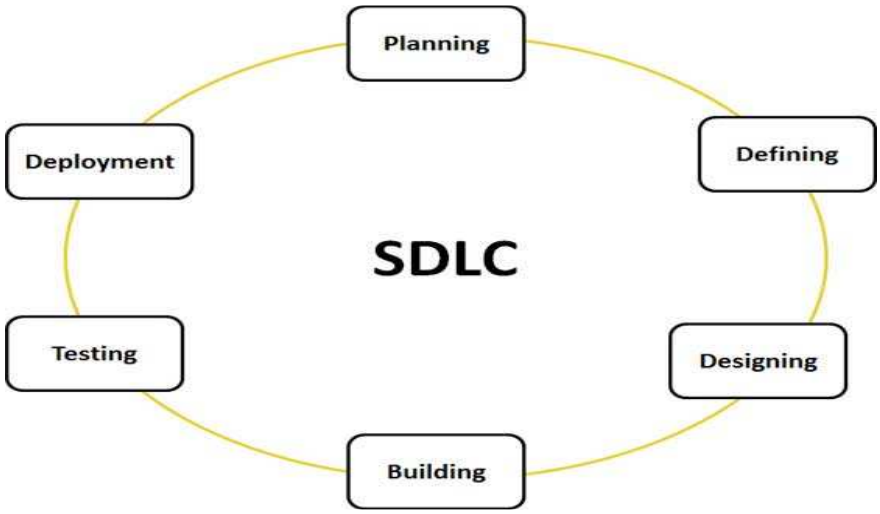
Software Testing Concept

Software Development Life Cycle (SDLC) :

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.



A typical Software Development life cycle consists of the following stages:

Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational, and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through .SRS. . Software Requirement Specification document which consists of all the product requirements to be designed and developed during the project life cycle.

Stage 3: Designing the product architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity , budget and time constraints , the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java, and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However this stage refers to the testing only stage of the product where products defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometime product deployment happens in stages as per the organizations. business strategy. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

SDLC Models

There are various software development life cycle models defined and designed which are followed during software development process. These models are also referred as "Software Development Process Models".

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Model

Software Testing Life Cycle (STLC)

Software Testing Life Cycle refers to a testing process which has specific steps to be executed in a definite sequence to ensure that the quality goals have been met. In STLC process, each activity is carried out in a planned and systematic way. Each phase has different goals and deliverables. Different organizations have different phases in STLC; however the basis remains the same.

Below are the phases of STLC:

1. Requirements phase
2. Planning Phase
3. Analysis phase
4. Design Phase
5. Implementation Phase
6. Execution Phase
7. Conclusion Phase
8. Closure Phase

1. Requirement Phase:

During this phase of STLC, analyze and study the requirements. Have brain storming sessions with other teams and try to find out whether the requirements are testable or not. This phase helps to identify the scope of the testing. If any feature is not testable, communicate it during this phase so that the mitigation strategy can be planned.

2. Planning Phase:

In practical scenarios, Test planning is the first step of the testing process. In this phase we identify the activities and resources which would help to meet the testing objectives. During planning we also try to identify the metrics, the method of gathering and tracking those metrics.

On what basis the planning is done? Only requirements?
The answer is NO. Requirements do form one of the bases but there are 2 other very important factors which influence test planning. These are:
– Test strategy of the organization.
– Risk analysis / Risk Management and mitigation.

3. Analysis Phase:

This STLC phase defines “WHAT” to be tested. We basically identify the test conditions through the requirements document, product risks and other test basis. The test condition should be traceable back to the requirement. There are various factors which effect the identification of test conditions:

– Levels and depth of testing
– Complexity of the product
– Product and project risks
– Software development life cycle involved.
– Test management
– Skills and knowledge of the team.
– Availability of the stakeholders.

We should try to write down the test conditions in a detailed way. For example, for an e-commerce web application, you can have a test condition as “User should be able to make a payment”. Or you can detail it out by saying “User should be able to make payment through NEFT, debit card and

credit card". The most important advantage of writing the detailed test condition is that it increases the test coverage, since the test cases will be written on the basis of the test condition, these details will trigger to write more detailed test cases which will eventually increase the coverage. Also identify the exit criteria of the testing, i.e determine some conditions when you will stop the testing.

4. Design Phase:

This phase defines "HOW" to test. This phase involves the following tasks:
– Detail the test condition. Break down the test conditions into multiple sub conditions to increase coverage.
– Identify and get the test data
– Identify and set up the test environment.
– Create the requirement traceability metrics
– Create the test coverage metrics.

5. Implementation Phase:

The major task in this STLC phase is of creation of the detailed test cases. Prioritize the test cases also identify which test case will become part of the regression suite. Before finalizing the test case, It is important to carry out the review to ensure the correctness of the test cases. Also don't forget to take the sign off of the test cases before actual execution starts. If your project involves automation, identify the candidate test cases for automation and proceed for scripting the test cases. Don't forget to review them!

6. Execution Phase:

As the name suggests, this is the Software Testing Life Cycle phase where the actual execution takes place. But before you start your execution, make sure that your entry criterion is met. Execute the test cases, log defects in case of any discrepancy. Simultaneously fill your traceability metrics to track your progress.

7. Conclusion Phase:

This STLC phase concentrates on the exit criteria and reporting. Depending on your project and stakeholders choice, you can decide on reporting whether you want to send out a daily report of weekly report etc. There are different types of reports (DSR – Daily status report, WSR – Weekly status reports) which you can send, but the important point is, the content of the report changes and depends upon whom you are sending your reports. If Project managers belong to testing background then they are more interested in the technical aspect of the project, so include the technical things in your report
(number of test cases passed, failed, defects raised, severity 1 defects etc.). But if you are reporting to upper stakeholders, they might not be interested in the technical things so report them about the risks that have been mitigated through the testing.

8. Closure Phase:

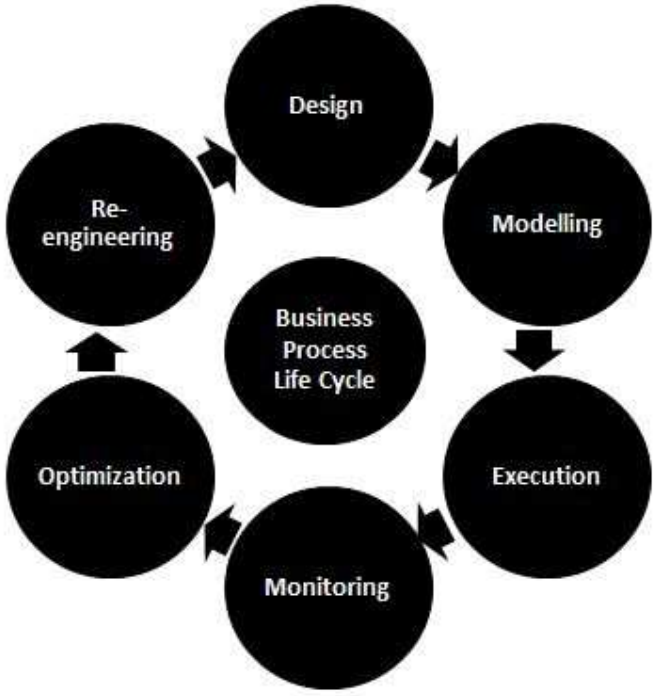
Tasks for the closure activities include the following:
– Check for the completion of the test. Whether all the test cases are executed or mitigated deliberately. Check there are no severity 1 defects opened.
– Do lessons learnt meeting and create lessons learnt document.(Include what went well, where are the scope of improvements and what can be improved)

What is a Business Process?

A business process is a collection of activities or tasks that produce a specific service or product for end users. It is usually represented as a flowchart as a sequence of activities that points to a Process Matrix.

Business process Modelling is carried out by Process owners or product owners to enable the test team to test efficiently. It aims to improve business performance by optimizing the efficiency of the associated activities of a product or service.

Business Process Life Cycle:



Business Process Testing [BPT]:

It is a tool used for an automated and manual testing for designing tests, maintaining tests and executing tests. The reusable tests are usually designed by Business Analysts for improving test efficiency.

Benefits and Features of BPT :

- Allows non-technical subject matter expertise to quickly build a reusable test workflow.
- It reduces the effort required for test maintenance.
- It converts the manual tests to manual test components.
- It provides a framework to build User Acceptance Tests to meet the requirements.

SDLC - Waterfall Model

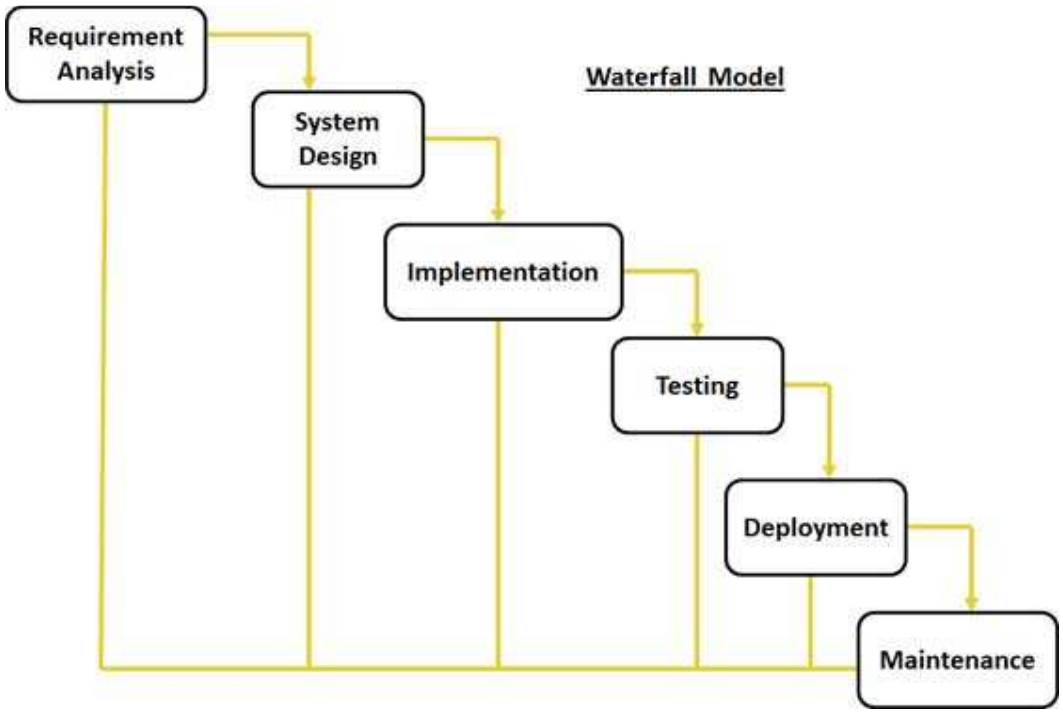
The Waterfall Model is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

Waterfall model is the earliest SDLC approach that was used for software development.

Waterfall Model design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Following is a diagrammatic representation of different phases of waterfall model.



The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Waterfall Model Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model Pros & Cons :

Advantage

The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Disadvantage

The disadvantage of waterfall development is that it does not allow for much reflection or revision.

Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The following table lists out the pros and cons of Waterfall model:

Pros	Cons
• Simple and easy to understand and use • Easy to manage due to the rigidity of the model . each phase has specific deliverables and a review process. • Phases are processed and completed one at a time. • Works well for smaller projects where requirements are very well understood. • Clearly defined stages. • Well understood milestones. • Easy to arrange tasks. • Process and results are well documented.	• No working software is produced until late during the life cycle. • High amounts of risk and uncertainty. • Not a good model for complex and object-oriented projects. • Poor model for long and ongoing projects. • Not suitable for the projects where requirements are at a moderate to high risk of changing. So risk and uncertainty is high with this process model. • It is difficult to measure progress within stages. • Cannot accommodate changing requirements. • No working software is produced until late in the life cycle. • Adjusting scope during the life cycle can end a project. • Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

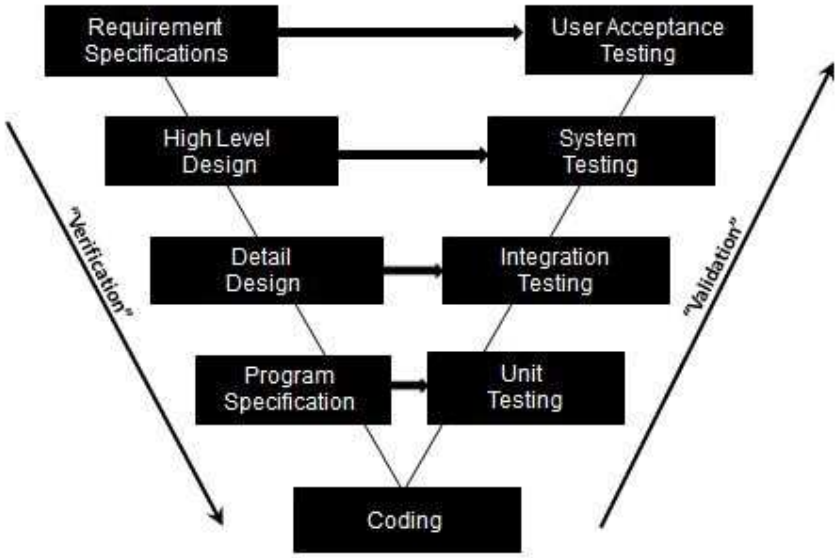
SDLC - V-Model

V model, a software development life cycle methodology, describes the activities to be performed and the results that have to be produced during the life cycle of the product. It is known as verification and validation model Validation answers the question – "Are we developing the product which attempts all that user needs from this software?" and Verification answers the question– "Are we developing this product by firmly following all design specifications ?"

V-Model Objectives:

- Project Risks Minimization
- Guaranteed Quality
- Total Cost reduction of the Entire Project
- Improved Communication between all Parties Involved

V-Model Different Phases:



- **The Requirements phase :** a document describing what the software is required to do after the software is gathered and analyzed and the corresponding test activity is **user acceptance testing**.
- **The Architectural Design phase**: where a software architecture is designed and building the components within the software and the establishing the relationships between the components and the corresponding test activity is System Testing.
- **The High Level Design phase:** breaking the system into subsystems with identified interfaces; then gets translated to a more detailed design and the corresponding test activity is Integration testing.
- **The Detailed Design phase:** where the detailed implementation of each component is specified. The detailed design broken into Data structures, Algorithm used and the corresponding test activity is unit Testing.
- **Coding :** in which each component of the software is coded and tested to verify if faithfully implements the detailed design.

Advantages and Limitations of V-Model:

ADVANTAGES:

- Emphasize for verification and validation of the product in early stages of product development.
- Each stage is testable
- Project management can track progress by milestones
- Easy to understand implement and use

LIMITATIONS:

- Does not easily handle events concurrently.
- Does not handle iterations or phases
- Does not easily handle dynamic changes in requirements
- Does not contain risk analysis or Mitigation activities.

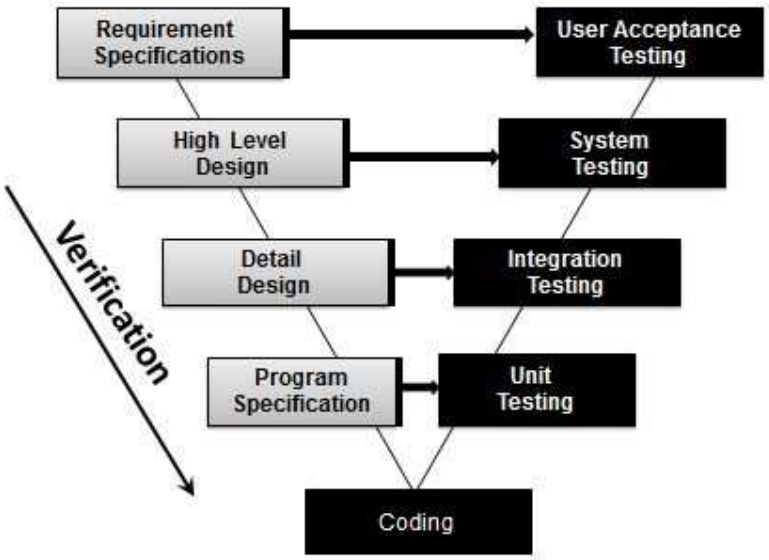
Software Testing – Verification Testing

Verification is the process of evaluating work-products of a development phase to determine whether they meet the specified requirements.

Verification ensures that the product is built according to the requirements and design specifications. It also answers to the question, Are we building the product right?

Verification Testing - Workflow:

Verification testing can be best demonstrated using V-Model. The artefacts such as test Plans, requirement specification, design, code and test cases are evaluated.



Activities:

- Reviews
- Walkthroughs
- Inspection

Software Testing - Validation Testing

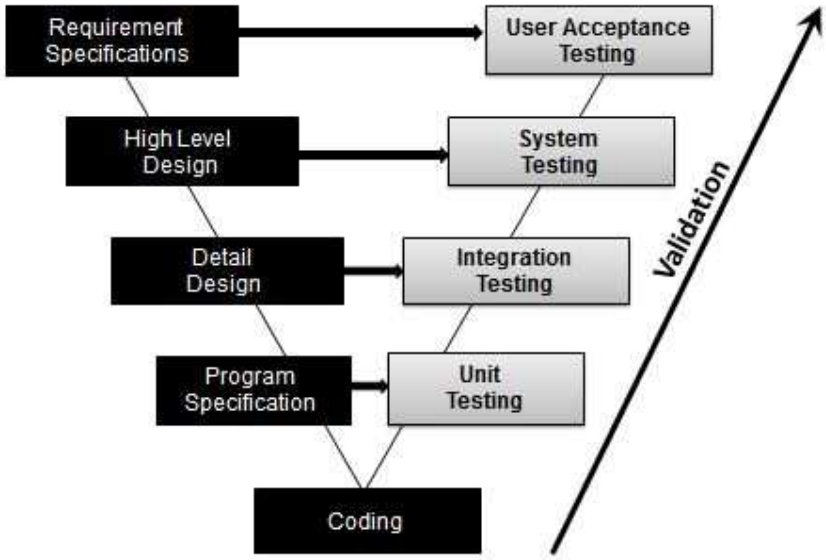
The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.

Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

It answers to the question, Are we building the right product?

Validation Testing - Workflow:

Validation testing can be best demonstrated using V-Model. The Software/product under test is evaluated during this type of testing.



Activities:

- Unit Testing
- Integration Testing
- System Testing
- User Acceptance Testing

What is a Requirement?

The requirements are the high-level descriptions about a particular system services, constraints or to a detailed specification that are generated during the requirements gathering process.

Requirement Types:

- **User Requirements -** It is a detailed description in natural language along with diagrams of the services the system provides and its operational constraints. It is usually developed by end users.
- **System requirements -** It is a structured document detailing the descriptions of the system's functions, services and operational constraints.
- **Functional Requirements -** It describes the services of the system, how the system should react to particular inputs and how the system should behave in definite situations.
- **Non-functional Requirements -** It describes the attributes of the system.
- **Domain Requirements -** Requirements that arises from the domain of the application and that reflect characteristics of that domain. It can be either functional or non-functional specifications.

Requirement Document Structure:

- Preface
- Introduction
- User requirements definition
- System architecture
- System requirements specification
- System models
- Appendix

What is Software Requirement Specification - [SRS]?

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase.

Qualities of SRS:

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable
- Modifiable
- Traceabl

Types of Requirements:

The below diagram depicts the various types of requirements that are captured during SRS.

What is Business Requirements?

Business requirements is a phase in Software development life cycle which felicitates the requirements of the end users as the very first task in order to guide the design of the future system. Business requirements are usually captured by business analysts or product owners who analyze business activities who in turn act as subject matter expertise (SME's).

Contents of Business Requirements:

- Purpose, In-scope, Out of Scope, Targeted Audiences
- Use Case diagrams
- Data Requirements
- Non Functional Requirements
- Interface Requirements
- Limitations
- Risks
- Assumptions
- Reporting Requirements
- Checklists

What is a Functional Requirement?

A functional requirement document defines the functionality of a system or one of its subsystems. It also depends upon the type of software, expected users and the type of system where the software is used.

Functional user requirements may be high-level statements of what the system should do but functional system requirements should also describe clearly about the system services in detail.

Functional Requirement Specifications document:

- Purpose of the Document
- Scope
- Business Processes
- Functional Requirements
- Data and Integration
- Security Requirements
- Performance
- Data Migration & Conversion

What do you meant by Review?

A review is a systematic examination of a document by one or more people with the main aim of finding and removing errors early in the software development life cycle. Reviews are used to verify documents such as requirements, system designs, code, test plans and test cases.

Reviews are usually performed manually while static analysis of the tools is performed using tools.

Importance of Review Process:

- Productivity of Dev team is improved and timescales reduced because the correction of defects in early stages and work-products will help to ensure that those work-products are clear and unambiguous.
- Testing costs and time is reduced as there is enough time spent during the initial phase.
- Reduction in costs because fewer defects in the final software.

Types of Defects during Review Process:

- Deviations from standards either internally defined or defined by regulatory or a trade organisation.
- Requirements defects.
- Design defects.
- Incorrect interface specifications.

What is a Test Plan?

Test planning, the most important activity to ensure that there is initially a list of tasks and milestones in a baseline plan to track the progress of the project. It also defines the size of the test effort.

It is the main document often called as master test plan or a project test plan and usually developed during the early phase of the project.

Test Plan Identifiers:

S.No.	Parameter	Description
1.	Test plan identifier	Unique identifying reference.
2.	Introduction	A brief introduction about the project and to the document.
3.	Test items	A test item is a software item that is the application under test.
4.	Features to be tested	A feature that needs to tested on the testware.
5.	Features not to be tested	Identify the features and the reasons for not including as part of testing.
6.	Approach	Details about the overall approach to testing.
7.	Item pass/fail criteria	Documented whether a software item has passed or failed its test.
8.	Test deliverables	The deliverables that are delivered as part of the testing process,such as test plans, test specifications and test summary reports.
9.	Testing tasks	All tasks for planning and executing the testing.
10.	Environmental needs	Defining the environmental requirements such as hardware, software, OS, network configurations, tools required.
11.	Responsibilities	Lists the roles and responsibilities of the team members.
12.	Staffing and training needs	Captures the actual staffing requirements and any specific skills and training requirements.
13.	Schedule	States the important project delivery dates and key milestones.
14.	Risks and Mitigation	High-level project risks and assumptions and a mitigating plan for each identified risk.
15.	Approvals	Captures all approvers of the document, their titles and the sign off date.

Test Planning Activities:

- To determine the scope and the risks that need to be tested and that are NOT to be tested.
- Documenting Test Strategy.
- Making sure that the testing activities have been included.
- Deciding Entry and Exit criteria.
- Evaluating the test estimate.
- Planning when and how to test and deciding how the test results will be evaluated, and defining test exit criterion.
- The Test artefacts delivered as part of test execution.
- Defining the management information, including the metrics required and defect resolution and risk issues.
- Ensuring that the test documentation generates repeatable test asset

Test Approach:

A test approach is the test strategy implementation of a project, defines how testing would be carried out. Test approach has two techniques:

- **Proactive -** An approach in which the test design process is initiated as early as possible in order to find and fix the defects before the build is created.
- **Reactive -** An approach in which the testing is not started until after design and coding are completed.

Different Test approaches:

- Dynamic and heuristic approaches
- Consultative approaches
- Model-based approach that uses statistical information about failure rates.
- Approaches based on risk-based testing where the entire development takes place based on the risk
- Methodical approach, which is based on failures.
- Standard-compliant approach specified by industry-specific standards.

Factors to be considered:

- Risks of product or risk of failure or the environment and the company.
- Expertise and experience of the people in the proposed tools and techniques.
- Regulatory and legal aspects, such as external and internal regulations of the development process.
- The nature of the product and the domain.

What is Test Bed?

The test execution environment configured for testing. Test bed consists of specific hardware, software, Operating system, network configuration, the product under test, other system software and application software.

Test Bed Configuration:
It is the combination of hardware and software environment on which the tests will be executed. It includes hardware configuration, operating system settings, software configuration, test terminals and other support to perform the test.

Example:
A typical test bed for a web-based application is given below:

```
Web Server - IIS/Apache
Database - MS SQL
OS - Windows/ Linux
Browser - IE/FireFox
Java version : version 6
```

What is an Entry Criterion?

Entry criterion is used to determine when a given test activity should start. It also includes the beginning of a level of testing, when test design or when test execution is ready to start.

Examples for Entry Criterion:

- Verify if the Test environment is available and ready for use.
- Verify if test tools installed in the environment are ready for use.
- Verify if Testable code is available.
- Verify if Test Data is available and validated for correctness of Data.

What is an Exit Criterion?

Exit criterion is used to determine whether a given test activity has been completed or NOT. Exit criteria can be defined for all of the test activities right from planning, specification and execution. Exit criterion should be part of test plan and decided in the planning stage.

Examples of Exit Criteria:

- Verify if All tests planned have been run.
- Verify if the level of requirement coverage has been met.
- Verify if there are NO Critical or high severity defects that are left outstanding.
- Verify if all high risk areas are completely tested.
- Verify if software development activities are completed within the projected cost.
- Verify if software development activities are completed within the projected timelines.

What is Adhoc Testing?

When a software testing performed without proper planning and documentation, it is said to be Adhoc Testing. Such kind of tests are executed only once unless we uncover the defects.

Adhoc Tests are done after formal testing is performed on the application. Adhoc methods are the least formal type of testing as it is NOT a structured approach. Hence, defects found using this method are hard to replicate as there are no test cases aligned for those scenarios.

Testing is carried out with the knowledge of the tester about the application and the tester tests randomly without following the specifications/requirements. Hence the success of Adhoc testing depends upon the capability of the tester, who carries out the test. The tester has to find defects without any proper planning and documentation, solely based on tester's intuition.

When to Execute Adhoc Testing ?

Adhoc testing can be performed when there is limited time to do exhaustive testing and usually performed after the formal test execution. Adhoc testing will be effective only if the tester has in-depth understanding about the System Under Test.

Forms of Adhoc Testing :

1. **Buddy Testing:** Two buddies, one from development team and one from test team mutually work on identifying defects in the same module. Buddy testing helps the testers develop better test cases while development team can also make design changes early. This kind of testing happens usually after completing the unit testing.
2. **Pair Testing:** Two testers are assigned the same modules and they share ideas and work on the same systems to find defects. One tester executes the tests while another tester records the notes on their findings.
3. **Monkey Testing:** Testing is performed randomly without any test cases in order to break the system.

Various ways to make Adhoc Testing More Effective

1. **Preparation:** By getting the defect details of a similar application, the probability of finding defects in the application is more.
2. **Creating a Rough Idea:** By creating a rough idea in place the tester will have a focussed approach. It is NOT required to document a detailed plan as what to test and how to test.
3. **Divide and Rule:** By testing the application part by part, we will have a better focus and better understanding of the problems if any.
4. **Targeting Critical Functionalities:** A tester should target those areas that are NOT covered while designing test cases.
5. **Using Tools:** Defects can also be brought to the lime light by using profilers, debuggers and even task monitors. Hence being proficient in using these tools one can uncover several defects.
6. **Documenting the findings:** Though testing is performed randomly, it is better to document the tests if time permits and note down the deviations if any. If defects are found, corresponding test cases are created so that it helps the testers to retest the scenario.

What is Agile Testing?

A software testing practice that follows the principles of agile software development is called Agile Testing. Agile is an iterative development methodology, where requirements evolve through collaboration between the customer and self-organizing teams and agile aligns development with customer needs.

Advantages of Agile Testing

- Agile Testing Saves Time and Money
- Less Documentation
- Regular feedback from the end user
- Daily meetings can help to determine the issues well in advance

Principles of Agile Testing

- **Testing is NOT a Phase:** Agile team tests continuously and continuous testing is the only way to ensure continuous progress.
- **Testing Moves the project Forward:** When following conventional methods, testing is considered as quality gate but agile testing provide feedback on an ongoing basis and the product meets the business demands.
- **Everyone Tests:** In conventional SDLC, only test team tests while in agile including developers and BA's test the application.
- **Shortening Feedback Response Time:** In conventional SDLC, only during the acceptance testing, the Business team will get to know the product development, while in agile for each and every iteration, they are involved and continuous feedback shortens the feedback response time and cost involved in fixing is also less.
- **Clean Code:** Raised defects are fixed within the same iteration and thereby keeping the code clean.
- **Reduce Test Documentation:** Instead of very lengthy documentation, agile testers use reusable checklist, focus on the essence of the test rather than the incidental details.
- **Test Driven:** In conventional methods, testing is performed after implementation while in agile testing, testing is done while implementation.

What is Requirements based Testing?

Requirements-based testing is a testing approach in which test cases, conditions and data are derived from requirements. It includes functional tests and also non-functional attributes such as performance, reliability or usability.

Stages in Requirements based Testing:

- **Defining Test Completion Criteria -** Testing is completed only when all the functional and non-functional testing is complete.
- **Design Test Cases -** A Test case has five parameters namely the initial state or precondition, data setup, the inputs, expected outcomes and actual outcomes.
- **Execute Tests** - Execute the test cases against the system under test and document the results.
- **Verify Test Results -** Verify if the expected and actual results match each other.
- **Verify Test Coverage -** Verify if the tests cover both functional and non-functional aspects of the requirement.
- **Track and Manage Defects -** Any defects detected during the testing process goes through the defect life cycle and are tracked to resolution. Defect Statistics are maintained which will give us the overall status of the project.

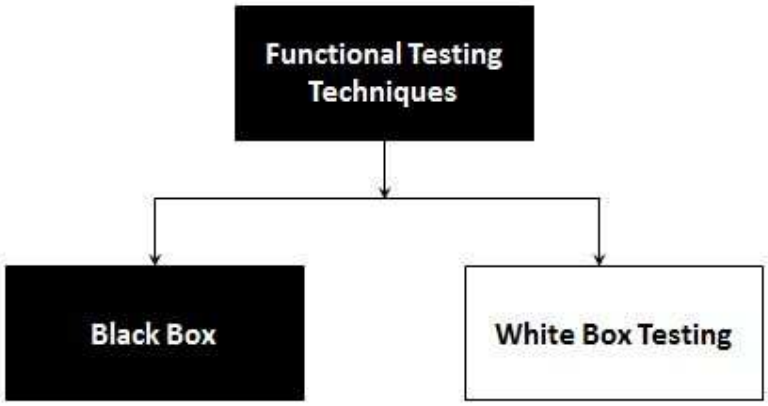
Requirements Testing process:

- Testing must be carried out in a timely manner.
- Testing process should add value to the software life cycle, hence it needs to be effective.
- Testing the system exhaustively is impossible hence the testing process needs to be efficient as well.
- Testing must provide the overall status of the project, hence it should be manageable.

What is Functional Testing?

Functional Testing is a testing technique that is used to test the features/functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases.

There are two major Functional Testing techniques as shown below:



The other major Functional Testing techniques include:

- Unit Testing
- Integration Testing
- Smoke Testing
- User Acceptance Testing
- Localization Testing
- Interface Testing
- Usability Testing
- System Testing
- Regression Testing
- Globalization Testing

What is Unit Testing?

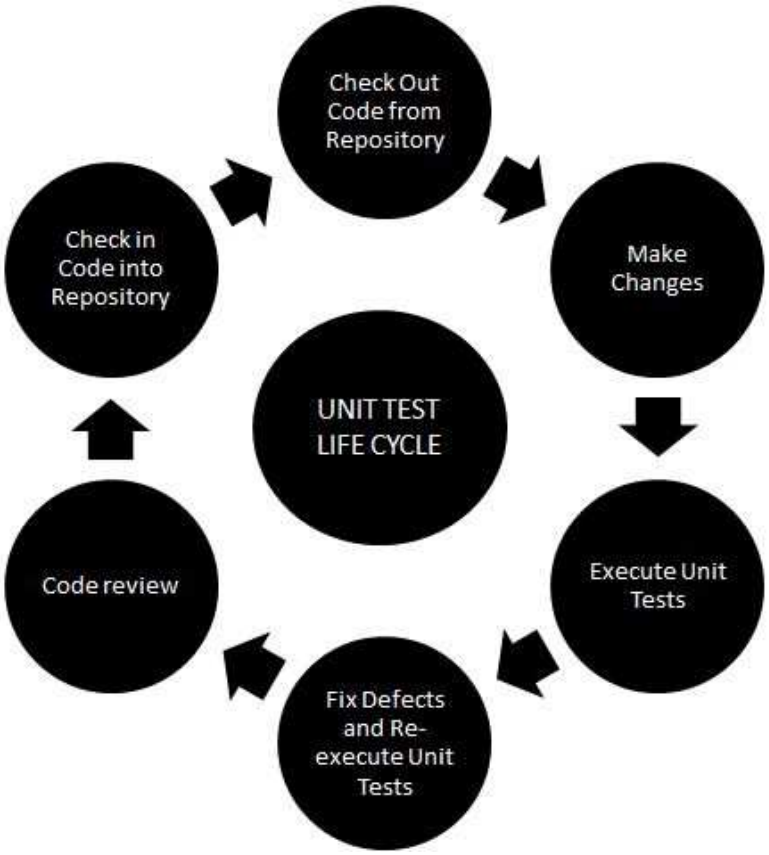
Unit testing, a testing technique using which individual modules are tested to determine if there are any issues by the developer himself. It is concerned with functional correctness of the standalone modules.

The main aim is to isolate each unit of the system to identify, analyze and fix the defects.

Unit Testing - Advantages:

- Reduces Defects in the Newly developed features or reduces bugs when changing the existing functionality.
- Reduces Cost of Testing as defects are captured in very early phase.
- Improves design and allows better refactoring of code.
- Unit Tests, when integrated with build gives the quality of the build as well.

Unit Testing LifeCyle:



Unit Testing Techniques:

- **Black Box Testing -** Using which the user interface, input and output are tested.
- **White Box Testing -** used to test each one of those functions behaviour is tested.
- **Gray Box Testing -** Used to execute tests, risks and assessment methods.

What is Use Case Testing?

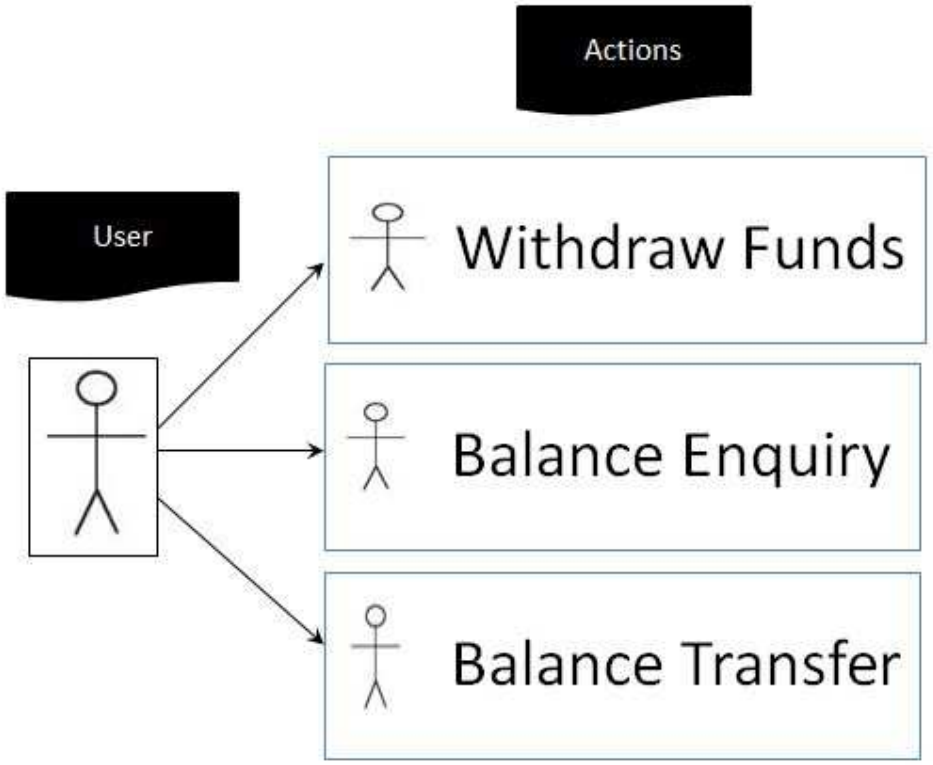
Use Case Testing is a functional black box testing technique that helps testers to identify test scenarios that exercise the whole system on each transaction basis from start to finish.

Characteristics of Use Case Testing:

- Use Cases capture the interactions between 'actors' and the 'system'.
- 'Actors' represents user and their interactions that each user takes part into.
- Test cases based on use cases and are referred as scenarios.
- Capability to identify gaps in the system which would not be found by testing individual components in isolation.
- Very effective in defining the scope of acceptance tests.

Example:

The Below example clearly shows the interaction between users and possible actions.



What is User Acceptance Testing?

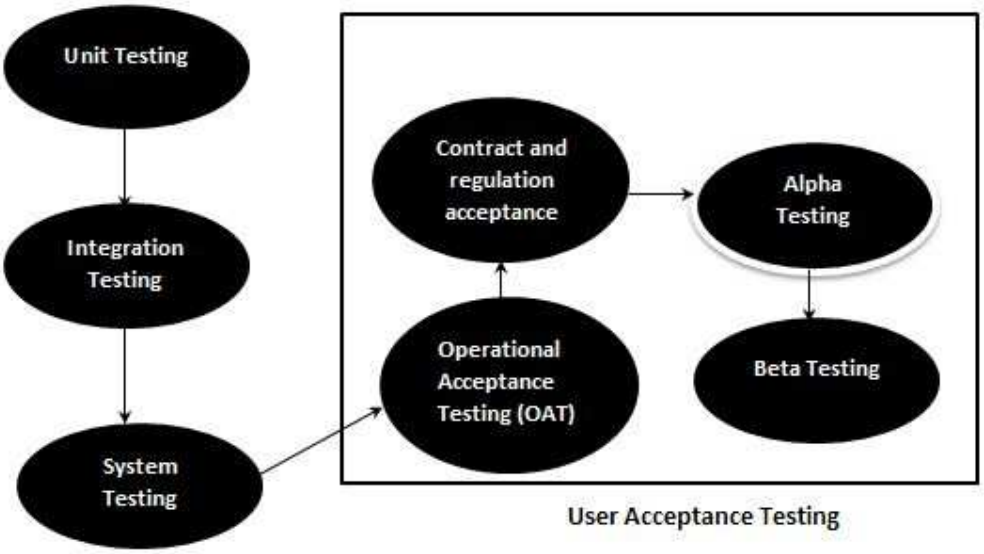
User acceptance testing, a testing methodology where the clients/end users involved in testing the product to validate the product against their requirements. It is performed at client location at developer's site.

For industry such as medicine or aviation industry, contract and regulatory compliance testing and operational acceptance testing is also carried out as part of user acceptance testing.

UAT is context dependent and the UAT plans are prepared based on the requirements and NOT mandatory to execute all kinds of user acceptance tests and even coordinated and contributed by testing team.

User Acceptance Testing - In SDLC :

The following diagram explains the fitment of user acceptance testing in the software development life cycle:



The acceptance test cases are executed against the test data or using an acceptance test script and then the results are compared with the expected ones.

Acceptance Criteria :

Acceptance criteria are defined on the basis of the following attributes:

- Functional Correctness and Completeness
- Data Integrity
- Data Conversion
- Usability
- Performance
- Timeliness
- Confidentiality and Availability

- Installability and Upgradability
- Scalability
- Documentation

Acceptance Test Plan – Attributes:

The acceptance test activities are carried out in phases. Firstly the basic tests are executed and if the test results are satisfactory then the execution of more complex scenarios are carried out.

The Acceptance test plan has the following attributes

- Introduction
- Acceptance Test Category
- operation Environment
- Test case ID
- Test Title
- Test Objective
- Test Procedure
- Test Schedule
- Resources

The acceptance test activities are designed to reach at one of the conclusions:

1. Accept the system as delivered
2. Accept the system after the requested modifications have been made
3. Do not accept the system

Acceptance Test Report – Attributes:

The Acceptance test Report has the following attributes:

- Report Identifier
- Summary of Results
- Variations
- Recommendations
- Summary of To-DO List
- Approval Decision

What is User Interface Testing?

User interface testing, a testing technique used to identify the presence of defects is a product/software under test by using Graphical user interface [GUI].

GUI Testing - Characteristics:

- GUI is a hierarchical, graphical front end to the application, contains graphical objects with a set of properties.
- During execution, the values of the properties of each objects of a GUI define the GUI state.
- It has capabilities to exercise GUI events like key press/mouse click.
- Able to provide inputs to the GUI Objects.
- To check the GUI representations to see if they are consistent with the expected ones.
- It strongly depends on the used technology.

GUI Testing - Approaches:

- **Manual Based -** Based on the domain and application knowledge of the tester.
- **Capture and Replay -** Based on capture and replay of user actions.
- **Model-based testing -** Based on the execution of user sessions based on a GUI model. Various GUI models are briefly discussed below.

Model Based Testing - In Brief:

- **Event-based model -** Based on all events of the GUI need to be executed at least once.
- **State-based model -** "all states" of the GUI are to be exercised at least once.
- **Domain model -** Based on the application domain and its functionality.

GUI Testing Checklist:

- Check Screen Validations
- Verify All Navigations
- Check usability Conditions
- Verify Data Integrity
- Verify the object states
- Verify the date Field and Numeric Field Formats

GUI Automation Tools:

Following are some of the open source GUI automation tools in the market:

Product	Licensed Under	URL
AutoHotkey	GPL	http://www.autohotkey.com/
Selenium	Apache	http://docs.seleniumhq.org/

Sikuli	MIT	http://sikuli.org
Robot Framework	Apache	www.robotframework.org
watir	BSD	http://www.watir.com/
Dojo Toolkit	BSD	http://dojotoolkit.org/

Following are some of the Commercial GUI automation tools in the market.

Product	Vendor	URL
AutoIT	AutoIT	http://www.autoitscript.com/site/autoit/
EggPlant	TestPlant	www.testplant.com
QTP	Hp	http://www8.hp.com/us/en/software-solutions/
Rational Functional Tester	IBM	http://www-03.ibm.com/software/products/us/en/functional
Infragistics	Infragistics	www.infragistics.com
iMacros	iOpus	http://www.iopus.com/iMacros/
CodedUI	Microsoft	http://www.microsoft.com/visualstudio/
Sikuli	Micro Focus International	http://www.microfocus.com/

What is Vulnerability Testing?

Vulnerability testing, a software testing technique performed to evaluate the quantum of risks involved in the system in order to reduce the probability of the event.

Vulnerability Testing - checklist:

- Verify the strength of the password as it provides some degree of security.
- Verify the access controls with the Operating systems/technology adopted.
- Verifies how easily the system can be taken over by online attackers.
- Evaluates the safety level of the data of system.
- Checks if the system configuration or application configuration files are protected.
- Checks if the system allows user to execute malicious script.

Vulnerability Testing - Methods:

- Active and Passive testing
- Network and distributed testing
- Verifying File/system access

What is Workflow Testing?

Workflow processes technique in software testing by routing a record through each possible path. These tests are performed to ensure that each workflow process accurately reflects the business process.

This kind of testing holds good for workflow-based applications.

Workflow Testing - Process:

- Understand the business workflow
- Develop test cases using various techniques (use case, decision table, etc.).
- Verify the flow with various user types (viz - Admin, Update user, View).
- Perform positive and negative tests.
- Compare the expected and actual results and log defects.
- Fix defects and deploy.

White Box Testing

White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

White Box Testing Techniques:

- **Statement Coverage -** This technique is aimed at exercising all programming statements with minimal tests.
- **Branch Coverage -** This technique is running a series of tests to ensure that all branches are tested at least once.
- **Path Coverage -** This technique corresponds to testing all possible paths which means that each statement and branch is covered.

Calculating Structural Testing Effectiveness:

Statement Testing = (Number of Statements Exercised / Total Number of Statements) x 100 %

Branch Testing = (Number of decisions outcomes tested / Total Number of decision Outcomes) x 100 %

Path Coverage = (Number paths exercised / Total Number of paths in the program) x 100 %

Advantages of White Box Testing:

- Forces test developer to reason carefully about implementation.
- Reveals errors in "hidden" code.
- Spots the Dead Code or other issues with respect to best programming practices.

Disadvantages of White Box Testing:

- Expensive as one has to spend both time and money to perform white box testing.
- Every possibility that few lines of code are missed accidentally.
- In-depth knowledge about the programming language is necessary to perform white box testing.

Non-Functional Testing

Non-Functional testing is a software testing technique that verifies the attributes of the system such as memory leaks, performance or robustness of the system. Non-Functional testing is performed at all test levels.

Non-Functional Testing Techniques:

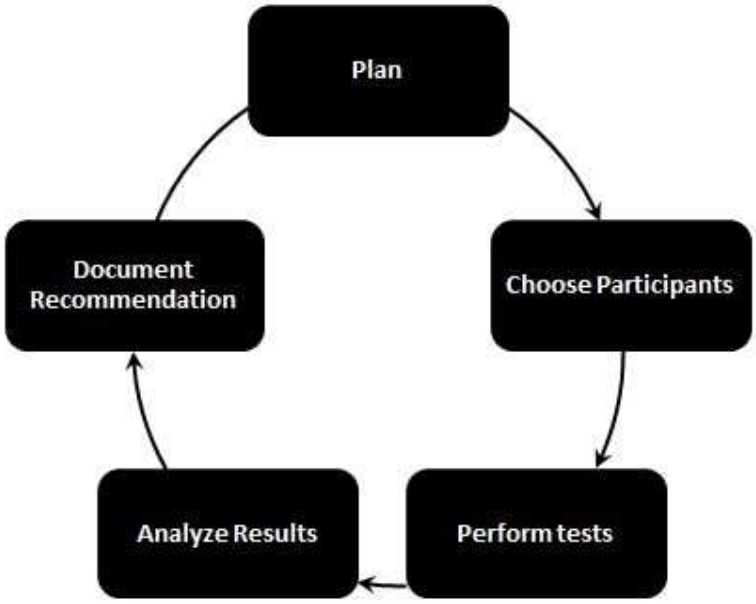
- Compatibility testing
- Compliance testing
- Endurance testing
- Load testing
- Localization testing
- Internationalization testing
- Performance testing
- Recovery testing
- Resilience testing
- Security testing
- Scalability testing
- Stress testing
- Usability testing
- Volume testing

What is Usability Testing ?

Usability testing, a non-functional testing technique that is a measure of how easily the system can be used by end users. It is difficult to evaluate and measure but can be evaluated based on the below parameters:

- Level of Skill required to learn/use the software. It should maintain the balance for both novice and expert user.
- Time required to get used to in using the software.
- The measure of increase in user productivity if any.
- Assessment of a user's attitude towards using the software.

Usability Testing Process:



What is Security Testing?

Security testing is a testing technique to determine if an information system protects data and maintains functionality as intended. It also aims at verifying 6 basic principles as listed below:

- Confidentiality
- Integrity
- Authentication
- Authorization
- Availability
- Non-repudiation

Security Testing - Techniques:

- Injection
- Broken Authentication and Session Management
- Cross-Site Scripting (XSS)
- Insecure Direct Object References
- Security Misconfiguration
- Sensitive Data Exposure
- Missing Function Level Access Control

- Cross-Site Request Forgery (CSRF)
- Using Components with Known Vulnerabilities
- Unvalidated Redirects and Forwards

Open Source/Free Security Testing Tools:

Product	Vendor	URL
FxCop	Microsoft	https://www.owasp.org/index.php/FxCop
FindBugs	The University of Maryland	http://findbugs.sourceforge.net/
FlawFinder	GPL	http://www.dwheeler.com/flawfinder/
Ramp Ascend	GPL	http://www.deque.com

Commercial Security Testing Tools:

Product	Vendor	URL
Armorize CodeSecure	Armorize Technologies	http://www.armorize.com/index.php?link_id=codesecure
GrammaTech	GrammaTech	http://www.grammatech.com/
Appscan	IBM	http://www-03.ibm.com/software/products/en/appscan-source
Veracode	VERACODE	http://www.veracode.com

What is Volume Testing?

Volume testing is a Non-functional testing that is performed as part of performance testing where the software is subjected to a huge volume of data. It is also referred as flood testing.

Volume Testing Characteristics:

- During development phase, only small amount of data is tested.
- The performance of the software deteriorates over time as there is enormous amount of data overtime.
- Test cases are derived from design documents.
- Test data is usually generated using a test data generator.
- Test data need not be logically correct but the data is to assess the system performance.
- Upon completion of testing, results are logged and tracked to bring it to closure.

Volume Testing - Checklist:

- Verify if there is any data loss.
- Check the system's response time.
- Verify if the data is stored incorrectly.
- Check if the data is overwritten without any notification.

What is Acceptance Testing?

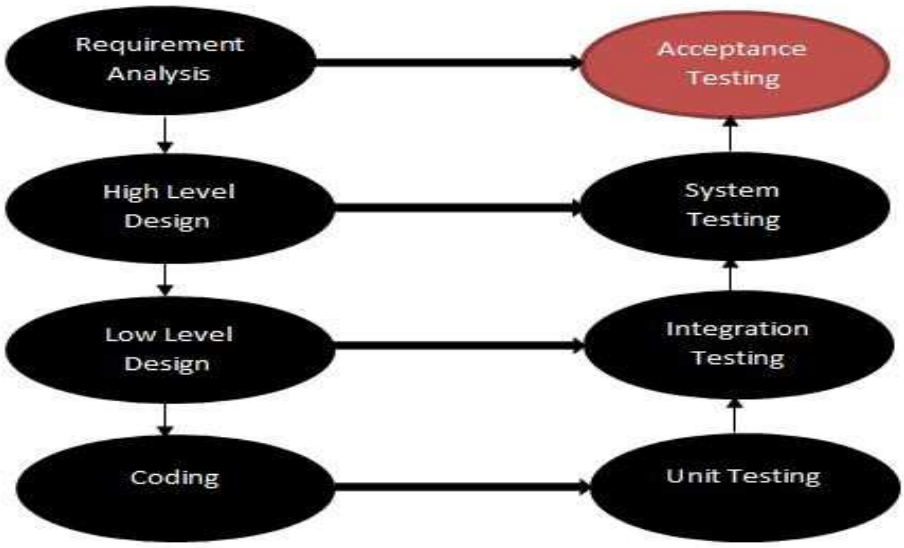
Acceptance testing, a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it is has met the required criteria for delivery to end users.

There are various forms of acceptance testing:

- User acceptance Testing
- Business acceptance Testing
- Alpha Testing
- Beta Testing

Acceptance Testing - In SDLC

The following diagram explains the fitment of acceptance testing in the software development life cycle.



The acceptance test cases are executed against the test data or using an acceptance test script and then the results are compared with the expected ones.

Acceptance Criteria :

Acceptance criteria are defined on the basis of the following attributes

- Functional Correctness and Completeness
- Data Integrity
- Data Conversion
- Usability
- Performance

- Timeliness
- Confidentiality and Availability
- Installability and Upgradability
- Scalability
- Documentation

Acceptance Test Plan - Attributes

The acceptance test activities are carried out in phases. Firstly, the basic tests are executed, and if the test results are satisfactory then the execution of more complex scenarios are carried out.

The Acceptance test plan has the following attributes:

- Introduction
- Acceptance Test Category
- operation Environment
- Test case ID
- Test Title
- Test Objective
- Test Procedure
- Test Schedule
- Resources

The acceptance test activities are designed to reach at one of the conclusions:

1. Accept the system as delivered
2. Accept the system after the requested modifications have been made
3. Do not accept the system

Acceptance Test Report - Attributes

The Acceptance test Report has the following attributes:

- Report Identifier
- Summary of Results
- Variations
- Recommendations
- Summary of To-DO List
- Approval Decision

What is Compatibility Testing?

Compatibility testing is a non-functional testing conducted on the application to evaluate the application's compatibility within different environments. It can be of two types - forward compatibility testing and backward compatibility testing.

- Operating system Compatibility Testing - Linux , Mac OS, Windows
- Database Compatibility Testing - Oracle SQL Server
- Browser Compatibility Testing - IE , Chrome, Firefox
- Other System Software - Web server, networking/ messaging tool, etc.

What is Configuration Testing?

Configuration testing is the process of testing the system with each one of the supported software and hardware configurations. The Execution area supports configuration testing by allowing reuse of the created tests.

- Executing Tests with Various Configurations:
- Operating System Configuration - Win XP, Win 7 32 bit/64 bit, Win 8 32 bit/64 bit
- Database Configuration - Oracle, DB2, MySql, MSSQL Server, Sybase
- Browser Configuration - IE 8, IE 9, FF 16.0, Chrome

What is Data Integrity Testing?

Data integrity corresponds to the quality of data in the databases and to the level by which users examine data quality, integrity and reliability. Data integrity testing verifies that the data in the database is accurate and functions as expected within a given application.

Characteristics of Data Integrity Testing Data Integrity testing involves:

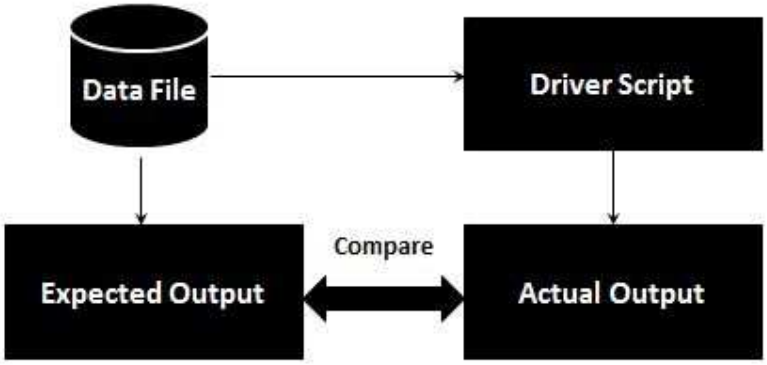
- Checking whether or NOT a blank value or default value can be retrieved from the database.
- Validating each value if it is successfully saved to the database.
- Ensuring the data compatibility against old hardware or old versions of operating systems.
- Verifying the data in data tables can be modified and deleted
- Running data tests for all data files, including clip art, tutorials, templates, etc.

What is Data Driven Testing?

Data-driven testing is creation of test scripts where test data and/or output values are read from data files instead of using the same hard-coded values each time the test runs. This way, testers can test how the application handles various inputs effectively. It can be any of the below data files.

- Datapools
- Excel files
- ADO objects
- CSV files
- ODBC sources

Flow Diagram:



What is Database Testing?

Database testing involves the retrieved values from the database by the web or desktop application. Data in the User Interface should be matched as per the records are stored in the database.

Database Testing Validations the following verifications are carried out during database testing:

- Checking the data Mapping.
- ACID (Atomicity, Consistency, Isolation, Durability) properties validation.
- Data Integrity
- Business rule conformance

What is Exhaustive Testing?

Exhaustive testing is a test approach in which all possible data combinations are used for testing. Exploratory testing includes implicit data combinations present in the state of the software/data at the start of testing.

Example:
Consider an application in which a password field that accepts 3 characters, with no consecutive repeating entries. Hence, there are 26 * 26 * 26 input permutations for alphabets only. Including special characters and standard characters, there are much more combinations. So, there are 256 * 256 * 256 input combinations.

What is Globalization Testing?

A product is said to be Globalized when that particular product can be run independent of its geographical and cultural environment. This type of testing technique validates whether the application can be used all over the world that accepts all the language texts.

What needs to be Tested ?

- Sensitivity to the language vocabulary
- Date and time formatting
- Currency handling
- Paper sizes for printing
- Address and telephone number formatting
- Zip Code Format

Advantages of Globalization Testing

- It reduces overall testing and support costs
- It helps us to reduce time for testing which result faster time-to-market
- It is more flexible and product is easily scalable

What is Localization Testing?

Localization testing is performed to verify the quality of a product's localization for a particular target culture/locale and is executed only on the localized version of the product.

Localization Testing - Characteristics:

- Modules affected by localization, such as UI and content
- Modules specific to Culture/locale-specific, language-specific, and region-specific
- Critical Business Scenarios Testing
- Installation and upgrading tests run in the localized environment
- Plan application and hardware compatibility tests according to the product's target region.

Localization Testing - UI Testing:

- Check for linguistic errors and resource attributes
- Typographical errors
- Verify the systems adherence to the input, and display environment standards
- Usability testing of the User interface
- Verify cultural appropriateness of UI such as colour, design, etc.

What is GUI Software Testing?

GUI testing is a testing technique in which the application's user interface is tested whether the application performs as expected with respect to user interface behaviour.

GUI Testing includes the application behaviour towards keyboard and mouse movements and how different GUI objects such as toolbars, buttons, menubars, dialog boxes, edit fields, lists, behavior to the user input.

GUI Testing Guidelines:

- Check Screen Validations
- Verify All Navigations
- Check usability Conditions
- Verify Data Integrity
- Verify the object states
- Verify the date Field and Numeric Field Formats

GUI Automation Tools :

Following are some of the **open source** GUI automation tools in the market:

Product	Licensed Under	URL
AutoHotkey	GPL	http://www.autohotkey.com/
Selenium	Apache	http://docs.seleniumhq.org/
Sikuli	MIT	http://sikuli.org
Robot Framework	Apache	www.robotframework.org
watir	BSD	http://www.watir.com/
Dojo Toolkit	BSD	http://dojotoolkit.org/

Following are some of the Commercial GUI automation tools in the market.

Product	Vendor	URL
AutoIT	AutoIT	http://www.autoitscript.com/site/autoit/
EggPlant	TestPlant	www.testplant.com

QTP	Hp	http://www8.hp.com/us/en/software-solutions/
Rational Functional Tester	IBM	http://www-03.ibm.com/software/products/us/en/functional
Infragistics	Infragistics	www.infragistics.com
iMacros	iOpus	http://www.iopus.com/iMacros/
CodedUI	Microsoft	http://www.microsoft.com/visualstudio/
Sikuli	Micro Focus International	http://www.microfocus.com/

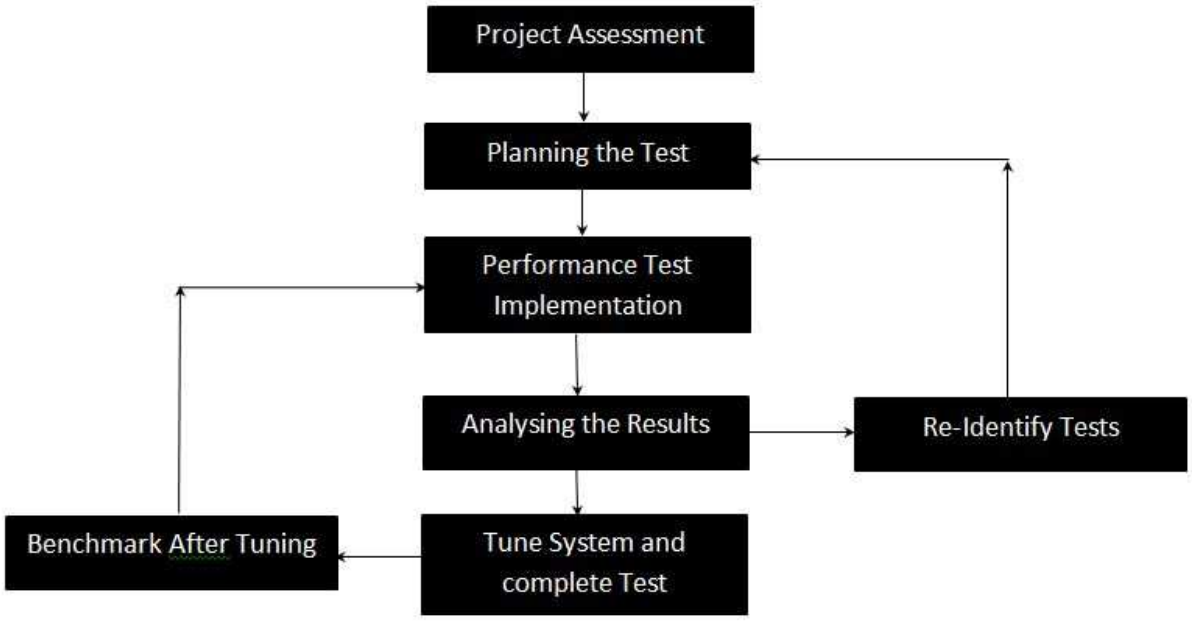
What is Performance Testing?

Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.

Performance Testing Techniques:

- **Load testing -** It is the simplest form of testing conducted to understand the behaviour of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc., are also monitored.
- **Stress testing -** It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected maximum.
- **Soak testing -** Soak Testing also known as endurance testing, is performed to determine the system parameters under continuous expected load. During soak tests the parameters such as memory utilization is monitored to detect memory leaks or other performance issues. The main aim is to discover the system's performance under sustained use.
- **Spike testing -** Spike testing is performed by increasing the number of users suddenly by a very large amount and measuring the performance of the system. The main aim is to determine whether the system will be able to sustain the workload.

Performance Testing Process:



Attributes of Performance Testing:

- Speed
- Scalability
- Stability
- reliability

Performance Testing Tools :

- Jmeter - http://jmeter.apache.org/
- Open STA - http://opensta.org/
- Load Runner - http://www.hp.com/
- Web Load - http://www.radview.com/

What is Load Testing ?

Load testing is performance testing technique using which the response of the system is measured under various load conditions. The load testing is performed for normal and peak load conditions.

Load Testing Approach:

- Evaluate performance acceptance criteria
- Identify critical scenarios
- Design workload Model
- Identify the target load levels
- Design the tests
- Execute Tests
- Analyze the Results

Objectives of Load Testing:

- Response time
- Throughput
- Resource utilization
- Maximum user load
- Business-related metrics

What is a load generator?

Load Generator is a system, which is used to simulate load for performing performance testing. It can be for concurrency testing or SQL performance Testing. It is a system, which sends a request remotely called as host system or load driving system.

Load distribution among load generators is a very usual way to generate load while performing load testing.

Example: Hp - Load Runner, Apache Jmeter

Virtual Users:

Virtual user is a common terminology used from a performance testing point of view. A Virtual user generator enables testers to create virtual users in order to increase the user load on the application under test.

Virtual user generator captures the requests to create a virtual user and it can read user operations.

What is Stress Testing?

Stress testing a Non-Functional testing technique that is performed as part of performance testing. During stress testing, the system is monitored after subjecting the system to overload to ensure that the system can sustain the stress.

The recovery of the system from such phase (after stress) is very critical as it is highly likely to happen in production environment.

Reasons for conducting Stress Testing:

- It allows the test team to monitor system performance during failures.
- To verify if the system has saved the data before crashing or NOT.
- To verify if the system prints meaning error messages while crashing or did it print some random exceptions.
- To verify if unexpected failures do not cause security issues.

Stress Testing - Scenarios:

- Monitor the system behaviour when maximum number of users logged in at the same time.
- All users performing the critical operations at the same time.
- All users Accessing the same file at the same time.
- Hardware issues such as database server down or some of the servers in a server park crashed.

What is Portability Testing?

Portability testing is a process of testing with ease with which the software or product can be moved from one environment to another. It is measured in terms of maximum amount of effort required to transfer from one system to another system.

The portability testing is performed regularly throughout the software development life cycle in an iterative and incremental manner.

Portability Testing attributes:

- Adaptability
- Installability
- Replaceability
- Co-existence

Portability Testing Checklists:

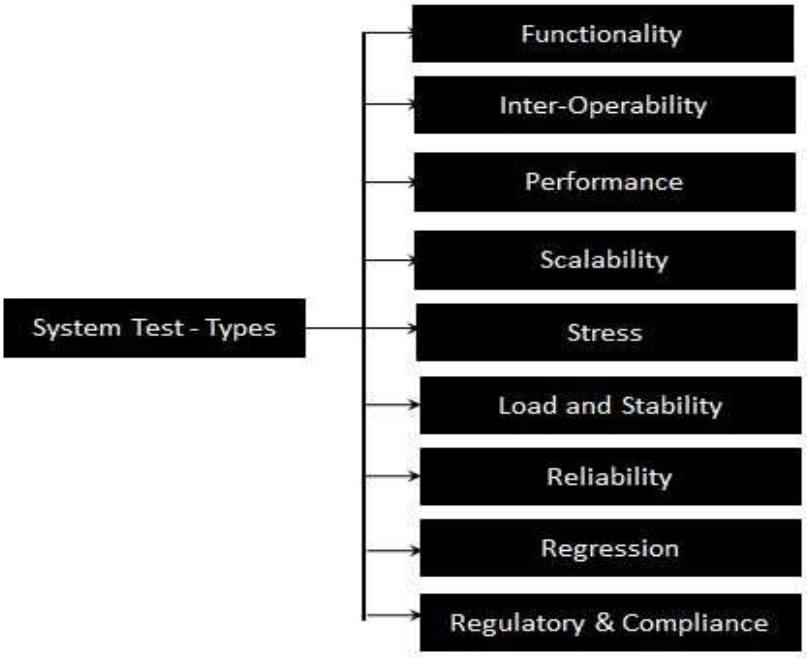
- Verify if the application is able to fulfil the portability requirements.
- Determine the look and feel of the application in the various browser types and various browser versions.
- Report the defects to the development teams so that they can be associated and defects can be fixed.
- The failures during the portability testing can help to identify defects that were not detected during unit and integration testing.

What is System Testing?

System Testing (ST) is a black box testing technique performed to evaluate the complete system the system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective.

System Testing is usually carried out by a team that is independent of the development team in order to measure the quality of the system unbiased. It includes both functional and Non-Functional testing.

Types of System Tests:



What is Test Environment?

Test Environment consists of elements that support test execution with software, hardware and network configured. Test environment configuration must mimic the production environment in order to uncover any environment/configuration related issues.

Factors for designing Test Environment:

- Determine if test environment needs archiving in order to take backups.
- Verify the network configuration.
- Identify the required server operating system, databases and other components.
- Identify the number of license required by the test team.

Environmental Configuration:
It is the combination of hardware and software environment on which the tests will be executed. It includes hardware configuration, operating system settings, software configuration, test terminals and other support to perform the test.

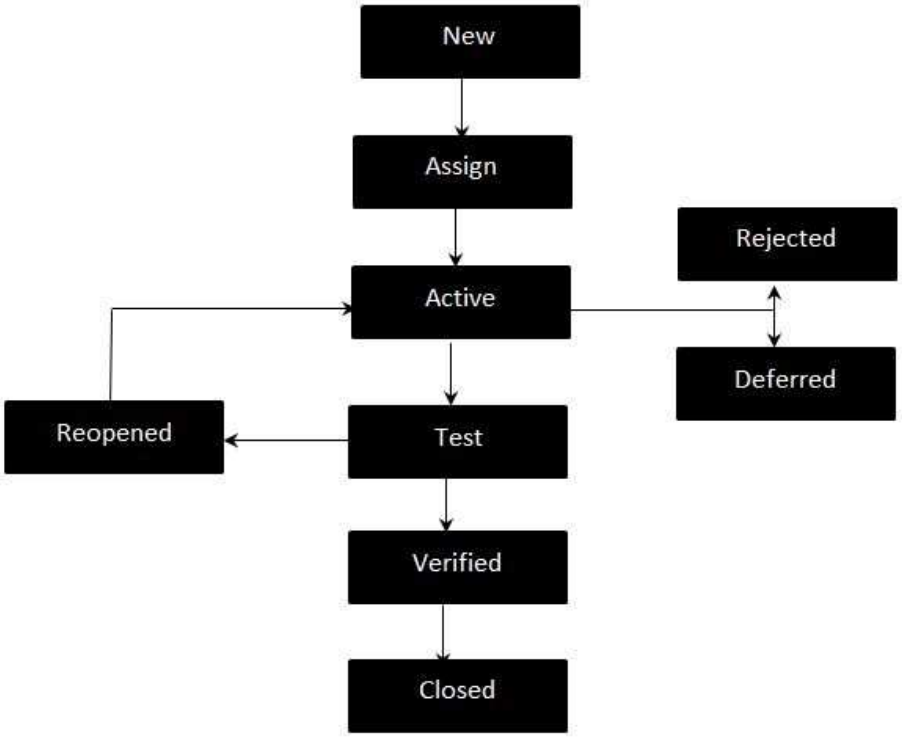
Example:
A typical Environmental Configuration for a web-based application is given below:

```
Web Server - IIS/Apache
Database - MS SQL
OS - Windows/ Linux
Browser - IE/FireFox
Java version : version 6
```

What is a Bug?

In Software testing, when the expected and actual behavior is not matching, an incident needs to be raised. An incident may be a Bug. It is a programmer's fault where a programmer intended to implement a certain behavior, but the code fails to correctly conform to this behavior because of incorrect implementation in coding. It is also known as Defect.Life Cycle .

Defect Life Cycle - Workflow:



Defect Life Cycle States:

- **New -** Potential defect that is raised and yet to be validated.
- **Assigned -** Assigned against a development team to address it but not yet resolved.
- **Active -** The Defect is being addressed by the developer and investigation is under progress. At this stage there are two possible outcomes; viz - Deferred or Rejected.
- **Test -** The Defect is fixed and ready for testing.
- **Verified -** The Defect that is retested and the test has been verified by QA.
- **Closed -** The final state of the defect that can be closed after the QA retesting or can be closed if the defect is duplicate or considered as NOT a defect.
- **Reopened -** When the defect is NOT fixed, QA reopens/reactivates the defect.
- **Deferred -** When a defect cannot be addressed in that particular cycle it is deferred to future release.

- **Rejected -** A defect can be rejected for any of the 3 reasons; viz - duplicate defect, NOT a Defect, Non Reproducible.

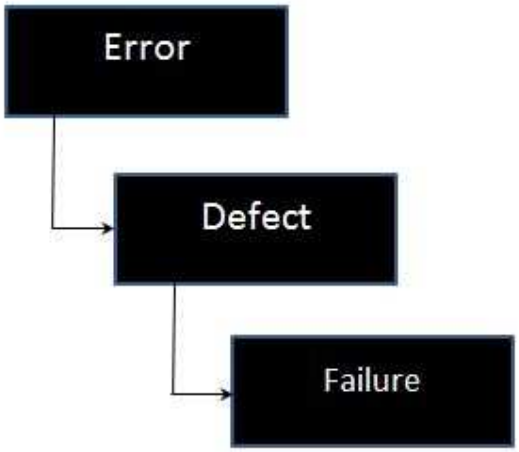
Parameters of a Bug:

- Date of issue, author, approvals and status.
- Severity and priority of the incident.
- The associated test case that revealed the problem
- Expected and actual results.
- Identification of the test item and environment.
- Description of the incident with steps to Reproduce
- Status of the incident
- Conclusions, recommendations and approvals.

What is a Failure?

Under certain circumstances, the product may produce wrong results. It is defined as the deviation of the delivered service from compliance with the specification.

Not all the defects result in failure as defects in dead code do not cause failure.



Reasons for Failure:

- Environmental conditions, which might cause hardware failures or change in any of the environmental variables.
- Human Error while interacting with the software by keying in wrong inputs.
- Failures may occur if the user tries to perform some operation with intention of breaking the system.

Results of Failure

- Loss of Time
- Loss of Money
- Loss of Business Reputation.
- Injury
- Death

What is a Fault?

Software fault is also known as defect, arises when the expected result don't match with the actual results. It can also be error, flaw, failure, or fault in a computer program. Most bugs arise from mistakes and errors made by developers, architects.

Fault Types :

- Business Logic Faults
- Functional and Logical Faults
- Faulty GUI
- Performance Faults
- Security Faults

Preventing Faults

Following are the methods for preventing programmers from introducing Faulty code during development:

- Programming Techniques adopted
- Software Development methodologies
- Peer Review
- Code Analysis

What is retesting?

Retesting is executing a previously failed test against new software to check if the problem is resolved. After a defect has been fixed, retesting is performed to check the scenario under the same environmental conditions.

During Retesting, testers look for granular details at the changed area of functionality, whereas regression testing covers all the main functions to ensure that no functionalities are broken due to this change.

What is Regression Testing?

Regression testing a black box testing technique that consists of re-executing those tests that are impacted by the code changes. These tests should be executed as often as possible throughout the software development life cycle.

Types of Regression Tests:

- **Final Regression Tests: -** A "final regression testing" is performed to validate the build that hasn't changed for a period of time. This build is deployed or shipped to customers.
- **Regression Tests: -** A normal regression testing is performed to verify if the build has NOT broken any other parts of the application by the recent code changes for defect fixing or for enhancement.

Selecting Regression Tests:

- Requires knowledge about the system and how it affects by the existing functionalities.
- Tests are selected based on the area of frequent defects.
- Tests are selected to include the area, which has undergone code changes many a times.
- Tests are selected based on the criticality of the features.

Regression Testing Steps:

Regression tests are the ideal cases of automation which results in better **R**eturn **O**n **I**nvestment (ROI).

- Select the Tests for Regression.
- Choose the apt tool and automate the Regression Tests
- Verify applications with Checkpoints
- Manage Regression Tests/update when required
- Schedule the tests
- Integrate with the builds
- Analyze the results

What is Sanity Testing?

Sanity testing, a software testing technique performed by the test team for some basic tests. The aim of basic test is to be conducted whenever a new build is received for testing. The terminologies such as Smoke Test or Build Verification Test or Basic Acceptance Test or Sanity Test are interchangeably used, however, each one of them is used under a slightly different scenario.

Sanity test is usually unscripted, helps to identify the dependent missing functionalities. It is used to determine if the section of the application is still working after a minor change.

Sanity testing can be narrow and deep. Sanity test is a narrow regression test that focuses on one or a few areas of functionality.

What is Smoke Testing?

Smoke Testing is a testing technique that is inspired from hardware testing, which checks for the smoke from the hardware components once the hardware's power is switched on. Similarly in Software testing context, smoke testing refers to testing the basic functionality of the build.

If the Test fails, build is declared as unstable and it is NOT tested anymore until the smoke test of the build passes.

Smoke Testing - Features:

- Identifying the business critical functionalities that a product must satisfy.
- Designing and executing the basic functionalities of the application.
- Ensuring that the smoke test passes each and every build in order to proceed with the testing.
- Smoke Tests enables uncovering obvious errors which saves time and effort of test team.
- Smoke Tests can be manual or automated.

What is Priority and Severity in Software Testing?

Priority :

As already discussed priority determines how quickly the defect turnaround time must be. If there are multiple defects, the priority decides which defect has to be fixed and verified immediately versus which defect can be fixed a bit later.

While opening a defect, the tester generally assigns the priority initially as he views the product from the end user perspective. In line with these, here are different levels:

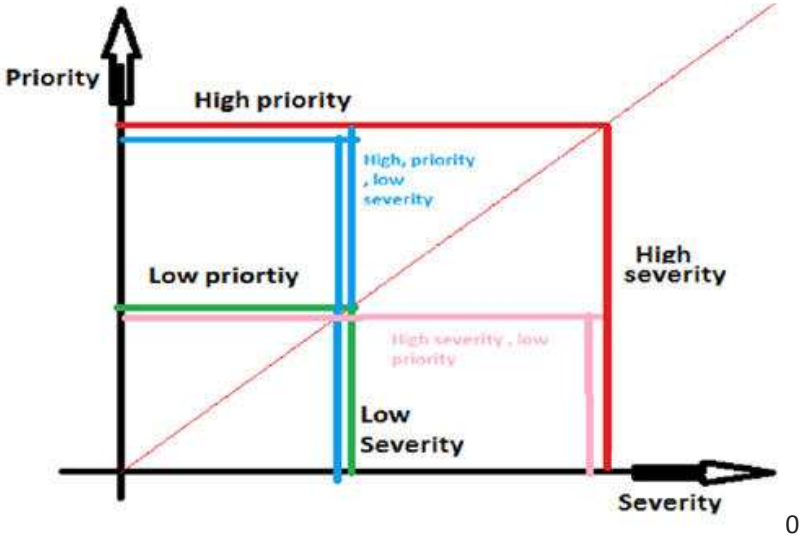
- **Critical (P1):** This has to be fixed immediately within 24 hours. This generally occurs in cases when an entire functionality is blocked and no testing can proceed as a result of this. Or in certain other cases if there are significant memory leaks, then generally the defect is classified as a priority -1 meaning the program/ feature is unusable in the current state.
- **High (P2):** Once the critical defects have been fixed, a defect having this priority is the next candidate which has to be fixed for any test activity to match the “exit” criteria. Normally when a feature is not usable as it’s supposed to be, due to a program defect, or that a new code has to be written or sometimes even because some environmental problem has to be handled through the code, a defect may qualify for a priority 2.
- **Medium (P3)**: A defect with this priority must be in contention to be fixed as it could also deal with functionality issues which is not as per expectation. Sometimes even cosmetic errors such as expecting the right error message during the failure could qualify to be a priority 3 defect.
- **Low (P4):** A defect with low priority indicates that there is definitely an issue, but it doesn’t have to be fixed to match the “exit” criteria. However this must be fixed before the GA is done. Typically, some typing errors or even cosmetic errors as discussed previously could be

categorized in here. Sometimes defects with priority low are also opened to suggest some enhancements in the existing design or a request to implement a small feature to enhance user experience.

Severity:

Severity is a parameter to denote the implication of defect on the system – how critical defect is and what is the impact of the defect on the whole system's functionality? The severity is a parameter set by the tester while he opens a defect and is mainly in control of the tester. Again different organizations have different tools to use for defects, but on a generic level these are the following severity levels:

- **Critical / Show Stopper (S1):** A defect that completely hampers or blocks testing of the product/ feature is a critical defect. An example would be in case of UI testing where after going through a wizard, the UI just hangs at one pane or doesn't go further to trigger the function. Or in some other cases, when the feature developed itself is missing from the build.
- **Major or Severe (S2):** A major defect occurs when the functionality is functioning grossly away from the expectations or not doing what it should be doing. An example could be: Say that a VLAN needs to be deployed on the switch and you are using a UI template that triggers this function. When this template to configure VLAN fails on the switch, it gets classified as a severe functionality drawback.
- **Moderate/ Normal (S3):** A moderate defect occurs when the product or application doesn't meet certain criteria or still exhibits some unnatural behavior, however the functionality as a whole is not impacted. For example in the VLAN template deploy above, a moderate or normal defect would occur when the template is deployed successfully on the switch however there is no indication being sent to the user.
- **Low or Minor (S4):** A minor bug occurs when there is almost no impact to the functionality, but is still a valid defect that should be corrected. Examples of this could include spelling mistakes in error messages printed to user or defects to enhance the look and feel of a feature.



Examples of sorting defects with Priority and Severity :

High Priority & High Severity: An error which occurs on the basic functionality of the application and will not allow the user to use the system. (Eg. A site maintaining the student details, on saving record if it, doesn't allow to save the record then this is high priority and high severity bug.)

High Priority & Low Severity: The spelling mistakes that happens on the cover page or heading or title of an application.

High Severity & Low Priority: An error which occurs on the functionality of the application (for which there is no workaround) and will not allow the user to use the system but on click of link which is rarely used by the end user.

Low Priority and Low Severity: Any cosmetic or spelling issues which is within a paragraph or in the report (Not on cover page, heading, title).

Examples of sorting defects with Priority and Severity For Banking Domain :

High Priority & High Severity: Any Bank account holder withdraw the Money it's own account using ATM but it Charged For Rs.20

High Priority & Low Severity: Bank Logo Spelling Format or Wrong Spell of Bank Name.

High Severity & Low Priority: Bank gives interest Rs.2 for Rs.1000 at end of the year. But Bank found bug instead of Rs.2 application gives Rs.4 for Rs.1000.It means interest rate is double.

What is Test case?

A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

Typical Test Case Parameters:

- Test Case ID
- Test Scenario
- Test Case Description
- Test Steps
- Prerequisite
- Test Data
- Expected Result
- Test Parameters
- Actual Result
- Environment Information
- Comments

Example:

Let us say that we need to check an input field that can accept maximum of 10 characters.

While developing the test cases for the above scenario, the test cases are documented the following way. In the below example, the first case is a pass scenario while the second case is a FAIL.

Scenario	Test Step	Expected Result	Actual Outcome
Verify that the input field that can accept maximum of 10 characters	Login to application and key in 10 characters	Application should be able to accept all 10 characters.	Application accepts all 10 characters.
Verify that the input field that can accept maximum of 11 characters	Login to application and key in 11 characters	Application should NOT accept all 11 characters.	Application accepts all 10 characters.

If the expected result doesn't match with the actual result, then we log a defect. The defect goes through the defect life cycle and the testers address the same after fix.

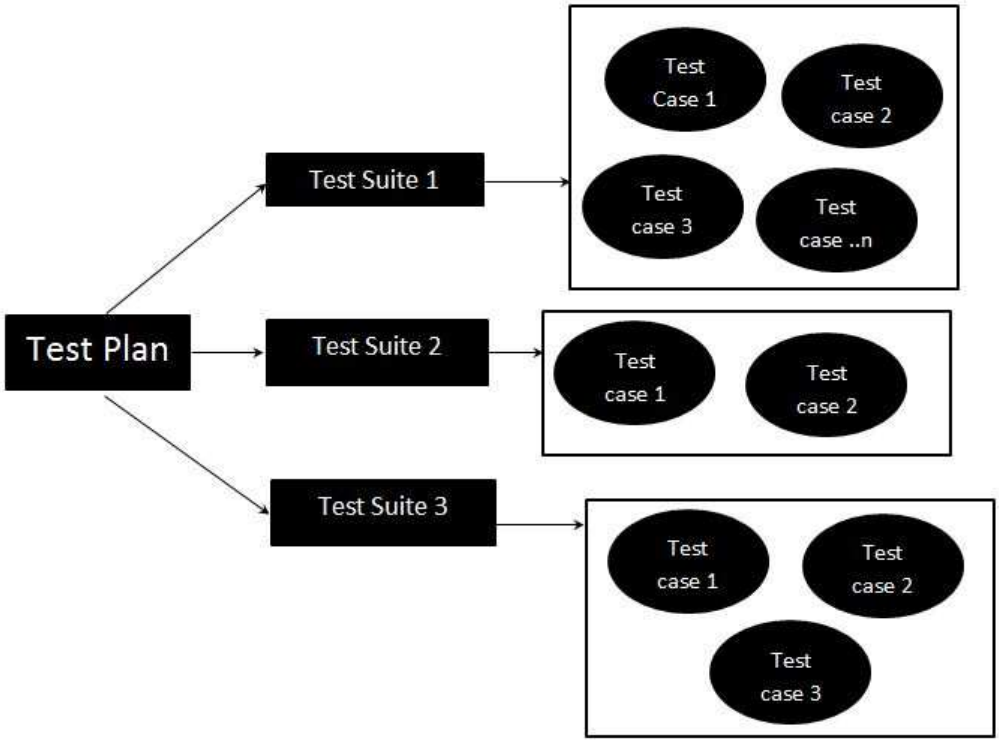
What is a Test Suite?

Test suite is a container that has a set of tests which helps testers in executing and reporting the test execution status. It can take any of the three states namely Active, Inprogress and completed.

A Test case can be added to multiple test suites and test plans. After creating a test plan, test suites are created which in turn can have any number of tests.

Test suites are created based on the cycle or based on the scope. It can contain any type of tests, viz - functional or Non-Functional.

Test Suite - Diagram:



Difference between Test case and Test scenario:

- Test case consist of set of input values, execution precondition, excepted Results and executed post condition, developed to cover certain test Condition. While Test scenario is nothing but test procedure.
- A Test Scenarios have one to many relation with Test case, Means A scenario have multiple test case. Every time we have write test cases for test scenario. So while starting testing first prepare test scenarios then create different-2 test cases for each scenario.
- Test cases are derived (or written) from test scenario. The scenarios are derived from use cases.
- Test Scenario represents a series of actions that are associated together. While test Case represents a single (low level) action by the user.
- Scenario is thread of operations where as Test cases are set of input and output given to the System.

For example:

Checking the functionality of Login button is Test scenario and
Test Cases for this Test Scenario are:
1. Click the button without entering user name and password.
2. Click the button only entering User name.
3. Click the button while entering wrong user name and wrong password.

Difference between Re-testing and Regression testing:

Re-Testing:

- When tester finds the bug and report to developers and developer fix that bug now if tester tests only that test case in which he found the bug with same or different data then it is known as retesting.
- Re testing requires re-running of the failed test cases.
- Re testing is plan based for bug fixes in build notes and docs.

Regression Testing:

- After modification or fixing the bug if tester test that test case in which he found the bug and test all the or specified test cases which he executes earlier then it is known as regression testing . The aim of this testing is that bug fixing is not affect the passed test cases.
- It is an important activity, performed on modified software to provide confidence that changes are correct and do not affect the other functionality and components.
- Regression testing is generic and may not be always specific to defect fixes.

Difference between Sanity and Smoke Testing:

Smoke Testing:

- When a build is received smoke testing is done to ensure that whether the build is ready or stable for further testing.
- Smoke testing is a wide approach where all areas of software application are tested without getting into deeper.
- Test Cases for smoke testing can be manual or automated.
- Smoke testing is conducted to ensure whether the most crucial functions of a program are working, but not bothering with finer details.

Sanity Testing:

- After receiving a software build, with minor changes in code, or functionality, Sanity testing is performed to ascertain that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected.
- Sanity testing exercises only the particular component of the entire system.
- A sanity test is usually unscripted and without test scripts or test cases.
- Sanity Testing is narrow and deep
- Sanity testing is to verify whether requirements are met or not, checking all features breadth-first
- Sanity Testing is like specialized health check up

Difference between Load Testing and Stress Testing:

- Testing the app with maximum number of user and input is defined as **load testing.** While testing the app with more than maximum number of user and input is defined as **stress testing**.
- **In Load testing we** measure the system performance based on a volume of users**. While in Stress testing we** measure the breakpoint of a system.

Load Testing is testing the application for a given load requirements which may include any of the following criteria:

1. Total number of users.
2. Response Time
3. Through Put
4. Some parameters to check State of servers/application.

- While stress testing is testing the application for unexpected load. It includes
  1. Vusers
  2. Think-Time

Example:

If an app is build for 500 users, then for load testing we check up to 500 users and for stress testing we check greater than 500.

What is Test Data?

Test Data is data that is used to execute the tests on testware. Test data needs to be precise and exhaustive to uncover the defects.

Test data preparation tools:

Product	Vendor	URL
DTM Data Generator	SQLEdit	http://www.sqledit.com/
SQL Data Generator	Red-Gate	http://www.red-gate.com/
EMS Data Generator	EMS	http://www.sqlmanager.net/
E-Naxos DataGen	E-Naxos	http://www.e-naxos.com/UsIndex.html
IBM DB2 Test Database	IBM	http://www.ibm.com/us/en/

Test Data Generation Techniques:

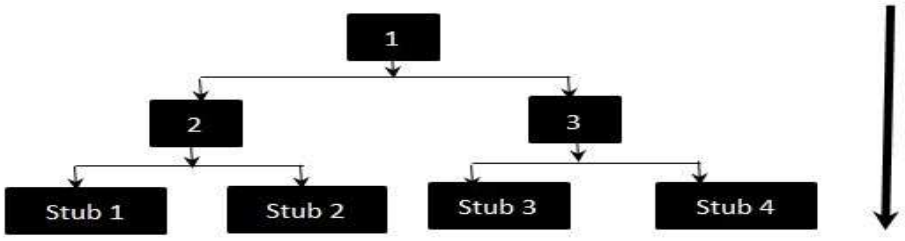
- Random Test Data Generators
- Goal-Oriented Test Data Generators
- Pathwise Test Data Generators
- Intelligent Test Data Generators

What is a Stub?

Stubs are used during Top-down integration testing, in order to simulate the behaviour of the lower-level modules that are not yet integrated. Stubs are the modules that act as temporary replacement for a called module and give the same output as that of the actual product.

Stubs are also used when the software needs to interact with an external system.

Stub - Flow Diagram :



The above diagram clearly states that Modules 1, 2 and 3 are available for integration, whereas, below modules are still under development that cannot be integrated at this point of time. Hence, Stubs are used to test the modules. The order of Integration will be:

1,2
1,3
2,Stub 1
2,Stub 2
3,Stub 3
3,Stub 4

Testing Approach:

+ Firstly, the integration between the modules 1,2 and 3
+ Test the integration between the module 2 and stub 1,stub 2
+ Test the integration between the module 3 and stub 3,stub 4

Difference between defect, error, bug, failure and fault:

"A mistake in coding is called error ,error found by tester is called defect, defect accepted by development team then it is called bug ,build does not meet the requirements then it Is failure."

Error: A discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. This can be a misunderstanding of the internal state of the software, an oversight in terms of memory management, confusion about the proper way to calculate a value, etc.

Failure: The inability of a system or component to perform its required functions within specified performance requirements. See: bug, crash, exception, and fault.

Bug: A fault in a program which causes the program to perform in an unintended or unanticipated manner. See: anomaly, defect, error, exception, and fault. Bug is terminology of Tester.

Fault: An incorrect step, process, or data definition in a computer program which causes the program to perform in an unintended or unanticipated manner. See: bug, defect, error, exception.

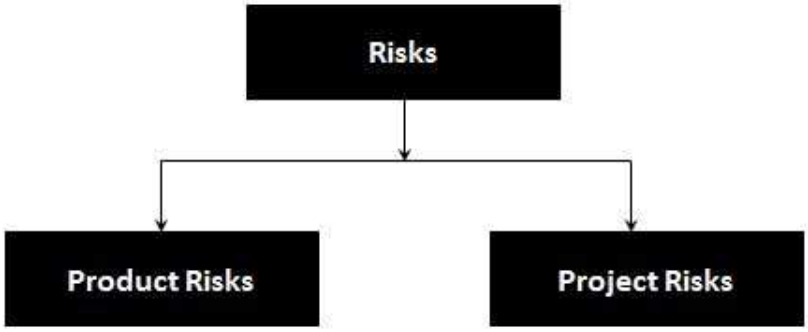
Defect: Commonly refers to several troubles with the software products, with its external behavior or with its internal features.

What is Risk?

Risk can be defined as the probability of an event, hazard, accident, threat or situation occurring and its undesirable consequences. It is a factor that could result in negative consequences and usually expressed as the product of impact and likelihood.

Risk = Probability of the event occurring x Impact if it did happen

Risk Types:
In software terminology, the risk is broadly divided into two main categories:



Project Risks:

- Supplier issues
- Organizational factors
- Technical issues

Product Risks:

Below are some of the product risks occurring in a LIVE environment:

- Defect Prone Software delivered
- The Critical defects in the product that could cause harm to an individual or company
- Poor software Features
- Inconsistent Software Features

What is Risk Management?

Risk management is the process of identifying, assessing, and prioritizing the risks to minimize, monitor, and control the probability of unfortunate events.

Risk Management Process:
Risk Management process can be easily understood with use of the following workflow:

Risk Management Practices:

- Software Risk Evaluation (SRE)
- Continuous Risk Management (CRM)
- Team Risk Management (TRM)

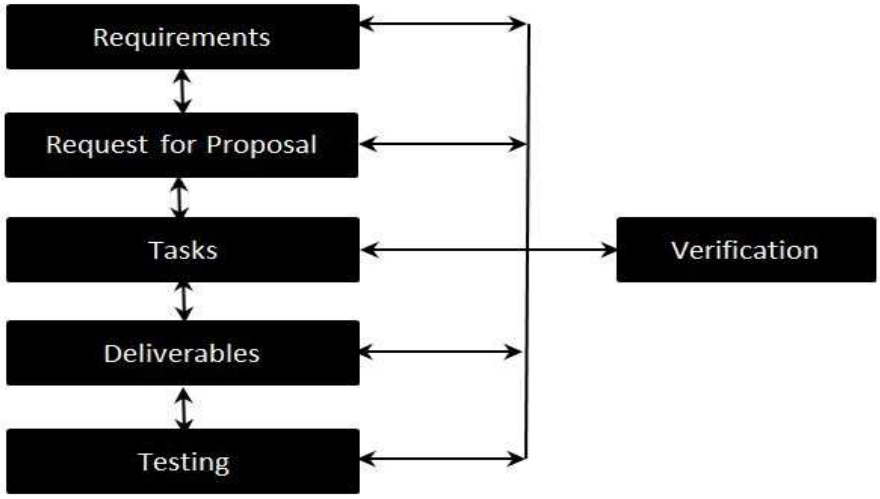
What is Requirement traceability Matrix (RTM)?

Requirements tracing, a process of documenting the links between the requirements and the work products developed to implement and verify those requirements. The RTM captures all requirements and their traceability in a single document delivered at the conclusion of the life cycle.

RTM - WorkFlow:

The Matrix is created at the very beginning of a project as it forms the basis of the project's scope and deliverables that will be produced.

The Matrix is bi-directional, as it tracks the requirement forward by examining the output of the deliverables and backward by looking at the business requirement that was specified for a particular feature of the product.



Requirement traceability Matrix - Parameters:

- Requirement ID
- Risks
- Requirement Type
- Requirement Description
- Trace to Design Specification
- Unit Test Cases
- Integration Test Cases
- System Test Cases
- User Acceptance Test Cases
- Trace to Test Script

What is Release Notes?

Release notes is a document, which is released as part of the final build that contains new enhancements that went in as part of that release and also the known issues of that build.

Release Notes are usually written by technical writers which are communication documents shared with clients. Release notes also feed the process of end-user documentation, user guide and training materials.

Release Notes Format:

- **Header -** Name of the document, which carries product name, release number, release date, release note date and version.
- **Overview -** An overview of the product and changes to the recent software version.
- **Purpose -** An overview of the purpose of the release notes which lists the new feature, enhancements and defects of the current build.
- **Issue Summary -** Provides description about the defect.
- **End-User Impact -** Provides information about the end-users impact due to the defect.
- **Contact -** Support contact information.

Testing Tools:

Tools from a software testing context can be defined as a product that supports one or more test activities right from planning, requirements, creating a build, test execution, defect logging and test analysis.

Classification of Tools :

Tools can be classified based on several parameters. They include:

- The purpose of the tool
- The Activities that are supported within the tool
- The Type/level of testing it supports
- The Kind of licensing (open source, freeware, commercial)
- The technology used

Types of Tools:

S.No.	Tool Type	Used for	Used by
1.	Test Management Tool	Test Managing, scheduling, defect logging, tracking and analysis.	testers
2.	Configuration management tool	For Implementation, execution, tracking changes	All Team members
3.	Static Analysis Tools	Static Testing	Developers

4.	Test data Preparation Tools	Analysis and Design, Test data generation	Testers
5.	Test Execution Tools	Implementation, Execution	Testers
6.	Test Comparators	Comparing expected and actual results	All Team members
7.	Coverage measurement tools	Provides structural coverage	Developers
8.	Performance Testing tools	Monitoring the performance, response time	Testers
9.	Project planning and Tracking Tools	For Planning	Project Managers
10.	Incident Management Tools	For managing the tests	Testers

What is Test Automation?

Software Test automation makes use of specialized tools to control the execution of tests and compares the actual results against the expected result. Usually, regression tests, which are repetitive actions, are automated.

Testing Tools not only helps us to perform regression tests but also helps us to automate data set up generation, product installation, GUI interaction, defect logging, etc. Automation tools are used for both Functional and Non-Functional testing.

Criteria for Tool Selection:

For automating any application, the following parameters should be considered:

- Data driven capabilities
- Debugging and logging capabilities
- Platform independence
- Extensibility & Customizability
- E-mail Notifications
- Version control friendly
- Support unattended test runs

Types of Frameworks:

Typically, there are 4 test automation frameworks that are adopted while automating the applications:

- Data Driven Automation Framework
- Keyword Driven Automation Framework

- Modular Automation Framework
- Hybrid Automation Framework

Popular Tools that are used for Functional automation:

Product	**Vendor**	**URL**
Quick Test Professional	HP	www.hp.com/go/qtp
Rational Robot	IBM	http://www-03.ibm.com/software/products/us/en/robot/
Coded UI	Microsoft	http://msdn.microsoft.com/en-us/library/dd286726.aspx
Selenium	Open Source	http://docs.seleniumhq.org/
Auto IT	Open Source	http://www.autoitscript.com/site/

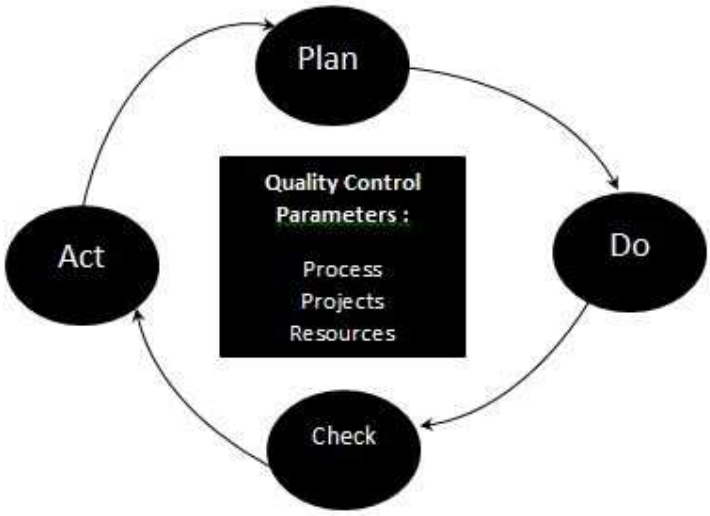
Popular Tools that are used for Non Functional automation:

Product	**Vendor**	**URL**
Load Runner	HP	www.hp.com/go/LoadRunner
Jmeter	Apache	jmeter.apache.org/
Burp Suite	PortSwigger	http://portswigger.net/burp/
Acunetix	Acunetix	http://www.acunetix.com/

What is Quality Control?

Quality control is a set of methods used by organizations to achieve quality parameters or quality goals and continually improve the organization's ability to ensure that a software product will meet quality goals.

Quality Control Process:



The three class parameters that control software quality are:

- Products
- Processes
- Resources

The total quality control process consists of:

- **Plan** - It is the stage where the Quality control processes are planned
- **Do** - Use a defined parameter to develop the quality
- **Check** - Stage to verify if the quality of the parameters are met
- **Act** - Take corrective action if needed and repeat the work

Quality Control characteristics:

- Process adopted to deliver a quality product to the clients at best cost.
- Goal is to learn from other organizations so that quality would be better each time.
- To avoid making errors by proper planning and execution with correct review process.

What is Quality Assurance?

Quality Assurance is defined as the auditing and reporting procedures used to provide the stakeholders with data needed to make well-informed decisions.
It is the Degree to which a system meets specified requirements and customer expectations. It is also monitoring the processes and products throughout the SDLC.

Quality Assurance Criteria:
Below are the Quality assurance criteria against which the software would be evaluated against:

- correctness
- efficiency
- flexibility
- integrity
- interoperability
- maintainability
- portability
- reliability
- reusability
- testability
- usability

What is Web Application Testing?

Web application testing, a software testing technique exclusively adopted to test the applications that are hosted on web in which the application interfaces and other functionalities are tested.

Web Application Testing - Techniques:

1. **Functionality Testing** - The below are some of the checks that are performed but not limited to the below list:

- Verify there is no dead page or invalid redirects.
- First check all the validations on each field.
- Wrong inputs to perform negative testing.
- Verify the workflow of the system.
- Verify the data integrity.

2. **Usability testing -** To verify how the application is easy to use with.

- Test the navigation and controls.
- Content checking.
- Check for user intuition.

3. **Interface testing -** Performed to verify the interface and the dataflow from one system to other.

4. **Compatibility testing-** Compatibility testing is performed based on the context of the application.

- Browser compatibility
- Operating system compatibility
- Compatible to various devices like notebook, mobile, etc.

5. **Performance testing -** Performed to verify the server response time and throughput under various load conditions.

- **Load testing -** It is the simplest form of testing conducted to understand the behaviour of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc. are also monitored.
- **Stress testing -** It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected maximum.
- **Soak testing -** Soak Testing also known as endurance testing, is performed to determine the system parameters under continuous expected load. During soak tests the parameters such as memory utilization is monitored to detect memory leaks or other performance issues. The main aim is to discover the system's performance under sustained use.
- **Spike testing -** Spike testing is performed by increasing the number of users suddenly by a very large amount and measuring the performance of the system. The main aim is to determine whether the system will be able to sustain the work load.

6. **Security testing -** Performed to verify if the application is secured on web as data theft and unauthorized access are more common issues and below are some of the techniques to verify the security level of the system.

- Injection
- Broken Authentication and Session Management
- Cross-Site Scripting (XSS)
- Insecure Direct Object References
- Security Misconfiguration
- Sensitive Data Exposure
- Missing Function Level Access Control
- Cross-Site Request Forgery (CSRF)
- Using Components with Known Vulnerabilities
- Unvalidated Redirects and Forwards