - Policies, which changes over the time, such as taxation and newly introduced constraints like, how to maintain bookkeeping, may trigger need for modification.
- **Client Requirements** - Over the time, customer may ask for new features or functions in the software.
- **Host Modifications** - If any of the hardware and/or platform (such as operating system) of the target host changes, software changes are needed to keep adaptability.
- **Organization Changes** - If there is any business level change at client end, such as reduction of organization strength, acquiring another company, organization venturing into new business, need to modify in the original software may arise.

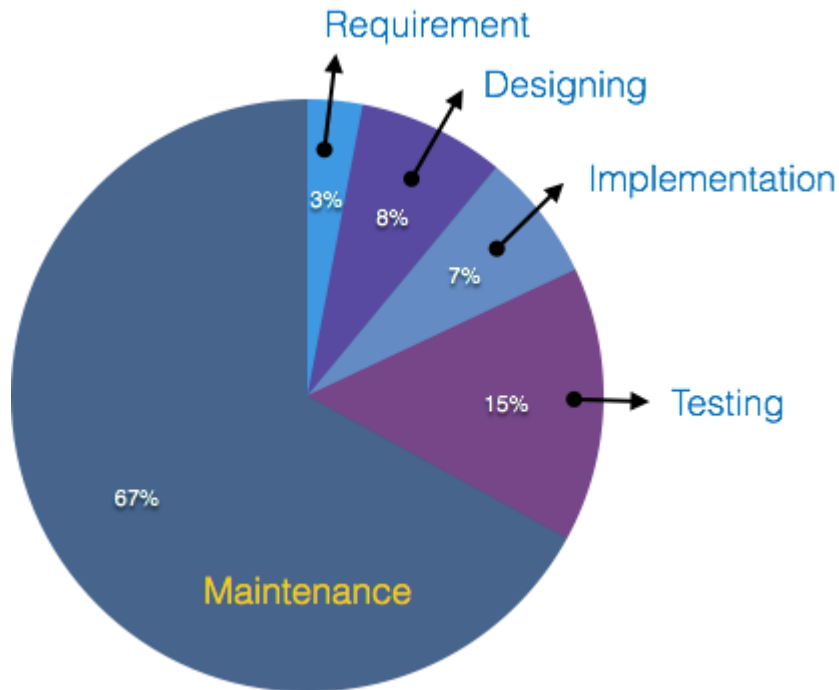
Types of maintenance

In a software lifetime, type of maintenance may vary based on its nature. It may be just a routine maintenance tasks as some bug discovered by some user or it may be a large event in itself based on maintenance size or nature. Following are some types of maintenance based on their characteristics:

- **Corrective Maintenance** - This includes modifications and updations done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.
- **Adaptive Maintenance** - This includes modifications and updations applied to keep the software product up-to date and tuned to the ever changing world of technology and business environment.
- **Perfective Maintenance** - This includes modifications and updates done in order to keep the software usable over long period of time. It includes new features, new user requirements for refining the software and improve its reliability and performance.
- **Preventive Maintenance** - This includes modifications and updations to prevent future problems of the software. It aims to attend problems, which are not significant at this moment but may cause serious issues in future.

Cost of Maintenance

Reports suggest that the cost of maintenance is high. A study on estimating software maintenance found that the cost of maintenance is as high as 67% of the cost of entire software process cycle.



On an average, the cost of software maintenance is more than 50% of all SDLC phases. There are various factors, which trigger maintenance cost go high, such as:

Real-world factors affecting Maintenance Cost

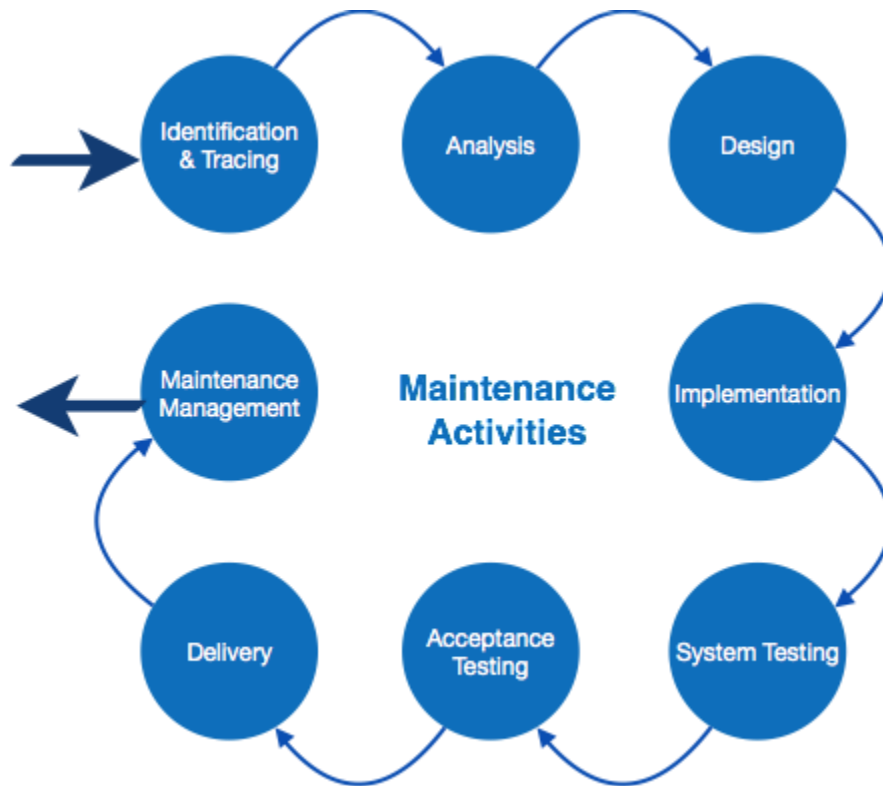
- The standard age of any software is considered up to 10 to 15 years.
- Older softwares, which were meant to work on slow machines with less memory and storage capacity cannot keep themselves challenging against newly coming enhanced softwares on modern hardware.
- As technology advances, it becomes costly to maintain old software.
- Most maintenance engineers are newbie and use trial and error method to rectify problem.
- Often, changes made can easily hurt the original structure of the software, making it hard for any subsequent changes.
- Changes are often left undocumented which may cause more conflicts in future.

Software-end factors affecting Maintenance Cost

- Structure of Software Program
- Programming Language
- Dependence on external environment
- Staff reliability and availability

Maintenance Activities

IEEE provides a framework for sequential maintenance process activities. It can be used in iterative manner and can be extended so that customized items and processes can be included.



These activities go hand-in-hand with each of the following phase:

- **Identification & Tracing** - It involves activities pertaining to identification of requirement of modification or maintenance. It is generated by user or system may itself report via logs or error messages. Here, the maintenance type is classified also.
- **Analysis** - The modification is analyzed for its impact on the system including safety and security implications. If probable impact is severe, alternative solution is looked for. A set of required modifications is then materialized into requirement specifications. The cost of modification/maintenance is analyzed and estimation is concluded.
- **Design** - New modules, which need to be replaced or modified, are designed against requirement specifications set in the previous stage. Test cases are created for validation and verification.
- **Implementation** - The new modules are coded with the help of structured design created in the design step. Every programmer is expected to do unit testing in parallel.
- **System Testing** - Integration testing is done among newly created modules. Integration testing is also carried out between new modules and the system. Finally the system is tested as a whole, following regressive testing procedures.
- **Acceptance Testing** - After testing the system internally, it is tested for acceptance with the help of users. If at this state, user complaints some issues they are addressed or noted to address in next iteration.
- **Delivery** - After acceptance test, the system is deployed all over the organization either by small update package or fresh installation of the system. The final testing takes place at client end after the software is delivered.

Training facility is provided if required, in addition to the hard copy of user manual.

- **Maintenance management** - Configuration management is an essential part of system maintenance. It is aided with version control tools to control versions, semi-version or patch management.

System Security and Audit

System Audit

It is an investigation to review the performance of an operational system. The objectives of conducting a system audit are as follows –

- To compare actual and planned performance.
- To verify that the stated objectives of system are still valid in current environment.
- To evaluate the achievement of stated objectives.
- To ensure the reliability of computer based financial and other information.
- To ensure all records included while processing.
- To ensure protection from frauds.

Audit of Computer System Usage

Data processing auditors audits the usage of computer system in order to control it. The auditor need control data which is obtained by computer system itself.

The System Auditor

The role of auditor begins at the initial stage of system development so that resulting system is secure. It describes an idea of utilization of system that can be recorded which helps in load planning and deciding on hardware and software specifications. It gives an indication of wise use of the computer system and possible misuse of the system.

Audit Trial

An audit trial or audit log is a security record which is comprised of who has accessed a computer system and what operations are performed during a given period of time. Audit trials are used to do detailed tracing of how data on the system has changed.

It provides documentary evidence of various control techniques that a transaction is subject to during its processing. Audit trails do not exist independently. They are carried out as a part of accounting for recovering lost transactions.

Audit Methods

Auditing can be done in two different ways –

Auditing around the Computer

- Take sample inputs and manually apply processing rules.
- Compare outputs with computer outputs.

Auditing through the Computer

- Establish audit trail which allows examining selected intermediate results.
- Control totals provide intermediate checks.

Audit Considerations

Audit considerations examine the results of the analysis by using both the narratives and models to identify the problems caused due to misplaced functions, split processes or functions, broken data flows, missing data, redundant or incomplete processing, and nonaddressed automation opportunities.

The activities under this phase are as follows –

- Identification of the current environment problems
- Identification of problem causes
- Identification of alternative solutions
- Evaluation and feasibility analysis of each solution
- Selection and recommendation of most practical and appropriate solution
- Project cost estimation and cost benefit analysis

Security

System security refers to protecting the system from theft, unauthorized access and modifications, and accidental or unintentional damage. In computerized systems, security involves protecting all the parts of computer system which includes data, software, and hardware. Systems security includes system privacy and system integrity.

- **System privacy** deals with protecting individuals systems from being accessed and used without the permission/knowledge of the concerned individuals.
- **System integrity** is concerned with the quality and reliability of raw as well as processed data in the system.