Unit 01: Basics of software testing

Software testing

- Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is defect free.
- The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.
- The goal of software tester to find whether developed software or product met the specified requirement or not . And to identifys the defect to ensure that the product is defect free in order to produce a quality product.
- In simple word, software testing is an activity to check that the software system is defect free.
- So the goal of software tester is to find bugs and find them as early as possible and make sure they are fixed.
- Testing can be done in two ways: **Manual testing** and **automated testing.** Manual testing is done by human testers who check codes and bugs in software manually. On the other hand, automated testing is the process of using computer programs to execute a system.

Quality:

- "Fit for use or purpose." It is all about meeting the needs and expectations of customers with respect to functionality, design, reliability, durability, & price of the product.
- Quality of a product or service is its ability to satisfy the needs and expectations of the customer.
- Quality can briefly be defined as “a degree of excellence”. High quality software
- usually conforms to the user requirements.

Software Quality:

- software quality measures how well the software is designed (quality of design), and how well the software conforms to that design (quality of conformance). It is often described as the ‘fitness for purpose’ of a piece of software.”

Why Software Testing is Important?

Software Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

What are the benefits of Software Testing?

Here are the benefits of using software testing:

Cost-Effective: It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.

Security: It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

Product quality: It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.

Customer Satisfaction: The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

DEFECT: It can be simply defined as a variance between expected and actual. Defect is an error found AFTER the application goes into production. In other words Defect is the difference between expected and actual result in the context of testing. It is the deviation of the customer requirement.

Failure: Lots of defect leads to failure of the software.

Error Problem in code leads to the errors.

fault :fault (defect) is introduced into the software as the result of an error. It is an anomaly in the software that may cause it to behave incorrectly, and not according to its specification

BUG: A bug is the result of a coding error. An Error found in the development environment before the product is shipped to the customer. A programming error that causes a program to work poorly, produce incorrect results or crash. An error in software or hardware that causes a program to malfunction. Bug is terminology of Tester.

Objectives of testing

- Executing a program with the intent of finding an *error*.
- To check if the system meets the requirements and be executed successfully in the Intended environment.
- To check if the system is " Fit for purpose".
- To check if the system does what it is expected to do.
- A good test case is one that has a probability of finding an as yet undiscovered error.
- Gaining confidence in software application and providing information about the level of quality.
- Verifying that the final result meets the business and user requirements.
- Ensuring that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
- Gaining customers confidence by providing them a quality product.

TEST CASE

- A **TEST CASE** is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data,precondition, expected result,actual result developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.
- It is an in-details document that contains all possible inputs (positive as well as negative) and the navigation steps, which are used for the test execution process.
- Test case gives detailed information about testing strategy, testing process, preconditions, and expected output. These are executed during the testing process to check whether the software application is performing the task for that it was developed or not.
- Test case helps the tester in defect reporting by linking defect with test case ID.
- Detailed test case documentation works as a full proof guard for the testing team because if developer missed something, then it can be caught during execution of these full-proof test cases.
- A test case is the set of steps that need to be done in order to test a specific function of the software. They are developed for various scenarios so that testers can determine whether the software is working the way it should and producing the expected results.

When to Start Testing?(entry criteria)

- An early start to testing reduces the cost and time to rework and produce error-free software that is delivered to the client.

- testing can be started from the Requirements Gathering phase and continued till the deployment of the software.
- During the requirement gathering phase, the analysis and verification of requirements are also considered as testing.
- Reviewing the design in the design phase with the intent to improve the design is also considered as testing.
- Testing performed by a developer on completion of the code is also categorized as testing.

When to Stop Testing?(exit criteria)

The following aspects are to be considered for stopping the testing process −

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

Skills for software tester

- Find bugs as early as possible and make sure they get fixed.
- To understand the application well.
- Study the functionality in detail to find where the bugs are likely to occur.
- Study the code to ensure that each and every line of code is tested.
- Create test cases in such a way that testing is done to uncover the hidden bugs and also ensure that the software is usable and reliable

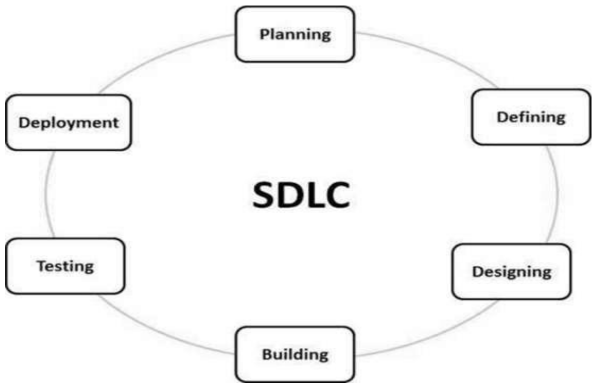
SDLC(software development life cycle)

- Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.
- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software Development Process.
- SDLC is a framework defining tasks performed at each step in the software development process.
- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

What is 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 an **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 must 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 product 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. Sometimes product deployment happens in stages as per the business strategy of that organization. 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.

- Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time.
  This procedure where the care is taken for the developed product is known as maintenance

<u>Quality Assurance</u>

- Quality Assurance is also known as **QA Testing**. ***QA*** is defined as an activity to ensure that an organization is providing the best product or service to the customers.
- Quality Assurance is a systematic way of creating an environment to ensure that the software product being developed meets the quality requirements. It is a preventive process whose aim is to establish the correct methodology and standard to provide a quality environment to the product being developed. Quality Assurance focuses on process standard, projects audit, and procedures for development.
- QA is also known as a set of activities designed to evaluate the process to produce a product.
- Software Quality assurance is all about the Software Development lifecycle that includes requirements management, software design, coding, testing, and release management.
- QA establishes and maintains set requirements for developing or manufacturing reliable products. A quality assurance system is meant to increase customer confidence and a company's credibility, while also improving work processes and efficiency, and it enables a company to better compete with others.
- Quality assurance helps a company create products and services that meet the needs, expectations and requirements of customers. It yields high-quality product offerings that build trust and loyalty with customers. The standards and procedures defined by a quality assurance program help prevent product defects before they arise.
- It's a Preventive technique.
- It is performed before Quality Control

<u>Quality Control</u>

- *Quality Control also known as QC* is a sequence of tasks to ensure the quality of software by identifying defects and correction of defects in the developed software. It is a reactive process, and the main purpose of this process is to correct all types of defects before releasing the software. The process is done by eliminating sources of problems (which cause to low the quality) through the corrective tools so that software can meet customer's requirements and high quality.
- The responsibility of quality control is of a specific team which is known as a testing team that tests the defects of software by validation and corrective tools.
- Quality control involves testing of units and determining if they are within the specifications for the final product. The purpose of the testing is to determine any needs for corrective actions in the manufacturing process. Good quality control helps companies meet consumer demands for better products.

<u>Difference between Quality Assurance (QA) and Quality Control (QC)</u>

Quality Assurance (QA)	**Quality Control (QC)**
It is a procedure that focuses on providing assurance that quality requested will be achieved	It is a procedure that focuses on fulfilling the quality requested.
QA aims to prevent the defect	QC aims to identify and fix defects

It is a method to manage the quality- Verification	It is a method to verify the quality-Validation
It does not involve executing the program	It always involves executing a program
It's a Preventive technique	It's a Corrective technique
It's a Proactive measure	It's a Reactive measure
It is the procedure to create the deliverables	It is the procedure to verify that deliverables
QA involves in full software development life cycle	QC involves in full software testing life cycle
In order to meet the customer requirements, QA defines standards and methodologies	QC confirms that the standards are followed while working on the product
It is performed before Quality Control	It is performed only after QA activity is done
It is a Low-Level Activity, it can identify an error and mistakes which QC cannot	It is a High-Level Activity, it can identify an error that QA cannot
Its main motive is to prevent defects in the system. It is a less time-consuming activity	Its main motive is to identify defects or bugs in the system. It is a more time-consuming activity
QA ensures that everything is executed in the right way, and that is why it falls under verification activity	QC ensures that whatever we have done is as per the requirement, and that is why it falls under validation activity
It requires the involvement of the whole team	It requires the involvement of the Testing team
The statistical technique applied on QA is known as SPC or Statistical Process Control (SPC)	The statistical technique applied to QC is known as SQC or Statistical Quality Control

KEY DIFFERENCE

- Quality Assurance is aimed to avoid the defect whereas Quality control is aimed to identify and fix the defects.
- Quality Assurance provides assurance that quality requested will be achieved whereas Quality Control is a procedure that focuses on fulfilling the quality requested.
- Quality Assurance is done in software development life cycle whereas Quality Control is done in software testing life cycle.
- Quality Assurance is a proactive measure whereas Quality Control is a Reactive measure.

- Quality Assurance requires the involvement of all team members whereas Quality Control needs only testing team.
- Quality Assurance is performed before Quality Control.

Verification

- is a process of checking documents, design, code, and program in order to check if the software has been built according to the requirements or not. The main goal of verification process is to ensure quality of software application, design, architecture etc. The verification process involves activities like reviews, walk-throughs and inspection.
- 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 includes different activities such as business requirements, system requirements, design review, and code walkthrough while developing a product.
- It is also known as static testing, where we are ensuring that "**we are developing the right product or not**". And it also checks that the developed application fulfilling all the requirements given by the client.

Validation

- 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 is testing where tester performed functional and non-functional testing.
  Here **functional testing** includes Unit Testing (UT), Integration Testing (IT) and System Testing (ST), and **non-functional** testing includes User acceptance testing (UAT).
- Validation testing is also known as dynamic testing, where we are ensuring that **"we have developed the product right."** And it also checks that the software meets the business needs of the client.

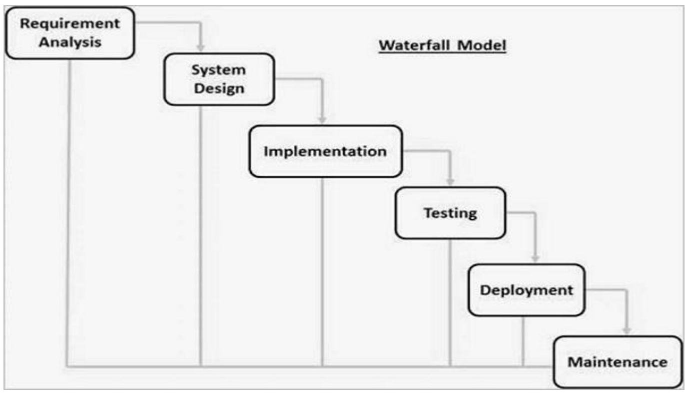
- Validation is done at the end of the development process and takes place after **verifications** are completed.
- It is a High level activity.
- Performed after a work product is produced against established criteria ensuring that the product integrates correctly into the environment.
- Determination of correctness of the final software product by a development project with respect to the user needs and requirements.

Waterfall Model:

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It 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.

- The Waterfall model is the earliest SDLC approach that was used for software development.
- The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.
- Each phase is designed for performing specific activity during the SDLC phase. It was introduced in 1970 by Winston Royce.

•



Requirement Gathering and analysis − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
In this phase, a large document called **Software Requirement Specification (SRS)** document is created which contained a detailed description of what the system will do in the common language.

Design phase − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language. It defines the overall software architecture together with high level and detailed design. All this work is documented as a Software Design Document (SDD).

Implementation: During this phase, design is implemented. If the SDD is complete, the implementation or coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD.
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.

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. The product or application is deemed fully functional and is deployed to a live environment.
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 - Advantages

- The advantages of waterfall development are 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.

  Some of the major advantages of the Waterfall Model are as follows −
- 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.

Waterfall Model - Disadvantages

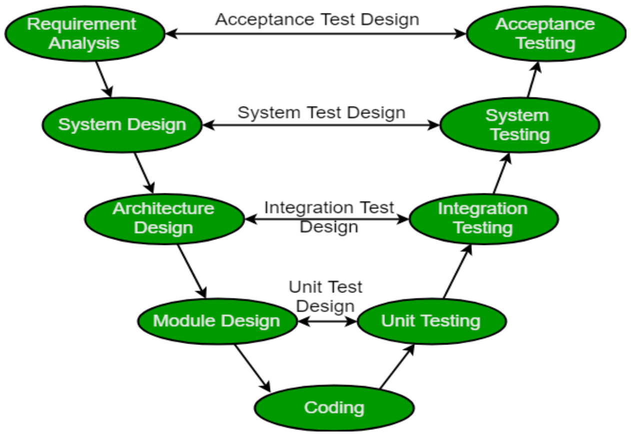
The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows −

- 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.
- 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.

The V-model

The V-model is a type of SDLC model where process executes in a sequential manner in V-shape. It is also known as Verification and Validation model. It is based on the association of a testing phase for each corresponding development stage. Development of each step directly associated with the testing phase. The next phase starts only after completion of the previous phase i.e. for each development activity, there is a testing activity corresponding to it.



Verification: It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements meet.

Validation: It involves dynamic analysis technique (functional, non-functional), testing done by executing code. Validation is the process to evaluate the software after the completion of the development phase to determine whether software meets the customer expectations and requirements.

- So V-Model contains Verification phases on one side of the Validation phases on the other side. Verification and Validation phases are joined by coding phase in V-shape. Thus it is called V-Model.

Design Phase:

- **Requirement Analysis:** This phase contains detailed communication with the customer to understand their requirements and expectations. This stage is known as Requirement Gathering.
- **System Design:** This phase contains the system design and the complete hardware and communication setup for developing product.
- **Architectural Design:** System design is broken down further into modules taking up different functionalities. The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood.
- **Module Design:** In this phase the system breaks dowm into small modules. The detailed design of modules is specified, also known
as Low-Level Design (LLD).

Coding Phase: After designing, the coding phase is started. Based on the requirements, a suitable programming language is decided. There are some guidelines and standards for coding. Before checking in the repository, the final build is optimized for better performance, and the code goes through many code reviews to check the performance.

Testing Phases:

- **Unit Testing:** Unit Test Plans are developed during module design phase. These Unit Test Plans are executed to eliminate bugs at code or unit level.
- **Integration testing:** After completion of unit testing Integration testing is performed. In integration testing, the modules are integrated and the system is tested. Integration testing is performed on the Architecture design phase. This test verifies the communication of modules among themselves.
- **System Testing:** System testing test the complete application with its functionality, inter dependency, and communication.It tests the functional and non-functional requirements of the developed application.
- **User Acceptance Testing (UAT):** UAT is performed in a user environment that resembles the production environment. UAT verifies that the delivered system meets user’s requirement and system is ready for use in real world.

Advantages:

- This is a highly disciplined model and Phases are completed one at a time.
- V-Model is used for small projects where project requirements are clear.
- Simple and easy to understand and use.
- This model focuses on verification and validation activities early in the life cycle thereby enhancing the probability of building an error-free and good quality product.
- It enables project management to track progress accurately.

Disadvantages:

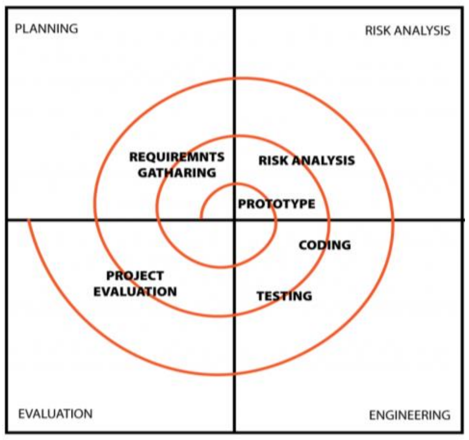
- High risk and uncertainty.
- It is not a good for complex and object-oriented projects.
- It is not suitable for projects where requirements are not clear and contains high risk of changing.
- This model does not support iteration of phases.
- It does not easily handle concurrent events.

Spiral Model:

Spiral Model was first described by Barry W. Boehm (American Software Engineer) in 1986.
The spiral model works in an iterative nature. It is a combination of both the Prototype development process and the Linear development process (waterfall model). This model places more emphasis on risk analysis. Mostly this model adapts to large and complicated projects where risk is high. Every Iteration starts with planning and ends with the product evaluation by the client.

The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

Spiral model is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a **Phase of the software development process.** The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using the spiral model.



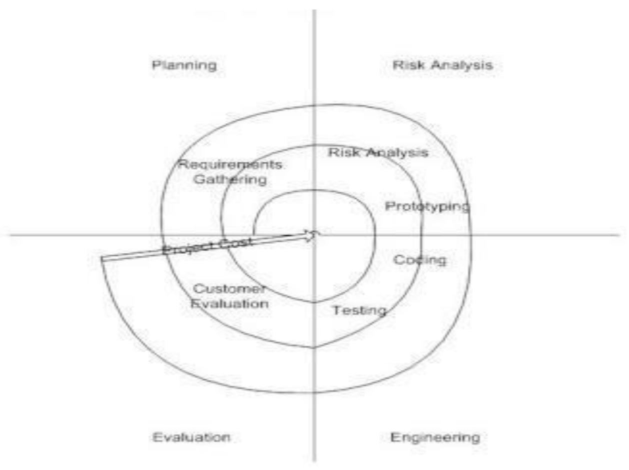
The spiral model has four phases:
-Planning,
-Risk Analysis,
-Engineering
- Evaluation

A software project repeatedly passes through these phases in iterations called Spirals.
Planning Phase: Requirements are gathered during the planning phase. Requirements like 'BRS' that is 'Bussiness Requirement Specifications' and 'SRS' that is 'System Requirement specifications.
The analyst gathered information about the requirements and start to understand what needs to be done.
This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst.

Risk Analysis: In the **risk analysis phase**, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis then alternate solutions are suggested and implemented.

Engineering Phase: In this phase software is **developed**, along with testing at the end of the phase. Hence in this phase the development and testing is done.

E**valuation phase:** This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.



Advantages

- -suitable for high risk project.
- Project monitoring is easy
- Changing requirements can be accommodate
- Users see the system early
- Additional Functionality can be added at a later date

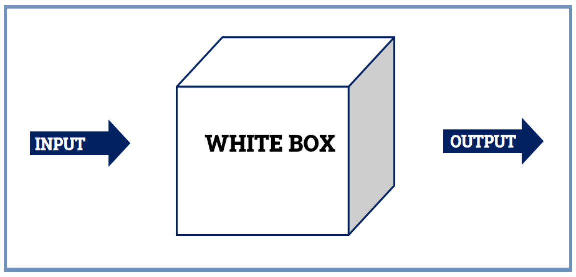
Disadvantages

- Can be a costly model to use.
- Doesn’t work well for smaller projects.
- Spiral may go on indefinitely.
- Large number of intermediate stages requires excessive documentation.
- Management is more complex.

Unit 02 Types of testing

White Box Testing

- White box testing techniques analyze the internal structures the used data structures, internal design, code structure and the working of the software .
- White Box Testing is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security.
- White box testing is software testing method in which internal structure/design is known to tester .The main aim of white box testing is check on how system is performing based on the code.it is mainly performed by the developer or white box testers who has knowledge on the programming.
- In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing,
  Code-based testing and Glass box testing, or structural testing.



Whitebox testing technique

1)Statement Coverage:- This technique requires every possible statement in the code to be tested at least once during the testing process.
in statement coverage, every node must be traversed at least once..
2)Branch Coverage - This technique checks every possible path (if-else and other conditional loops) of a software application.
Every edges must be traversed at least once.
As per flowchart all edges must be traversed at least once
3) Path Coverage
This technique is used to ensure that every possible path (each statement and branch) is executed and tested.

Types of White Box Testing

White box testing can take several forms:

- **Unit testing** — tests written as part of the application code, which test that each component is working as expected.
- **Mutation testing** — a type of unit testing that checks the robustness and consistency of the code by defining tests, making small, random changes to the code and seeing if the tests still pass.

Mutation testing is generally conducted to re-check any kind software bugs in the system.
Mutation testing is often used to discover the best coding techniques to use for expanding a software solution.

- **Integration testing** — tests specifically designed to check integration points between internal components in a software system, or integrations with external systems.
- **Penetration Testing:** In this testing, the tester/developer has full information of the application's source code, detailed network information, IP addresses involved and all server information the application runs on. The aim is to attack the code from several angles to expose security threats.

Classifications of white box testing

1. Static Testing : Static Testing is a type of a software testing method which is performed to check the defects in software without actually executing the code of the software application. Whereas in Dynamic Testing checks the code is executed to detect the defects

Static testing is done to avoid errors at an early stage of development as it is easier to identify the errors and solve the errors. It also helps finding errors that may not be found by Dynamic Testing
Static Testing, a software testing technique in which the software is tested without executing the code. It has two parts as listed below:

- Review - Typically used to find and eliminate errors or ambiguities in documents such as requirements, design, test cases, etc.
- Static analysis - The code written by developers are analysed (usually by tools) for structural defects that may lead to defects.
- Static testing involves manual or automated reviews of the documents. This review is done during an initial phase of testing to catch Defect early in STLC. It examines work documents and provides review comments. It is also called Non-execution testing or verification testing.

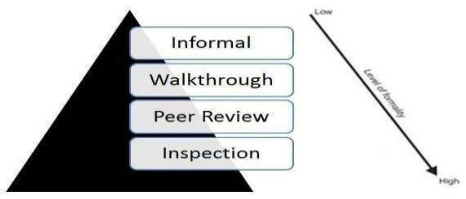
The two main types of static testing techniques are

- **Manual examinations**: Manual examinations include analysis of code done manually, also known as **REVIEWS.**
- **Automated analysis using tools:** Automated analysis are basically static analysis which is done using tools.

What is Tested in Static Testing
In Static Testing, following things are tested

- Unit Test Cases
- Business Requirements Document (BRD)
- Use Cases
- System/Functional Requirements
- Prototype
- Prototype Specification Document
- DB Fields Dictionary Spreadsheet
- Test Data
- Traceability Matrix Document
- User Manual/Training Guides/Documentation
- Test Plan Strategy Document/Test Cases
- Automation/Performance Test Scripts

The types of reviews can be given by a simple diagram:



1.Informal Reviews: This is one of the type of review which doesn't follow any process to find errors in the document. Under this technique, you just review the document and give informal comments on it.

2. **Walkthrough** -- the author of whichever document is being reviewed will explain the document to their team. Participants will ask questions, and any notes are written down.
The walkthrough can be formal or informal review.
Team member does not need to have detailed knowledge of the content as the author is well prepared for that and it is kind of knowledge transfer session .
Main objective is to enable learning and giving knowledge to other team members about the content.

2)Inspection -- a designated moderator will conduct a strict review as a process to find defects.

- inspection is one of the most formal kinds of Reviews.
- It is led by a trained Moderator who is not the author of the meeting.

- Reviewers are well prepared before the meeting about the documents or what needs to be discussed.
- Rules and checklists are used in this meeting during which time the product is examined and defects are logged.
- Defects found in the meeting are documented in the issue log or logging list.
- Meeting has proper entry and exit criteria.
- Reports created during the meeting are shared with the Author to take appropriate actions on that.

3)Technical reviews -- technical specifications are reviewed by peers in order to detect any errors.

- it is well documented and follows defect detection technique which involves peers and technical experts
- It is usually led by a trained Moderator and not the Author.
- In Technical Review, the product is examined and the defects are found which are mainly technical ones.
- No management participation is there in Technical Review.
- The full report is prepared to have a list of issues addressed.
  A team consisting of your peers, review the technical specification of the software product and checks whether it is suitable for the project. They try to find any discrepancies in the specifications and standards followed. This review concentrates mainly on the technical documentation related to the software such as Test Strategy, Test Plan and requirement specification documents.

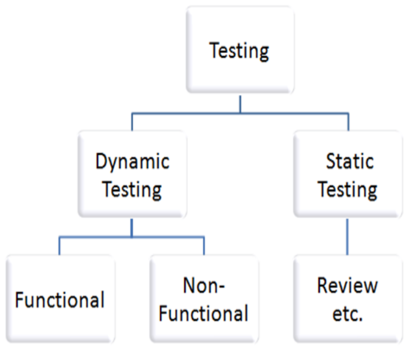
Summary:

- Static testing is to find defects as early as possible.
- Static testing not a substitute for dynamic testing, both find a different type of defects
- Reviews are an effective technique for Static Testing
- Reviews not only help to find defects but also understand missing requirements, design defects, non-maintainable code.

What is Dynamic Testing?

Under **Dynamic Testing**, a code is executed. It checks for functional behavior of software system, memory/cpu usage and overall performance of the system. Hence the name "Dynamic"

- The main objective of this testing is to confirm that the software product works in conformance with the business requirements. This testing is also called an Execution technique or validation testing.
- Dynamic testing executes the software and validates the output with the expected outcome. Dynamic testing is performed at all levels of testing and it can be either black or white box testing.



Structural Testing:

- Structure-based testing, therefore, can be defined as a type of software testing that tests the code's structure and intended flows. **For example,** verifying the actual code for aspects like the correct implementation of conditional statements, and whether every statement in the code is correctly executed. It is also known as structure-based testing.

- To carry out this type of testing, we need to thoroughly understand the code. This is why this testing is usually done by the developers who wrote the code as they understand it best.

- **Structural testing** is the type of testing carried out to test the structure of code. It is also known as White Box testing or Glass Box testing. This type of testing requires knowledge of the code, so, it is mostly done by the developers. It is more concerned with how system does it rather than the functionality of the system. It provides more coverage to the testing. For ex, to test certain error message in an application, we need to test the trigger condition for it, but there must be many trigger for it. It is possible to miss out one while testing the requirements drafted in SRS. But using this testing, the trigger is most likely to be covered since structural testing aims to cover all the nodes and paths in the structure of code.

- **Structural testing** is a type of software testing which uses the internal design of the software for testing or in other words the software testing which is performed by the team which knows the development phase of the software, is known as structural testing.
- Structural testing is basically related to the internal design and implementation of the software i.e. it involves the development team members in the testing team. It basically tests different aspects of the software according to its types. Structural testing is just the opposite of behavioral testing.
- The knowledge of the code's internal executions and how the software is implemented is a necessity for the test engineer to implement the structural testing.
- Throughout the structural testing, the test engineer intends on how the software performs, and it can be used at all levels of testing.

The intention behind the testing process is finding out how the system works not the functionality of it. To be more specific, if an error message is popping up in an application there will be a reason behind it. Structural testing can be used to find that issue and fix it
Advantages of Structural 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 Structural Box Testing:
· Expensive as one has to spend both time and money to perform white box testing.
Every possibility that few lines of code is missed accidentally.
· In-depth knowledge about the programming language is necessary to perform white box testing.

Functional Testing

- FUNCTIONAL TESTING is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.
- Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.
- Functional Testing is a type of Software Testing in which the system is tested against the functional requirements and specifications. Functional testing ensures that the requirements or specifications are properly satisfied by the application. This type of testing is particularly concerned with the result of processing. It focuses on simulation of actual system usage but does not develop any system structure assumptions.

Functional Testing:
It is a type of software testing which is used to verify the functionality of the software application, whether the function is working according to the requirement specification. In functional testing, each

function tested by giving the value, determining the output, and verifying the actual output with the expected value. Functional testing performed as black-box testing which is presented to confirm that the functionality of an application or system behaves as we are expecting. It is done to verify the functionality of the application.
Functional testing also called as black-box testing, because it focuses on application specification rather than actual code. Tester has to test only the program rather than the system.

How to do Functional Testing
Following is a step by step process on How to do Functional Testing :

- ✓ Understand the Functional Requirements
- ✓ Identify test input or test data based on requirements
- ✓ Compute the expected outcomes with selected test input values
- ✓ Execute test cases
- ✓ Compare actual and computed expected results

Code Coverage :
Code coverage is a software testing metric that determines the number of lines of code that is successfully validated under a test procedure, which in turn, helps in analyzing how comprehensively a software is verified.
Code coverage is a software testing metric or also termed as a Code Coverage Testing which helps in determining how much code of the source is tested which helps in accessing quality of test suite and analyzing how comprehensively a software is verified. Actually in simple code coverage refers to the degree of which the source code of the software code has been tested. This Code Coverage is considered as one of the form of white box testing
As we know at last of the development each client wants a quality software product as well as the developer team is also responsible for delivering a quality software product to the customer/client. Where this quality refers to the product's performance, functionalities, behavior, correctness, reliability, effectiveness, security, and maintainability. Where Code Coverage metric helps in determining the performance and quality aspects of any software.
Code coverage is one such software testing metric that can help in assessing the test performance and quality aspects of any software.
Such an insight will equally be beneficial to the development and QA team. For developers, this metric can help in dead code detection and elimination. On the other hand, for QA, it can help to check missed or uncovered test cases. They can track the health status and quality of the source code while paying more heed to the uncaptured parts of the code.

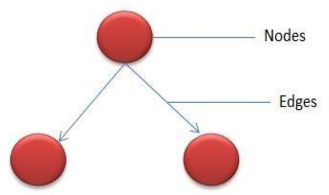
- Code Coverage testing is determining how much code is being tested. It can be calculated using the formula:
- Code Coverage = (Number of lines of code executed)/(Total Number of lines of code in a system component) * 100

Code Coverage Criteria
1.Statement coverage: how many of the statements in the program have been executed.
2.Branches coverage: how many of the branches of the control structures (if statements for instance) have been executed.
Condition coverage: how many of the boolean sub-expressions have been tested for a true and a false value.
3.Line coverage: how many of lines of source code have been tested.
These metrics are usually represented as the number of items actually tested, the items found in your code, and a coverage percentage (items tested / items found)
4.Function coverage: how many of the functions defined have been called.

Code Complexity Testing

- **Cyclomatic Complexity in Software Testing** is a testing metric used for measuring the complexity of a software program. It is a quantitative measure of independent paths in the source code of a software program. Cyclomatic complexity can be calculated by using control flow graphs or with respect to functions, modules, methods or classes within a software program.

- independent path is defined as a path that has at least one edge which has not been traversed before in any other paths.
- Cyclomatic Complexity is software metric useful for structured or White Box Testing. It is mainly used to evaluate complexity of a program. If the decision points are more, then complexity of the program is more. If program has high complexity number, then probability of error is high with increased time for maintenance and trouble shoot
- For example, if source code contains no control flow statement then its cyclomatic complexity will be 1 and source code contains a single path in it. Similarly, if the source code contains one **if condition** then cyclomatic complexity will be 2 because there will be two paths one for true and the other for false.
- This metric was developed by Thomas J. McCabe in 1976 and it is based on a control flow representation of the program. Control flow depicts a program as a graph which consists of Nodes and Edges
- In the graph, Nodes represent processing tasks while edges represent control flow between the nodes.



Uses of Cyclomatic Complexity:

- Cyclomatic Complexity can prove to be very helpful for developers and testers.
- Helps developers and testers to determine independent path executions
- Developers can assure that all the paths have been tested atleast once
- Helps us to focus more on the uncovered paths
- Improve code coverage in Software Engineering
- Evaluate the risk associated with the application or program
- Using these metrics early in the cycle reduces more risk of the program

Black box testing

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing. Black box testing is a type of software testing in which the functionality of the software is not known. The testing is done without the internal knowledge of the products.

- It is also known as specification based testing
- The black box testing is also known as an opaque, closed box, function-centric testing. It emphasizes on the behavior of the software. Black box testing checks scenarios where the system can break

Input
Black Box
Output

-

Types of Black Box Testing

There are many types of Black Box Testing but the following are the prominent ones -

- Functional testing - This black box testing type is related to the functional requirements of a system; it is done by software testers.

- Non-functional testing - This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- Regression testing – Regres is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Black Box Testing Techniques

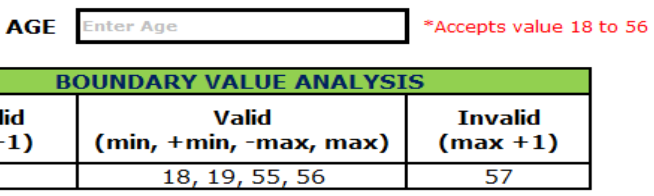
In order to systematically test a set of functions, it is necessary to design test cases. Testers can create test cases from the requirement specification document using the following Black Box Testing techniques.

- Equivalence Partitioning
- Boundary Value Analysis
- Decision Table Testing
- State Transition Testing
- Error Guessing
- Graph-Based Testing Methods
- Comparison Testing

1.Boundary Value Analysis(BVA)

- Boundary Value Technique is used to test boundary values, boundary values are those that contain the upper and lower limit of a variable. It tests, while entering boundary value whether the software is producing correct output or not.
- Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.
- Boundary value analysis is one of the widely used case design technique for black box testing. It is used to test boundary values because the input values near the boundary have higher chances of error.
- Whenever we do the testing by boundary value analysis, the tester focuses on, while entering boundary value whether the software is producing correct output or not.
- Boundary values are those that contain the upper and lower limit of a variable. Assume that, age is a variable of any function, and its minimum value is 18 and the maximum value is 30, both 18 and 30 will be considered as boundary values.
- The basic assumption of boundary value analysis is, the test cases that are created using boundary values are most likely to cause an error.
- There is 18 and 30 are the boundary values that's why tester pays more attention to these values, but this doesn't mean that the middle values like 19, 20, 21, 27, 29 are ignored. Test cases are developed for each and every value of the range.

Assume, we have to test a field which accepts Age 18 – 56


AGE
Enter Age
*Accepts value 18 to 56

BOUNDARY VALUE ANALYSIS		
Invalid (min -1)	Valid (min, +min, -max, max)	Invalid (max +1)
17	18, 19, 55, 56	57

- Minimum boundary value is 18
- Maximum boundary value is 56
- <u>Valid Inputs:</u> 18,19,55,56
- <u>Invalid Inputs:</u> 17 and 57
- Test case 1: Enter the value 17 (18-1) = Invalid
- Test case 2: Enter the value 18 = Valid
- Test case 3: Enter the value 19 (18+1) = Valid
- Test case 4: Enter the value 55 (56-1) = Valid
- Test case 5: Enter the value 56 = Valid
- Test case 6: Enter the value 57 (56+1) =Invalid

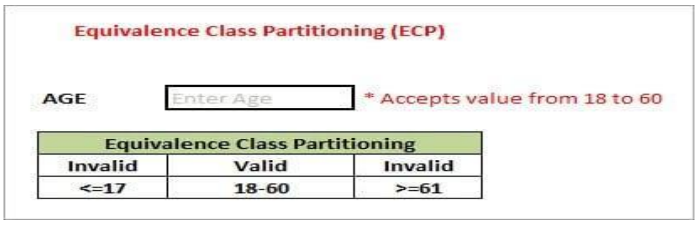
2. Equivalence partitioning

- Equivalence partitioning is a technique of software testing in which input data divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior.
- This technique is also known as Equivalence Class Partitioning (ECP). In this technique, input values to the system or application are divided into different classes or groups based on its similarity in the outcome.
- Hence, instead of using each and every input value we can now use any one value from the group/class to test the outcome. In this way, we can maintain the test coverage while we can reduce a lot of rework and most importantly the time spent.
- As present in the above image, an "AGE" text field accepts only the numbers from 18 to 60. There will be three sets of classes or groups.

Example:

Two invalid classes will be:

- a) Less than or equal to 17.
- b) Greater than or equal to 61.
- One valid class will be anything between 18 to 60.
- We have thus reduced the test cases to only 3 test cases based on the formed classes thereby covering all the possibilities. So, testing with anyone value from each set of the class is sufficient to test the above scenario.



Equivalence Class Partitioning (ECP)

AGE [Enter Age] * Accepts value from 18 to 60

Equivalence Class Partitioning		
Invalid	Valid	Invalid
<=17	18-60	>=61

3. Decision Table

- Decision Table Technique is a systematic approach where various input combinations and their respective system behavior are captured in a tabular form. It is appropriate for the functions that have a logical relationship between two and more than two inputs.
- Decision Table is aka Cause-Effect Table. This test technique is appropriate for functionalities which has logical relationships between inputs (if-else logic). In Decision table technique, we deal with combinations of inputs. To identify the test cases with decision table, we consider conditions and actions. We take conditions as inputs and actions as outputs.
- In some instances, the inputs combinations can become very complicated for tracking several possibilities.
- Such complex situations rely on decision tables, as it offers the testers an organized view about the inputs combination and the expected output.

Let's understand it by an example:

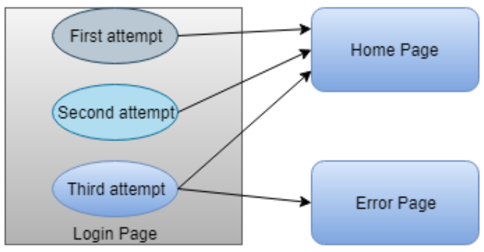
- Most of us use an email account, and when you want to use an email account, for this you need to enter the email and its associated password.
- If both email and password are correctly matched, the user will be directed to the email account's homepage; otherwise, it will come back to the login page with an error message specified with "Incorrect Email" or "Incorrect Password."
- Now, let's see how a decision table is created for the login function in which we can log in by using email and password. Both the email and the password are the conditions, and the expected result is action.

Email (condition1)	T	T	F	F
Password (condition2)	T	F	T	F
Expected Result (Action)	Account Page	Incorrect password	Incorrect email	Incorrect email

In the table, there are four conditions or test cases to test the login function. In the first condition if both email and password are correct, then the user should be directed to account's Homepage.
In the second condition if the email is correct, but the password is incorrect then the function should display Incorrect Password. In the third condition if the email is incorrect, but the password is correct, then it should display Incorrect Email.
Now, in fourth and last condition both email and password are incorrect then the function should display Incorrect Email.
In this example, all possible conditions or test cases have been included, and in the same way, the testing team also includes all possible test cases so that upcoming bugs can be cured at testing level.

4. State Transition Technique

- The general meaning of state transition is, different forms of the same situation, and according to the meaning, the state transition method does the same. It is used to capture the behavior of the software application when different input values are given to the same function.
- State Transition Testing is a technique that is used to test the different states of the system under test. The state of the system changes depending upon the conditions or events. The events trigger states which become scenarios and a tester needs to test them.
- .In some systems, significant responses are generated when the system transitions from one state to another. A common example is a login mechanism which allows users to authenticate, but after a specific number of login attempts, transition to a different state, locking the account.
- If testers identify a state transition mechanism, they can design test cases that probe the system when it transitions states. For example, for a system that locks the account after five failed login attempts, a test case can check what happens at the sixth login attempt.
- We all use the ATMs, when we withdraw money from it, it displays account details at last. Now we again do another transaction, then it again displays account details, but the details displayed after the second transaction are different from the first transaction, but both details are displayed by using the same function of the ATM. So the same function was used here but each time the output was different, this is called state transition. In the case of testing of a software application, this method tests whether the function is following state transition specifications on entering different inputs
- This applies to those types of applications that provide the specific number of attempts to access the application such as the login function of an application which gets locked after the specified number of incorrect attempts. Let's see in detail, in the login function we use email and password, it gives a specific number of attempts to access the application, after crossing the maximum number of attempts it gets locked with an error message.
- There is a login function of an application which provides a maximum three number of attempts, and after exceeding three attempts, it will be directed to an error page.



STATE	LOGIN	VALIDATION	REDIRECTED
S1	First Attempt	Invalid	S2
S2	Second Attempt	Invalid	S3
S3	Third Attempt	Invalid	S5
S4	Home Page		
S5	Error Page		

In the above state transition table, we see that state S1 denotes first login attempt. When the first attempt is invalid, the user will be directed to the second attempt (state S2). If the second attempt is also invalid, then the user will be directed to the third attempt (state S3). Now if the third and last attempt is invalid, then the user will be directed to the error page (state S5).
But if the third attempt is valid, then it will be directed to the homepage (state S4).

5. 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.

- The requirements-based testing (RBT) process is comprised of two phases: **ambiguity reviews and cause-effect graphing.**
- An ambiguity review is a technique for identifying ambiguities in functional requirements to improve the quality of those requirements.
- Cause-effect graphing is a test-case design technique that derives the minimum number of test cases to cover 100 percent of the functional requirements.
- It employs many methods like **creating graphs for cause and effect**, **analyzing test conditions** and **analyzing ambiguities**. Usually a list of defects in the requirements document is used to detect requirements ambiguities and then remove them.
- To determine **test conditions** that must be covered, one must make an in-depth **study of the requirements document**.

Stages in Requirements based Testing:

1.Defining Test Completion Criteria - Testing is completed only when all the functional and non-functional testing is complete.

2.Design Test Cases - A Test case has five parameters namely the initial state or precondition, data setup, the inputs, expected outcomes and actual outcomes.

3.Execute Tests - Execute the test cases against the system under test and document the results.

4.Verify Test Results - Verify if the expected and actual results match each other.

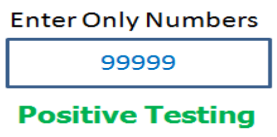
5.Verify Test Coverage - Verify if the tests cover both functional and non-functional aspects of the requirement.

6. 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.

6. Positive Testing

Positive Testing is a type of testing which is performed on a software application by providing the valid data sets as an input. It checks whether the software application behaves as expected with positive inputs or not. Positive testing is performed in order to check whether the software application does exactly what it is expected to do.

- There is a text box in an application which can accept only numbers. Entering values up to 99999 will be acceptable by the system and any other values apart from this should not be acceptable. To do positive testing, set the valid input values from 0 to 99999 and check whether the system is accepting the values.

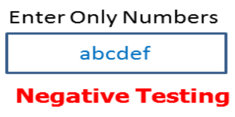


Positive Testing is testing process where the system is validated against the valid input data. In this testing, tester always check for only valid set of values and check if a application behaves as expected with its expected inputs. The main intention of this testing is to check whether software application does that what it is supposed to do. Positive Testing always tries to prove that a given product and project always meets the requirements and specifications. Positive testing is testing of the normal day to day life scenarios and to check the expected behavior of application.

Example of Positive Testing:Consider a scenario where you want to test an application which contains a simple text box to enter age and requirements say that it should take only numerical values. So here provide only positive numerical values to check whether it is working as expected or not is the Positive Testing.

7. Negative Testing

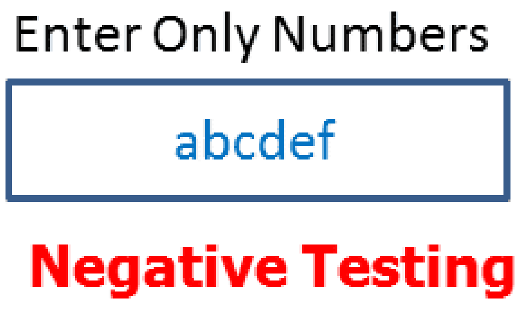
- **Negative Testing** is a testing method performed on the software application by providing invalid or improper data sets as input. It checks whether the software application behaves as expected with the negative or unwanted user inputs. The purpose of negative testing is to ensure that the software application does not crash and remains stable with invalid data inputs.
- For example -
- Negative testing can be performed by entering characters A to Z or from a to z. Either software system should not accept the values or else it should throw an error message for these invalid data inputs.



In Negative Testing the system is validated by providing invalid data as input. A negative test checks if an application behaves as expected with its negative inputs. This is to test the application that does not do anything that it is not supposed to do so. Such testing is to be carried out keeping negative point of view & only execute the test cases for only invalid set of input data.

The main reason behind Negative testing is to check the stability of the software application against the influences of different variety of incorrect validation data set. Negative testing helps to find more defects & improve the quality of the software application under test but it should be done once the positive testing is complete.

Example of Negative Testing : Considering example as we know phone no field accepts only numbers and does not accept the alphabets and special characters but if we type alphabets and special characters on phone number field to check it accepts the alphabets and special characters or not than it is negative testing.



Here, expectation is that the text box will not accept invalid values and will display an error message for the wrong entry.

In both the testing, the following needs to be considered:

- Input data
- An action which needs to be performed
- Output Result

Testing Technique used for Positive and Negative Testing:

Following techniques are used for Positive and negative validation of testing is:

- Boundary Value Analysis
- Equivalence Partitioning

8. User Documentation Testing

User Documentation covers all the manuals, user guides, installation guides, setup guides, read me files, software release notes, and online help that are provided along with the software to help the end user to understand the software system.

- User Documentation Testing should have two objectives:-
  1) To check if what is stated in the document is available in the software
  2) To check if what is there in the product is explained correctly in the document
- This testing is plays a vital role as the users will refer this document when they start using the software at their location. a badly written document can put off a user and bias them against the product even the product offers rich functionality.
- Defects found in the user documentation need to be tracked to closure like any regular software defect. Because these documents are the first interactions the users have with the product. A good User Documentation aids in reducing customer support calls. The effort and money spend on this effort would form a valuable investment in the long run for the organization.
- A good observation is that projects that have all the documents in place have a high level of maturity as compared to the un-documented project.
- Documentation for an organization saves time, cost and makes testing easy and systematic.
- It is equally important for the client's acceptance because documentation defines a software product's effectiveness. If the documentation is poor, deficient, or defective, it may affect the quality of software or application. QA practices should be documented such that they are repeatable, and are not dependent on any individuals.
- During manual software testing, documentation will include specifications, test designs, test plan, prevalent business rules, reports, configurations details, changes in code, test cases, bug reports, user manuals, etc.
- As a part of documentation, there needs to be a system for easily finding and obtaining documents and determining what documentation will have a particular piece of information.
- Once the details are documented, they should be placed at a common databank where easy search and timely availability of the records is feasible. These documents come handy in times of any dispute or comparing the requirement specification with the delivered product.

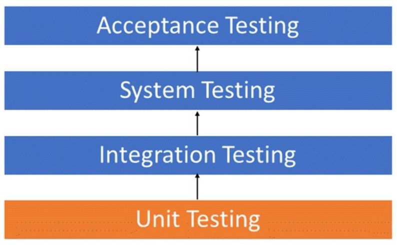
Few essential software testing documents that need to be used and maintained on a day to day basis:

1) Test design document
2) Test case specification
3) Test Strategy
4) Test summary reports
5) Document of Weekly Status Report
6) User Documents
7) Document of User Acceptance Report
8) Report of Risk Assessment
9) Test Log document
10) Test plan document
11) Bug reports document
12) Test data document
13) Test analysis

Unit 03 Levels of testing and special tests

UNIT TESTING :

- Unit testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.
- Unit testing is a White Box testing technique that is usually performed by the developer.
- 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 involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance.
- The purpose of unit testing is to test the correctness of isolated code. A unit component is an individual function or code of the application. White box testing approach used for unit testing and usually done by the developers.
- Whenever the application is ready and given to the Test engineer, he/she will start checking every component of the module or module of the application independently or one by one, and this process is known as Unit testing or components testing.
- **UNIT TESTING, also known as COMPONENT TESTING**, first is a level of software testing where individual units / components of a software are tested. The purpose is to validate that each unit of the software performs as designed.
- In a testing level hierarchy, unit testing is the first level of testing done before integration and other remaining levels of the testing.



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 Disadvantages

- Unit testing can't be expected to catch every error in a program. It is not possible to evaluate all execution paths even in the most trivial programs
- Unit testing by its very nature focuses on a unit of code. Hence it can't catch integration errors or broad system level errors.

Objective of Unit Testing:

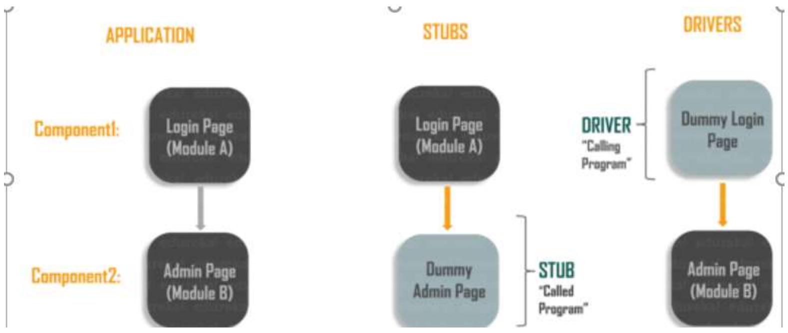
The objective of Unit Testing is:

1. To isolate a section of code.
2. To verify the correctness of code.
3. To test every function and procedure.
4. To fix bug early in development cycle and to save costs.
5. To help the developers to understand the code base and enable them to make changes quickly.
6. To help for code reuse.

Stubs and Drivers

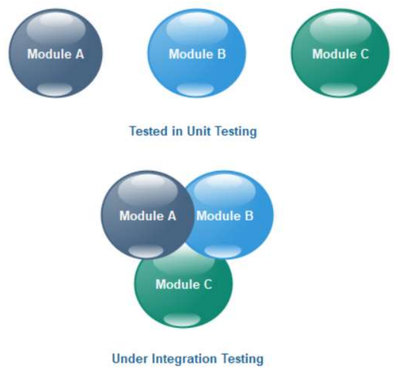
- Stubs and Drivers are the dummy programs in Integration testing used to facilitate the software testing activity. These programs act as a substitutes for the missing modules in the testing. They do not implement the entire programming logic of the software module but they simulate data communication with the calling module while testing.
- Stub: Is called by the Module under Test.
- Driver: Calls the Module to be tested.

Before we start discussing the types of integration testing available, we need to understand the concept of stubs and drivers. While testing, sometimes we face a situation where some of the modules are still under development. These modules for testing purpose are replaced with some dummy programs. These dummy programs are called stubs and drivers.
Imagine, we have an application with two modules i.e, Login Page(Module A) and Admin Page(Module B).
Case1: You have to test the Login Page which is developed and sent to the testing team. Login Page is dependent on Admin Page. But the Admin Page is not ready yet. To overcome this situation developers write a dummy program which acts as an Admin Page. This dummy program is Stub. Stubs are 'Called Programs'.
Case2: You have to test Admin Page but the Login Page is not ready yet. To overcome this situation developers write a dummy program which acts like the Login Page. This dummy program is Driver. Drivers are 'Calling programs'.



INTEGRATION TESTING

- INTEGRATION TESTING is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.
- Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.
- Unit testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules.
- Integration testing is the process of testing the interface between two software units or module. It's focus on determining the correctness of the interface. The purpose of the integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing is performed.
- Once all the components or modules are working independently, then we need to check the data flow between the dependent modules is known as integration testing.



Reason behind Integration Testing

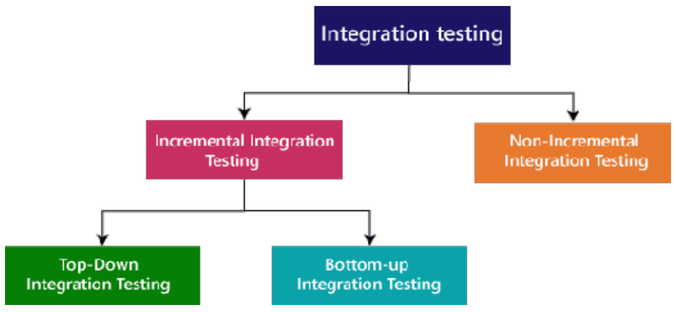
Although all modules of software application already tested in unit testing, errors still exist due to the following reasons:

1. Each module is designed by individual software developer whose programming logic may differ from developers of other modules so; integration testing becomes essential to determine the working of software modules.
2. To check the interaction of software modules with the database whether it is an erroneous or not.
3. Requirements can be changed or enhanced at the time of module development. These new requirements may not be tested at the level of unit testing hence integration testing becomes mandatory.
4. Incompatibility between modules of software could create errors.

Types of Integration Testing

Integration testing can be classified into two parts:

- Incremental integration testing
- Non-incremental integration testing


Integration testing
Incremental Integration Testing
Non-Incremental Integration Testing
Top-Down Integration Testing
Bottom-up Integration Testing

When we speak about the types of integration testing, we usually mean different approaches

- Big-Bang Integration (non-incremental integration)
- Incremental Testing: which is further divided into the following
- Top Down Integration
- Bottom Up Integration
- Sandwich/ Hybrid Integration

1. **Big Bang Integration Testing   (non-incremental integration)**

- In this testing approach, once all the modules are developed and tested individually, they are integrated once and tested together at once. The only advantage of this type of testing is that it is very much suitable for smaller systems.
- We will go for this method, when the data flow is very complex and when it is difficult to find who is a parent and who is a child. And in such case, we will create the data in any module bang on all other existing modules and check if the data is present. Hence, it is also known as the Big bang method.
- In this approach, testing is done via integration of all modules at once. It is convenient for small software systems, if used for large software systems identification of defects is difficult.

- Since this testing can be done after completion of all modules due to that testing team has less time for execution of this process so that internally linked interfaces and high-risk critical modules can be missed easily.

Advantages:

- It is convenient for small size software systems.

Disadvantages:

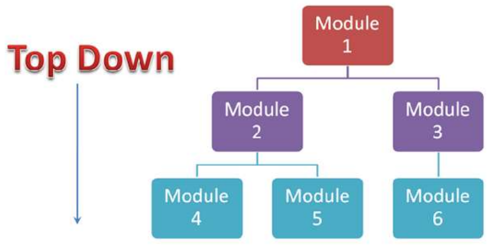
- Identification of defects is difficult because finding the error where it came from is a problem, and we don't know the source of the bug.
- Small modules missed easily.
- Time provided for testing is very less.
- We may miss to test some of the interfaces.

2. Incremental integration Testing

- Incremental Testing is performed by connecting two or more modules together that are logically related. Later more modules are added and tested for proper functionality. This is done until all the modules are integrated and tested successfully.
- In this type of testing, there is a strong relationship between the dependent modules. Suppose we take two or more modules and verify that the data flow between them is working fine. If it is, then add more modules and test again.
- It's further divided into Top-Down Approach, Bottom-Up Approach, and Sandwich Approach.

a) Top-Down Integration Testing

- The top-down approach starts by testing the top-most modules and gradually moving down to the lowest set of modules one-by-one. Testing takes place from top to down following the control flow of the software system. As there is a possibility that the lower level modules might not have been developed while top modules are tested, we use stubs instead of those not ready modules. For simple applications, stubs would simply return the control to their superior modules.
- The top-down testing strategy deals with the process in which higher level modules are tested with lower level modules until the successful completion of testing of all the modules. Major design flaws can be detected and fixed early because critical modules tested first. In this type of method, we will add the modules incrementally or one by one and check the data flow in the same order.



Advantages:

- Fault localization is easier
- The test product is extremely consistent
- The stubs can be written in lesser time compared to drivers
- Critical modules are tested on priority
- Major design flaws are detected as early as possible

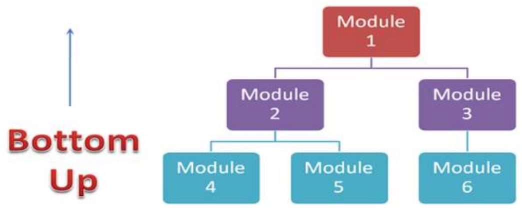
Disadvantages

- Requires several stubs
- Poor support for early release
- Basic functionality is tested at the end of the cycle

b) Bottom-Up Integration Testing

- The bottom-up approach starts with testing the lowest units of the application and gradually moving up one-by-one. Here testing takes place from the bottom of the control flow to upwards. Again it's possible that the higher level modules might not have been developed by the time lower modules are tested. In such cases, we simulate the functionality of missing modules by using drivers. These drivers perform a range of tasks such as invoking module under test, pass test data or receive output data.
- The bottom to up testing strategy deals with the process in which lower level modules are tested with higher level modules until the successful completion of testing of all the modules. Top level critical

modules are tested at last, so it may cause a defect. Or we can say that we will be adding the modules from bottom to the top and check the data flow in the same order.



Advantages

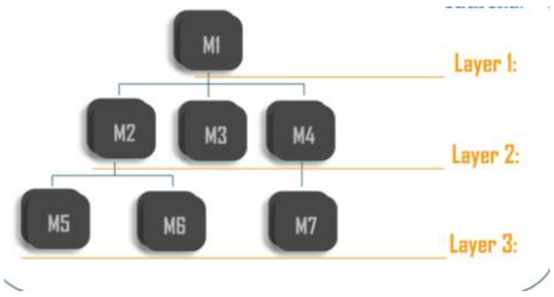
- Here development & testing can be done together so the product will be efficient
- Test conditions are much easy to create

Disadvantages

- Requires several drivers
- Data flow is tested very late
- Need for drivers complicates test data management
- Poor support for early release
- Key interfaces defects are detected late

c) Sandwich Integration Testing/ Hybrid/ Bidirectional Testing

- To overcome the limitations and to exploit the advantages of top-down and bottom-up approaches, a hybrid approach of integration testing is used. This approach is known as sandwich integration testing or mixed integration testing. Here, the system is viewed as three layers. Main target layer in the middle, another layer above the target layer, and the last layer below the target layer. The top-down approach is used on the layer from the top to the middle layer. The bottom-up approach is used on the layer from the bottom to middle.
- in this approach, both Top-Down and Bottom-Up approaches are combined for testing. In this process, top-level modules are tested with lower level modules and lower level modules tested with high-level modules simultaneously. There is less possibility of occurrence of defect because each module interface is tested.
- Sandwich Testing is a strategy in which top level modules are tested with lower level modules at the same time lower modules are integrated with top modules and tested as a system. It is a combination of Top-down and Bottom-up approaches therefore it is called Hybrid Integration Testing. It makes use of both stubs as well as drivers



Advantages

- Top-Down and Bottom-Up testing techniques can be performed in parallel or one after the other
- Very useful for large enterprises and huge projects that further have several subprojects

Disadvantages

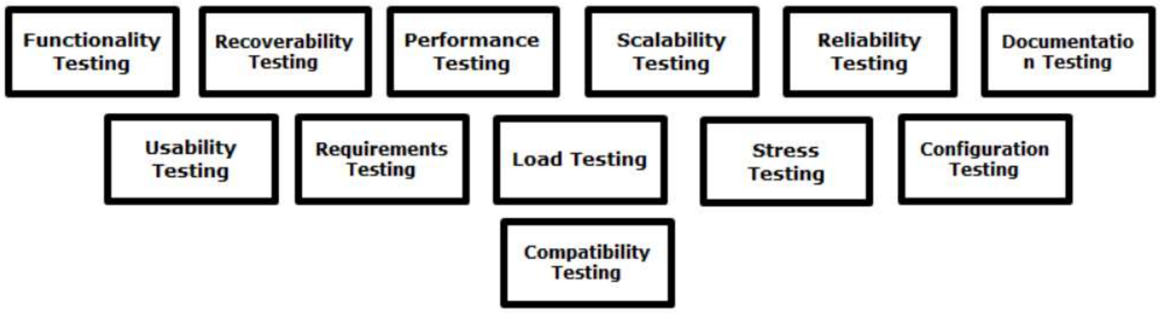
- The cost requirement is very high
- Cannot be used for smaller systems with huge interdependence between the modules
- Different skill sets are required for testers at different levels
- System Testing

SYSTEM TESTING

SYSTEM TESTING is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

- system Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.
- In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested.
- System testing is defined as testing of a complete and fully integrated software product. This testing falls in black-box testing wherein knowledge of the inner design of the code is not a pre-requisite and is done by the testing team.
- System testing is performed in the context of a System Requirement Specification (SRS) and/or a Functional Requirement Specifications (FRS). It is the final test to verify that the product to be delivered meets the specifications mentioned in the requirement document. It should investigate both functional and non-functional requirements.
  There are various types of system testing and the team should choose which ones they would need before application deployment.

Types of System Testing



a) Usability Testing

- Usability Testing also known as User Experience(UX) Testing, is a testing method for measuring how easy and user-friendly a software application is. A small set of target end-users, use software application to expose usability defects. Usability testing mainly focuses on user's ease of using application, flexibility of application to handle controls and ability of application to meet its objectives.
- This testing is recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.
- It is type of non functional testing
- To make sure that the system is easy to use, learn and operate
- Usability testing is testing, which checks the defect in the end-user interaction of software or the product.
- It is also known as User Experience (UX) Testing.
- Checking the user-friendliness, efficiency, and accuracy of the application is known as Usability Testing."
- USABILITY TESTING is a type of software testing done from an end-user's perspective to determine if the system is easily usable. It falls under non-functional testing.
- It is a wide testing where we need to have an application knowledge.
- When we use usability testing, it makes sure that the developed software is easy while using the system without facing any problem and makes end-user life easier.
- Usability testing is testing, which checks the defect in the end-user interaction of software or the product.
- Here, the user-friendliness can be described in many aspects, such as:
- Easy to understand
- Easy to access
- Look & feel
- Faster to Access
- Effective Navigation
- Good Error Handling

b) Regression Testing

- REGRESSION TESTING is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.
- Regression Testing is nothing but a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.
- This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that the old code still works once the latest code changes are done.

c) Performance Testing

- Performance Testing is a software testing process used for testing the speed, response time, stability, reliability, scalability and resource usage of a software application under particular workload. The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application. It is a subset of performance engineering and also known as “Perf Testing”.
- Performance testing will determine whether their software meets speed, scalability and stability requirements under expected workloads. Applications sent to market with poor performance metrics due to nonexistent or poor performance testing are likely to gain a bad reputation and fail to meet expected sales goals.
- The focus of Performance Testing is checking a software program's
- Speed - Determines whether the application responds quickly
- Scalability - Determines maximum user load the software application can handle.
- Stability - Determines if the application is stable under varying loads

d) Load Testing

- Load Testing is a non-functional software testing process in which the performance of software application is tested under a specific expected load. It determines how the software application behaves while being accessed by multiple users simultaneously. The goal of Load Testing is to improve performance bottlenecks and to ensure stability and smooth functioning of software application before deployment.
- Load testing examines how the system behaves during normal and high loads and determines if a system, piece of software, or computing device can handle high loads given a high demand of end-users.
- This testing usually identifies -
- The maximum operating capacity of an application
- Determine whether the current infrastructure is sufficient to run the application
- Sustainability of application with respect to peak user load
- Number of concurrent users that an application can support, and scalability to allow more users to access it.
- It is a type of non-functional testing. In Software Engineering, Load testing is commonly used for the Client/Server, Web-based applications - both Intranet and Internet.

e) Stress Testing

- Stress Testing is a type of software testing that verifies stability & reliability of software application. The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn't crash under crunch situations. It even tests beyond normal operating points and evaluates how software works under extreme conditions.
- The application under testing will be stressed when 5GB data is copied from the website and pasted in notepad. Notepad is under stress and gives 'Not Responded' error message.
- Stress testing is also extremely valuable for the following reasons:
- To check whether the system works under abnormal conditions.
- Displaying appropriate error message when the system is under stress.
- System failure under extreme conditions could result in enormous revenue loss
- It is better to be prepared for extreme conditions by executing Stress Testing.
- The goal of stress testing is to analyze the behavior of the system after a failure. For stress testing to be successful, a system should display an appropriate error message while it is under extreme conditions.
- stress testing. It refers to the testing of the software in determining whether its performance is satisfactory under extreme load conditions or not.

- It is a type of non-functional testing
- It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results
- It is a form of software testing that is used to determine the stability of a given system
- The main purpose of stress testing is to make sure that the system recovers after failure which is called as recoverability.

f) Recovery Testing

- Recovery testing is non -functional testing that determines the capability of the software to recover from failures such as software/hardware crashes or any network failures.
- Recovery testing is done in order to check how fast and better the application can recover after it has gone through any type of crash or hardware failure etc.
- To perform recovery testing software/hardware is forcefully failed to verify
- If recovery is successful or not.
- Whether the further operations of the software can be performed or not.
- The duration it will take to resume the operations.
- Lost data can be recovered completely or not.
- Percentage of scenarios in which the system can recover back.
- Before this testing is performed, backup is taken and saved to a secured location to avoid any data loss in case data is not recovered back successfully.
- Common failures that should be tested for recovery:
    1. Network issue
    2. Power failure
    3. External server not reachable
    4. Server not responding
    5. dll file missing
    6. Database overload
    7. Stopped services
    8. Physical conditions
    9. External device not responding
    10. Wireless network signal loss
- Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is properly performed.
- For example: When an application is receiving data from a network, unplug the connecting cable. After some time, plug the cable back in and analyze the application’s ability to continue receiving data from the point at which the network connection was broken.

g) Compatibility Testing

- Compatibility Testing is a type of Software testing to check whether your software is capable of running on different hardware, operating systems, applications, network environments or mobile devices.
- Compatibility Testing is a type of Non-functional testing
- Let's look into compatibility testing types
- Hardware: It checks software to be compatible with different hardware configurations.
- Operating Systems: It checks your software to be compatible with different Operating Systems like Windows, Unix, Mac OS etc.
- Software: It checks your developed software to be compatible with other software. For example, MS Word application should be compatible with other software like MS Outlook, MS Excel, VBA etc.
- Network: Evaluation of performance of a system in a network with varying parameters such as Bandwidth, Operating speed, Capacity. It also checks application in different networks with all parameters mentioned earlier.
- Browser: It checks the compatibility of your website with different browsers like Firefox, Google Chrome, Internet Explorer etc.
- Devices: It checks compatibility of your software with different devices like USB port Devices, Printers and Scanners, Other media devices and Blue tooth.
- Mobile: Checking your software is compatible with mobile platforms like Android, iOS etc.

- Versions of the software: It is verifying your software application to be compatible with different versions of the software. For instance checking your Microsoft Word to be compatible with Windows 7, Windows 7 SP1, Windows 7 SP2, Windows 7 SP3.

h) Security Testing

- Security Testing is a type of software testing that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders. It ensures that the software system and application are free from any threats or risks that can cause a loss. Security testing of any system is focuses on finding all possible loopholes and weaknesses of the system which might result into the loss of information or repute of the organization.
- Security testing is an integral part of software testing, which is used to discover the weaknesses, risks, or threats in the software application and also help us to stop the nasty attack from the outsiders and make sure the security of our software applications.
- The primary objective of security testing is to find all the potential ambiguities and vulnerabilities of the application so that the software does not stop working. If we perform security testing, then it helps us to identify all the possible security threats and also help the programmer to fix those errors.
- It is a testing procedure, which is used to define that the data will be safe and also continue the working process of the software.
- Goal of Security Testing:
- The goal of security testing is to:
- To identify the threats in the system.
- To measure the potential vulnerabilities of the system.
- To help in detecting every possible security risks in the system.
- To help developers in fixing the security problems through coding.

Acceptance Testing

- Acceptance Testing is a method of software testing where a system is tested for acceptability. The major aim of this test is to evaluate the compliance of the system with the business requirements and assess whether it is acceptable for delivery or not.
- Acceptance Testing is the last phase of software testing performed after System Testing and before making the system available for actual use.
- Acceptance testing is formal testing based on user requirements and function processing. It determines whether the software is conforming specified requirements and user requirements or not. It is conducted as a kind of Black Box testing where the number of required users involved testing the acceptance level of the system. It is the fourth and last level of software testing.
- **User acceptance testing (UAT)** is a type of testing, which is done by the customer before accepting the final product. Generally, UAT is done by the customer (domain expert) for their satisfaction, and check whether the application is working according to given business scenarios, real-time scenarios.
- In this, we concentrate only on those features and scenarios which are regularly used by the customer or mostly user scenarios for the business or those scenarios which are used daily by the end-user or the customer.
- However, the software has passed through three testing levels (Unit Testing, Integration Testing, System Testing) But still there are some minor errors which can be identified when the system is used by the end user in the actual scenario.

Use of Acceptance Testing:

- To find the defects missed during the functional testing phase.
- How well the product is developed.
- A product is what actually the customers need.
- Feedbacks help in improving the product performance and user experience.
- Minimize or eliminate the issues arising from the production.

Acceptance criteria

- the acceptance criteria consists of various predefined requirements and conditions, which are required to be met and accomplished, to make the software acceptable for end users and customers.

Acceptance criteria validates the development of the software as well as ensures that it fulfills its expected purpose, without any hindrance or issue.

- In short, acceptance criteria can be termed as a validation technique, which ascertains every aspect of the functional and nonfunctional requirements of the software and ensures that it meets the specified acceptance criteria accurately.
- Acceptance Criteria are conditions in which a software application should satisfy to be accepted by a user or customer. It mentions the defined standards of a software product must meet. These are a set of rules which cover the system behavior and from which we can make acceptance scenarios.
- Acceptance Criteria is a set of statements which mentions the result that is pass or fail for both functional and non-functional requirements of the project at the current stage. These functional and non-functional requirements are the conditions that can be accepted. There is no partial acceptance in acceptance criteria, it is is either passed or failed.
- This aims to solve the problem by considering the problem from a customer's point of view so therefore it must be written in the context of how a user actually experiences any particular application. It is about defining the user stories by considering all the predefined requirements of the customer. They decide that what are the user requirements and the scope of software application that needs to be completed via developers by taking into consideration the user story.

Example of Acceptance Criteria:
User Story: Creation of orders in online shopping cart
Criteria:

1. User should be able to selects multiple items and add to shopping cart.
2. The user should be able to see the items in the shopping cart.
3. The user should be able to purchase items using their local currency.
4. The user should be able to see an order number when the payment method is made.

Example
User story: As a user, I want to use a search field to type a city, name, or street, so that I could find matching hotel options.
Basic search interface acceptance criteria

- The search field is placed on the top bar
- Search starts once the user clicks "Search"
- The field contains a placeholder with a grey-colored text: "Where are you going?"
- The placeholder disappears once the user starts typing
- Search is performed if a user types in a city, hotel name, street, or all combined
- Search is in English, French, German, and Ukrainian
- The user can't type more than 200 symbols
- The search doesn't support special symbols (characters). If the user has typed a special symbol, show the warning message: "Search input cannot contain special symbols."

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

<u>Alpha Testing</u>

- Alpha Testing is a type of acceptance testing; performed to identify all possible issues and bugs before releasing the final product to the end users. Alpha testing is carried out by the testers who are internal employees of the organization. The main goal is to identify the tasks that a typical user might perform and test them.

- Alpha testing is performed at developer's site.

Beta Testing

- Beta Testing is performed by "real users" of the software application in "real environment" and it can be considered as a form of external user acceptance testing. It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of Beta Testing. This testing helps to test products in customer's environment.
- Beta testing is performed at end-user of the product.
- Testing is performed at client site.

Alpha Testing Vs Beta testing:

Following are the differences between Alpha and Beta Testing:

Alpha Testing	Beta Testing
Alpha testing performed by Testers who are usually internal employees of the organization	Beta testing is performed by Clients or End Users who are not employees of the organization
Alpha Testing performed at developer's site	Beta testing is performed at a client location or end user of the product
Reliability and Security Testing are not performed in-depth Alpha Testing	Reliability, Security, Robustness are checked during Beta Testing
Alpha testing involves both the white box and black box techniques	Beta Testing typically uses Black Box Testing
Alpha testing requires a lab environment or testing environment	Beta testing doesn't require any lab environment or testing environment. The software is made available to the public and is said to be real time environment
Long execution cycle may be required for Alpha testing	Only a few weeks of execution are required for Beta testing
Critical issues or fixes can be addressed by developers immediately in Alpha testing	Most of the issues or feedback is collected from Beta testing will be implemented in future versions of the product
Alpha testing is to ensure the quality of the product before moving to Beta testing	Beta testing also concentrates on the quality of the product, but gathers users input on the product and ensures that the product is ready for real time users.

Special Tests

1.Smoke Testing

Smoke Testing is a software testing process that determines whether the deployed software build is stable or not. Smoke testing is a confirmation for QA team to proceed with further software testing. It consists of a minimal set of tests run on each build to test software functionalities. Smoke testing is also known as "Build Verification Testing" or “Confidence Testing.”

Smoke Testing is a software testing technique performed post software build to verify that the critical functionalities of software are working fine. It is executed before any detailed functional or regression tests are executed. The main purpose of smoke testing is to reject a software application with defects so that QA team does not waste time testing broken software application.

The smoke tests qualify the build for further formal testing. The main aim of smoke testing is to detect early major issues. Smoke tests are designed to demonstrate system stability and conformance to requirements.

2. Sanity testing

Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, 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. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing

<u>Features of Sanity Testing:</u>

1. Subset of Regression Testing:
Sanity testing is a subset of regression testing and focuses on the smaller section of the application.
2. Unscripted:
Most of the times sanity testing is not scripted.
3. Not documented:
Usually sanity testing is undocumented.
4. Narrow and deep:
Sanity testing is narrow and deep approach of testing where limited functionalities are covered deeply.
5. Performed by testers:
Sanity testing is normally performed by testers.

In other words, we can say that sanity testing is performed to make sure that all the defects have been solved and no added issues come into the presence because of these modifications.

<u>Benefits of Sanity Testing</u>

1. Very simple to understand and execute, yet efficient in finding bugs
2. Detects errors and defects in the early stages of software development
3. For sanity testing, there is no need for scripting and documentation
4. Sanity testing helps identify any deployment or compilation issues
5. Sanity testing saves unnecessary testing effort and time
6. Less cost expensive when compared to other types of testing

<u>3. GUI Testing</u>

GUI Testing is a software testing type that checks the Graphical User Interface of the Software. The purpose of Graphical User Interface (GUI) Testing is to ensure the functionalities of software application work as per specifications by checking screens and controls like menus, buttons, icons, etc.
Graphical User Interface Testing (GUI) Testing is the process for ensuring proper functionality of the graphical user interface (GUI) for a specific application. GUI testing generally evaluates a design of elements such as layout, colors and also fonts, font sizes, labels, text boxes, text formatting, captions, buttons, lists, icons, links and content. GUI testing processes may be either

manual or automatic and are often performed by third -party companies, rather than developers or end users.

What do you Check-in GUI Testing?
The following checklist will ensure detailed GUI Testing in Software Testing.

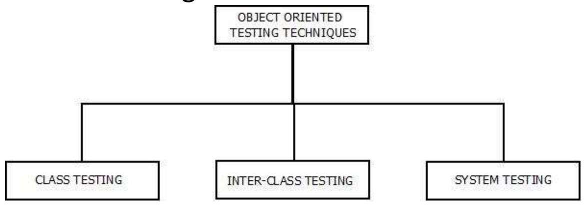
- Check all the GUI elements for size, position, width, length, and acceptance of characters or numbers. For instance, you must be able to provide inputs to the input fields.
- Check you can execute the intended functionality of the application using the GUI
- Check Error Messages are displayed correctly
- Check for Clear demarcation of different sections on screen
- Check Font used in an application is readable
- Check the alignment of the text is proper
- Check the Color of the font and warning messages is aesthetically pleasing
- Check that the images have good clarity
- Check that the images are properly aligned
- Check the positioning of GUI elements for different screen resolution.

Benefits of using GUI testing are:

- It releases an error-free application software
- It increases the efficiency of software
- Improves software quality

<u>4. Object Oriented Testing : Introduction</u>

- Whenever large scale systems are designed, object oriented testing is done rather than the conventional testing strategies as the concepts of object oriented programming is way different from that of conventional ones.
- The whole object oriented testing revolves around the fundamental entity known as “class”.
- With the help of “class” concept, larger systems can be divided into small well defined units which may then be implemented separately.
- The object oriented testing can be classified as like conventional systems. These are called as the levels for testing



Object Oriented Testing : Levels / Techniques
The levels of object oriented testing can be broadly classified into three categories. These are:

1. Class Testing

- Class testing is also known as unit testing.
- In class testing, every individual classes are tested for errors or bugs.
- Class testing ensures that the attributes of class are implemented as per the design and specifications. Also, it checks whether the interfaces and methods are error free of not.

2. Inter-Class Testing

- It is also called as integration or subsystem testing.
- Inter class testing involves the testing of modules or sub-systems and their coordination with other modules.

3. System Testing

- In system testing, the system is tested as whole and primarily functional testing techniques are used to test the system. Non-functional requirements like performance, reliability, usability and test-ability are also tested.

5. Application Testing

- Client-server Software: Client-server is software architecture consists of client and server systems which communicate to each other either over the computer network or on the same machine. In Client-Server Application Testing, the client system sends the request to the server system and the server system sends the response to the client system. This is also known as a two-tier application. Such applications are developed in Visual Basic, VC++, C, C++, Core JAVA, etc. and the back-end database could be IBM DB2, MS Access, Oracle, Sybase, SQL Server, Quad base, MySQL, etc.

Application Testing

What is Client Server Testing?

- Client Server applications run on two or more systems. It required knowledge on networking. System is installed on the server and an executable file in run of the systems/client machines in intranet. In this type of testing we test the application GUI on both the systems (server and client), we check the functionality, load, database and the interaction between client and server. In Client server testing the user needs to find out the load and performances issues and work on the code area. The test cases and test scenarios for this type of testing are derived from the requirements and experience.
- This type of testing is usually done for 2 tier applications (usually developed for LAN). Here we will be having front-end and backend. The application launched on front-end will be having forms and reports which will be monitoring and manipulating data.
- E.g: applications developed in VB, VC++, Core Java, C, C++, D2K, PowerBuilder etc.
- The backend for these applications would be MS Access, SQL Server, Oracle, Sybase, Mysql, Quadbase

The tests performed on these type of applications would be

- User interface testing
- Manual testing
- Functionality testing
- Compatibility testing
- Configuration testing

What is the Web Application Testing?

Web Applications are the applications which run on two or more machines. The web applications are URL driven and work on web browsers. I
n Web application the web application is loaded on the server machine who's may or may not be known and there is no executable file present which need to be installed on Client Systems. These types of applications are more complex and require broad testing. The tester needs to have knowledge about how the web application is interaction with the user. It also requires knowledge on various technologies like pHp, JavaScript, debugging.
The web testing is done on different browsers, different OS. The application is checked for its browser compatibility and operating system compatibility, features like functionality, back end, GUI, load testing , static page testing.

The tests performed under the web application testing would be:

- User interface testing
- Functionality testing
- Security testing
- Browser compatibility testing
- Operating System compatibility testing
- Load testing and performance testing, stress testing
- Inter-operability testing
- Storage and data volume testing

Test Management Unit 04

Test Plan

- A Test Plan is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test.
- The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.
- As per ISTQB definition: “Test Plan is A document describing the scope, approach, resources, and schedule of intended test activities.”
  A TEST PLAN is a document describing software testing scope and activities. It is the basis for formally testing any software / product in a project.
  Test Plan is a dynamic document. The success of a testing project depends upon a well-written Test Plan document that is current at all times. Test Plan is more or less like a blueprint of how the testing activity is going to take place in a project.
  Given below are a few pointers on a Test Plan:
  #1) Test Plan is a document that acts as a point of reference and only based on that testing is carried out within the QA team.
  #2) It is also a document that we share with the Business Analysts, Project Managers, Dev team and the other teams. This helps to enhance the level of transparency of the QA team’s work to the external teams.
  #3) It is documented by the QA manager/QA lead based on the inputs from the QA team members.

What is the Importance of Test Plan?

- Making Test Plan document has multiple benefits
- Help people outside the test team such as developers, business managers, customers understand the details of testing.
- Test Plan guides our thinking. It is like a rule book, which needs to be followed.
- Important aspects like test estimation, test scope, Test strategy are documented in Test Plan, so it can be reviewed by Management Team and re-used for other projects.

Components of a Test Plan

Scope: Details the objectives of the particular project. Also, it details user scenarios to be used in tests. If necessary, the scope can specify what scenarios or issues the project will not cover.
Schedule: Details start dates and deadlines for testers to deliver results.
Resource Allocation: Details which tester will work on which test.
Environment: Details the nature, configuration, and availability of the test environment.
Tools: Details what tools are to be used for testing, bug reporting, and other relevant activities.
Defect Management: Details how bugs will be reported, to whom and what each bug report needs to be accompanied by. For example, should bugs be reported with screenshots, text logs, or videos of their occurrence in the code?
Risk Management: Details what risks may occur during software testing, and what risks the software itself may suffer if released without sufficient testing.
Exit Parameters: Details when testing activities must stop. This part describes the results that are expected from the QA operations, giving testers a benchmark to compare actual results to.

How to create a Test Plan?

Creating a Test Plan involves the following steps:

1. Product Analysis
2. Designing Test Strategy
3. Defining Objectives
4. Establish Test Criteria
5. Planning Resource Allocation
6. Planning Setup of Test Environment
7. Determine test schedule and estimation
8. Establish Test Deliverables

What is Lifecycle

Lifecycle in the simple term refers to the sequence of changes from one form to other forms. In a similar fashion, Software is also an entity. Just like developing software involves a sequence of steps, testing also has steps which should be executed in a definite sequence.

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 the 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.
Software Testing Life Cycle (STLC) is a sequence of different activities performed during the software testing process.

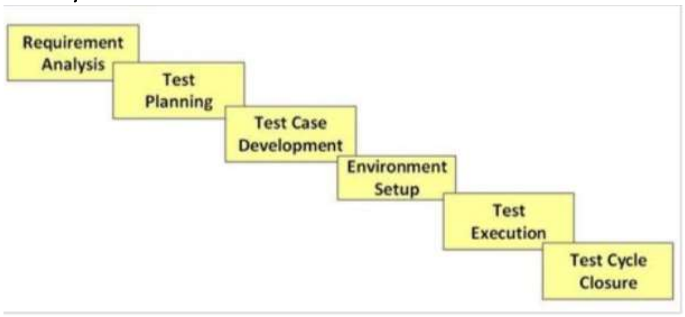
STLC Phases

There are different phases in STLC which are given below. The testing activities start from the Requirements analysis phase and goes through all the phases one by one before completing with the Test cycle closure phase.

- There are 6 STLC Phases in the STLC Lifecycle
- The entry criteria must be fulfilled before each phase can start
- The exit criteria should be fulfilled before exiting a phase
- Every phase has one or more deliverables that are produced at the end of the phase
- The phases are executed in a sequence

The 6 STLC Phases are given below:

1. Requirement analysis
2. Test Planning
3. Test case development
4. Environment Setup
5. Test Execution
6. Test Cycle Closure



- Each of the step mentioned above has some Entry Criteria (it is a minimum set of conditions that should be met before starting the software testing) as well as Exit Criteria (it is a minimum set of conditions that should be completed in order to stop the software testing) on the basis of which it can be decided whether we can move to the next phase of Testing Life cycle or not.

Each of these stages has a definite Entry and Exit criteria, Activities & Deliverables associated with it.
What is Entry and Exit Criteria in STLC?
Entry Criteria: Entry Criteria gives the prerequisite items that must be completed before testing can begin.
Exit Criteria: Exit Criteria defines the items that must be completed before testing can be concluded

- You have Entry and Exit Criteria for all levels in the Software Testing Life Cycle (STLC)

Phase 1: Requirement Analysis

During this phase, feature requirements collected in the SDLC process are evaluated to identify testable aspects. If necessary, testing teams may need to consult with stakeholders to clarify requirements. These requirements can either be functional or non-functional, defining what a feature can do or it's characteristics respectively. The ability to automate testing is also evaluated during this phase.

Activities to be done in Requirement analysis phase are given below:

- Analyzing the System Requirement specifications from the testing point of view

- Preparation of RTM that is Requirement Traceability Matrix
- Identifying the testing techniques and testing types
- Prioritizing the feature which need focused testing
- Analyzing the Automation feasibility
- Identifying the details about the testing environment where actual testing will be done
- Deliverables (Outcome) of Requirement analysis phase are:
- Requirement Traceability Matrix (RTM)
- Automation feasibility report

Phase 2: Test Planning

- During this phase, the test strategy is outlined in a test plan document. This strategy includes tools needed, testing steps, and roles and responsibilities. Part of determining this strategy is a risk and cost analysis and an estimated timeline for testing.

Activities to be done in Test Planning phase are given below:

- Estimation of testing effort
- Selection of Testing Approach
- Preparation of Test Plan, Test strategy documents
- Resource planning and assigning roles and responsibility to them
- Selection of Testing tool

Deliverables (Outcome) of Test Planning phase are:

- Test Plan document
- Test Strategy document
- Best suited Testing Approach
- Number of Resources, skill required and their roles and responsibilities
- Testing tool to be used

Phase 3: Test Case Development

During this phase, test cases are created. Each case defines test inputs, procedures, execution conditions, and anticipated results. Test cases should be transparent, efficient, and adaptable. Once all test cases are created, test coverage should be 100%. Any necessary automation scripts are also created during this phase.

Activities to be done in Test Case Development phase are given below:

- Creation of test cases
- Creation of test scripts if required
- Verification of test cases and automation scripts
- Creation of Test Data in testing environment

Deliverables (Outcome) of Test Case Development phase are:

- Test cases
- Test scripts (for automation if required)
- Test Data

Phase 4: Test Environment Setup

During this phase, testing environments are configured and deployed. This phase may include a variety of testing tools, including TestComplete, Selenium, Appium, or Katalon Studio. Sometimes, this phase also includes setting up test servers. Once environments are deployed, smoke tests are performed to ensure that environments are working as expected with all intended functionality.

Activities to be done in Test Environment Setup phase are given below:

- As per the Requirement and Architecture document the list of required software and hardware is prepared
- Setting up of test environment
- Creation of test data
- Installation of build and execution of Smoke testing on it

Deliverables (Outcome) of Test Environment Setup phase are:

- Test Environment setup is ready
- Test Data is created
- Results of Smoke testing

Phase 5: Test Execution
During this phase, features are tested in the deployed environment, using the established test cases. Expected test results are compared to actual and results are gathered to report back to development teams.

Activities to be done in Test Execution phase are given below:

- Execution of Test Cases
- Reporting test results
- Logging defects for the failed test cases
- Verification and retesting of the defect
- Closure of defects

Deliverables (Outcome) of Test Execution phase are:

- Test execution Report
- Updated test cases with results
- Bug Report

Phase 6: Test Cycle Closure
This is the last phase of the STLC, during which a test result report is prepared. This report should summarize the entire testing process and provide comparisons between expected results and actual. These comparisons include objectives met, time taken, total costs, test coverage, and any defects found.

Activities to be done in Test Cycle Closure phase are given below:

- To evaluate the test completion on the basis of Test Coverage and Software Quality
- Documentation of the learning from the project
- Analyzing the test results to find out the distribution of severe defects
- Test Closure Report preparation

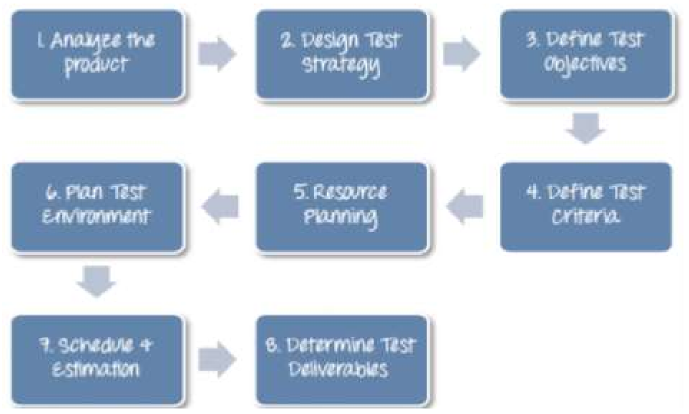
Deliverables (Outcome) of Test Cycle Closure phase are:

- Report of Test Closure

<u>How to write a Test Plan</u>
This means that the test plan conveys how testing will be performed at a particular level (such as system testing or user acceptance testing), or for a particular type of testing (such as performance testing or security testing).
The Test Plan (sometimes also referred to as a QA Test Plan) can be seen as the instruction manual or guide for your testing effort. It describes the objectives of testing (what are you planning to verify and/or validate), the scope of testing (what will and will not be tested), together with the general and sometimes detailed schedule of the activities you want to perform (how and when are you testing).

You already know that making a Test Plan is the most important task of Test Management Process. Follow the seven steps below to create a test plan as Analyze the product

1. Design the Test Strategy
2. Define the Test Objectives
3. Define Test Criteria
4. Resource Planning
5. Plan Test Environment
6. Schedule & Estimation
7. Determine Test Deliverables

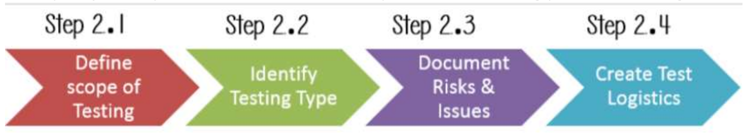


Step 1) Analyze the product

- How can you test a product without any information about it? The answer is Impossible. You must learn a product thoroughly before testing it.
- The product under test is ABC banking website. You should research clients and the end users to know their needs and expectations from the application
- Who will use the website?
- What is it used for?
- How will it work?
- What are software/ hardware the product uses?

Step 2) Develop Test Strategy

- Test Strategy is a critical step in making a Test Plan in Software Testing. A Test Strategy document, is a high-level document, which is usually developed by Test Manager. This document defines:
- The project's testing objectives and the means to achieve them
- Determines testing effort and costs
- For project, you need to develop Test Strategy for testing that banking website. You should follow steps below



Step 2.1) Define Scope of Testing

Before the start of any test activity, scope of the testing should be known. You must think hard about it.

- The components of the system to be tested (hardware, software, middleware, etc.) are defined as "**in scope**"
- The components of the system that will not be tested also need to be clearly defined as being "**out of scope**."

Defining the scope of your testing project is very important for all stakeholders. A precise scope helps you

- Give everyone a **confidence & accurate information** of the testing you are doing
- All project members will have a **clear** understanding about what is tested and what is not

How do you determine scope your project?

To determine scope, you must –

- Precise customer requirement
- Project Budget
- Product Specification
- Skills & talent of your test team

Now should clearly define the "in scope" and "out of scope" of the testing.

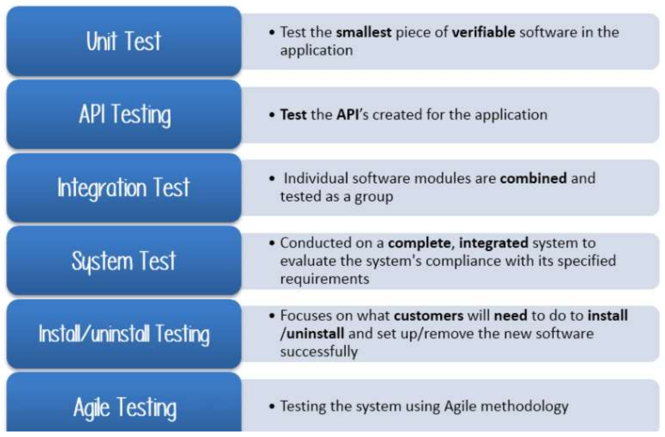
- As the software requirement specs, the project ABC Bank only focus on testing all the **functions** and external interface of website **ABC** Bank (**in scope** testing)
- Nonfunctional testing such as **stress**, **performance** or **logical database** currently will not be tested. (**out of** scope)

Step 2.2) Identify Testing Type

A **Testing Type** is a standard test procedure that gives an expected test outcome.

Each testing type is formulated to identify a specific type of product bugs. But, all Testing Types are aimed at achieving one common goal "**Early detection of** all the defects before releasing the product to the customer"

The **commonly used** testing types are described as following figureCommonly Used Testing Types



Unit Test	• Test the **smallest** piece of **verifiable** software in the application
API Testing	• **Test** the **API**'s created for the application
Integration Test	• Individual software modules are **combined** and tested as a group
System Test	• Conducted on a **complete, integrated** system to evaluate the system's compliance with its specified requirements
Install/uninstall Testing	• Focuses on what **customers** will **need** to do to **install /uninstall** and set up/remove the new software successfully
Agile Testing	• Testing the system using Agile methodology

There are **tons of Testing Types** for testing software product. Your team **cannot have** enough efforts to handle all kind of testing. As Test Manager, you must set **priority** of the Testing Types

- Which Testing Types should be **focused** for web application testing?
- Which Testing Types should be **ignored** for saving cost?

Step 2.3) Document Risk & Issues

Risk is future's **uncertain event** with a probability of **occurrence** and a **potential** for loss. When the risk actually happens, it becomes the '**issue'.**

Risk	**Mitigation**
Team member lack the required skills for website testing.	Plan **training course** to skill up your members
The project schedule is too tight; it's hard to complete this project on time	Set **Test Priority** for each of the test activity.
Test Manager has poor management skill	Plan **leadership training** for manager
A lack of cooperation negatively affects your employees' productivity	**Encourage** each team member in his task, **and inspire** them to greater efforts.
Wrong budget estimate and cost overruns	Establish the **scope** before beginning work, pay a lot of attention to project planning and constantly track and measure the progress

Step 2.4) Create Test Logistics

In Test Logistics, the Test Manager should answer the following questions:

- **Who** will test?
- **When** will the test occur?

Who will test?

You may not know exact names of the tester who will test, but the **type of tester** can be defined.

To select the right member for specified task, you have to consider if his skill is qualified for the task or not, also estimate the project budget. Selecting wrong member for the task may cause the project to **fail** or **delay**.

Person having the following skills is most ideal for performing software testing:

- Ability to **understand** customers point of view
- Strong **desire** for quality
- **Attention** to detail
- Good **cooperation**

In your project, the member who will take in charge for the test execution is the **tester.** Base on the project budget, you can choose in-source or outsource member as the tester.

When will the test occur?

Test activities must be matched with associated development activities.

You will start to test when you have **all required items** shown in following

- Test Specification
- Human Resources
- Test Environmnet

Step 3) Define Test Objective

- Defining the test objectives should be the ultimate goal of achieving the test executions. The main goal of testing is to find as many software bugs as possible, to ensure that the software is free from all the bugs. To interpret and document test objective, you will need to follow 2 steps;

1. List down all the features and functionality of the system including its performance and user interface.
2. Identify the target or the end result based on the above features.

Step 4) Define Test Criteria

- Test Criteria is a standard or rule on which a test procedure or test judgment can be based. There're 2 types of test criteria as following
  Suspension Criteria
  Specify the critical suspension criteria for a test. If the suspension criteria are met during testing, the active test cycle will be suspended until the criteria are resolved.
  Test Plan Example: If your team members report that there are **40%** of test cases failed, you should **suspend** testing until the development team fixes all the failed cases.

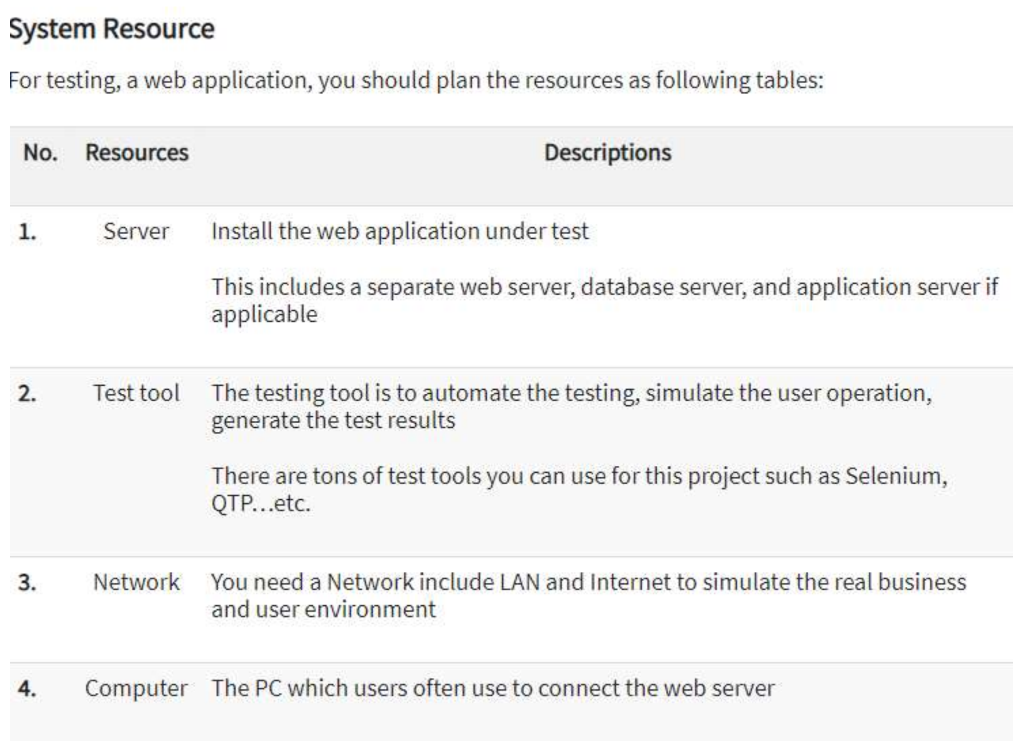
  Exit Criteria
  It specifies the criteria that denote a successful completion of a test phase. The exit criteria are the targeted results of the test and are necessary before proceeding to the next phase of development. Example: 95% of all critical test cases must pass.
  Example: **95%** of all critical test cases must pass.
  Some methods of defining exit criteria are by specifying a targeted **run rate** and **pass rate**.
- Run rate is ratio between **number test cases executed/total test cases** of test specification. For example, the test specification has total 120 TCs, but the tester only executed 100 TCs, So the run rate is 100/120 = 0.83 (83%)
- Pass rate is ratio between **numbers test cases passed / test cases executed**. For example, in above 100 TCs executed, there're 80 TCs that passed, so the pass rate is 80/100 = 0.8 (80%)
  This data can be retrieved in Test Metric documents.
- **Run** rate is mandatory to be **100%** unless a clear reason is given.
- **Pass** rate is dependent on project scope, but **achieving high pass rate** is a goal.

Step 5) Resource Planning

- Resource plan is a detailed summary of all types of resources required to complete project task. Resource could be human, equipment and materials needed to complete a project
- The resource planning is important factor of the test planning because helps in determining the number of resources (employee, equipment…) to be used for the project. Therefore, the Test Manager can make the correct schedule & estimation for the project.
- This section represents the recommended resources for your project.
  Human Resource
  The following table represents various members in your project team

No.	Member	Tasks
1.	Test Manager	**Manage** the whole project Define project **directions** Acquire appropriate resources

2.	Tester	Identifying and describing appropriate test techniques/tools/automation architecture Verify and assess the Test Approach **Execute** the tests, **Log** results, **Report** the defects. Tester could be in-sourced or out-sourced members, base on the project budget F20or the task which required **low** skill, I recommend you 0choose **outsourced** membe to **save** project cost.
3.	Developer	**Implement** the test cases, test program, test suite etc.
4.	Test Administrator	Builds up and ensures Test Environment and assets are **managed** and **maintained** **Support**Tester to use the test environment for test execution
5.	SQA members	Take in charge of quality assurance Check to confirm whether the testing process is meeting specified requirements



System Resource

For testing, a web application, you should plan the resources as following tables:

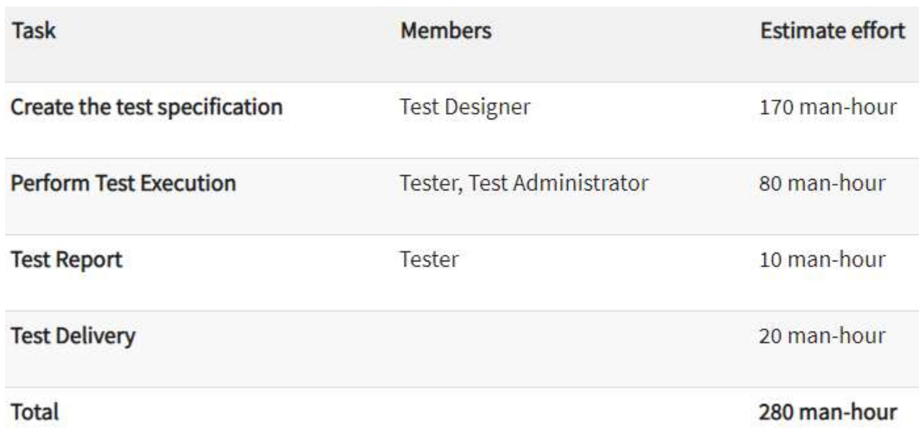
No.	Resources	Descriptions
1.	Server	Install the web application under test This includes a separate web server, database server, and application server if applicable
2.	Test tool	The testing tool is to automate the testing, simulate the user operation, generate the test results There are tons of test tools you can use for this project such as Selenium, QTP…etc.
3.	Network	You need a Network include LAN and Internet to simulate the real business and user environment
4.	Computer	The PC which users often use to connect the web server

Step 6) Plan Test Environment
What is the Test Environment?
A testing environment is a setup of software and hardware on which the testing team is going to execute test cases. The test environment consists of real business and user environment, as well as physical environments, such as server, front end running environment.
How to setup the Test Environment

You should ask the developer some questions to understand the web application under test clearly. Here're some recommended questions. Of course, you can ask the other questions if you need.
What is the maximum user connection which this website can handle at the same time?
What are hardware/software requirements to install this website?
Does the user's computer need any particular setting to browse the website?

Step 7) Estimation and Scheduling

- In the test environment phase, the test manager has already used techniques to come to the conclusion of estimating the project. Now along with the estimate its necessary you bind to the schedule of test planning.



Task	Members	Estimate effort
Create the test specification	Test Designer	170 man-hour
Perform Test Execution	Tester, Test Administrator	80 man-hour
Test Report	Tester	10 man-hour
Test Delivery		20 man-hour
Total		280 man-hour

- In the Test Estimation phase, suppose you break out the whole project into small tasks and add the estimation for each task as below
- Then you create the schedule
- To create the project schedule, the Test Manager needs several types of input as below:
  Employee and project deadline: The working days, the project deadline, resource availability are the factors which affected to the schedule
  Project estimation: Base on the estimation, the Test Manager knows how long it takes to complete the project. So he can make the appropriate project schedule
  Project Risk : Understanding the risk helps Test Manager add enough extra time to the project schedule to deal with the risks

Step 8) Test Deliverables

- Test Deliverables is a list of all the documents, tools and other components that has to be developed and maintained in support of the testing effort.
- There are different test deliverables at every phase of the software development lifecycle.

Before Testing
During Testing
After the Testing

Test deliverables are provided before testing phase.

- Test plans document.
- Test cases documents
- Test Design specifications.

Test deliverables are provided during the testing

- Test Scripts
- Simulators.
- Test Data
- Test Traceability Matrix
- Error logs and execution logs.

Test deliverables are provided after the testing cycles is over.

- Test Results/reports
- Defect Report
- Installation/ Test procedures guidelines
- Release notes

<u>Test Management</u>

Test Management is a process of managing the testing activities in order to ensure high quality and high-end testing of the software application.

The method consists of organizing, controlling, ensuring traceability and visibility of the testing process in order to deliver the high quality software application.

It ensures that the software testing process runs as expected.

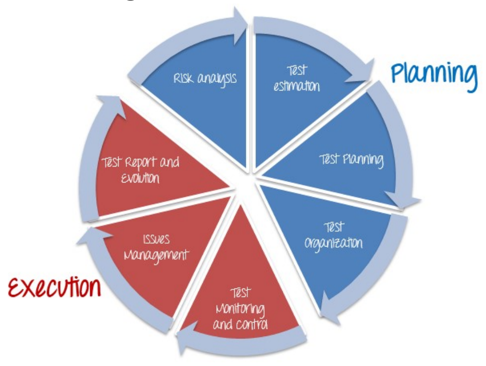
As suggested by its name, test management is a process of managing testing activities, such as planning, execution, monitoring, and controlling activities. Test management involves several crucial activities that caters to both manual and automation testing. With the assistance of this process, a team lead can easily manage the entire testing team, while monitoring their activities, as well as paying close attention to various details of SDLC.

Test management, process of managing the tests. A test management is also performed using tools to manage both types of tests, automated and manual, that have been previously specified by a test procedure.

Test Management Responsibilities:

- Test Management has a clear set of roles and responsibilities for improving the quality of the product.
- Test management helps the development and maintenance of product metrics during the course of project.
- Test management enables developers to make sure that there are fewer design or coding faults.

Test Management Phases



Test Management Process

Test Management Process is a procedure of managing the software testing activities from start to the end. The test management process provides planning, controlling, tracking and monitoring facilities throughout the whole project cycle. The process involves several activities like test planning, designing and test execution. It gives an initial plan and discipline to the software testing process.

There are two main Parts of Test Management Process: -

- Planning
  Risk Analysis
  Test Estimation
  Test Planning
  Test Organization
- Execution
  Test Monitoring and Control
  Issue Management
  Test Report and Evaluation

Planning

Risk Analysis and Solution

Risk is the potential loss (an undesirable outcome, however not necessarily so) resulting from a given action or an activity.
Risk Analysis is the first step which Test Manager should consider before starting any project. Because all projects may contain risks, early risk detection and identification of its solution will help Test Manager to avoid potential loss in the future & save on project cost.

Test Estimation

- An estimate is a forecast or prediction. Test Estimation is approximately determining how long a task would take to complete. Estimating effort for the test is one of the major and important tasks in Test Management.
- Benefits of correct estimation:

1. Accurate test estimates lead to better planning, execution and monitoring of tasks under a test manager's attention.
2. Allow for more accurate scheduling and help realize results more confidently.

Test Planning

- A Test Plan can be defined as a document describing the scope, approach, resources, and schedule of intended Testing activities.

- A project may fail without a complete Test Plan. Test planning is particularly important in large software system development.
- In software testing, a test plan gives detailed testing information regarding an upcoming testing effort, including:
- Test Strategy
- Test Objective
- Exit /Suspension Criteria
- Resource Planning
- Test Deliverables

Test Organization
Test Organization in Software Testing is a procedure of defining roles in the testing process. It defines who is responsible for which activities in testing process. Test functions, facilities and activities are also explained in the same process. The competencies and knowledge of the people involved are also defined however everyone is responsible for quality of testing process.

Execution
1. Test Monitoring and Control
Monitoring
- Monitoring is a process of collecting, recording, and reporting information about the project activity that the project manager and stakeholder needs to know.
- To Monitor, Test Manager does following activities
- Define the project goal, or project performance standard
- Observe the project performance, and compare between the actual and the planned performance expectations
- Record and report any detected problem which happens to the project

Controlling
- Project Controlling is a process of using data from monitoring activity to bring actual performance to planned performance.
- In this step, the Test Manager takes action to correct the deviations from the plan. In some cases, the plan has to be adjusted according to project situation.
- While testing work is in progress or while the testers are executing the test plan, all these work progress must be monitored. One should keep track of all this testing work. If test monitoring is done, then the test team and test manager will get feedback on how the testing progress is?
- Using this feedback, the test manager can guide the team members to improve the quality of further testing work. With the help of test monitoring, the project team will get visibility on the test results. It also helps to know about test coverage.
-

2. Issue Management
In the life cycle of any project, there will be always an unexpected problems and questions that crop up. For an example:
- The company cuts down your project budget
- Your project team lacks the skills to complete project
- The project schedule is too tight for your team to finish the project at the deadline.

Risk to be avoided while testing:
- Missing the deadline
- Exceed the project budget
- Lose the customer trust

3. Test Report & Evaluation
- The project has already completed. It’s now time for look back what you have done
- Test report helps to identify test coverage, quality of the developed product and the required process improvements. We can decide ‘how much testing is required?’
- If enough testing is done, then we can submit this test report to the stakeholders or clients. So that they will also get to know the quality of the product and have an idea of how much testing is performed on the product.

<u>Infrastructure testing</u>

What Is Infrastructure?

- IT Infrastructure Ecosystem includes Operating Systems platforms (such as Windows, UNIX, Linux, macOS), Computer Hardware platforms (such as Dell, IBM, Sun, HP, Apple), Internet platforms (such as Apache, Cisco, Microsoft IIS, .NET), Data Management and Storage (such as IBM DB2, Oracle, SQL Server, MySQL) and Enterprise Software Applications (such as SAP, Oracle, Microsoft).

What Is Infrastructure Testing?

Every software requires an infrastructure to perform its actions. Infrastructure testing is the testing process that covers hardware, software, and networks. It involves testing of any code that reads configuration values from different things in the IT framework and compares them to intended results.

It reduces the risks of failure. This testing incorporates testing exercises, procedures to guarantee that IT applications and the fundamental infrastructure are tuned to deliver on execution, adaptability, unwavering quality, accessibility, performance, and scalability. The aim is to test infrastructure between test environments, test tools, and office environments.

#Infrastructure Testing Team

The infrastructure-testing team has a good bunch of knowledge related to this testing. They are also involved with the Quality Assurance team. This team knows how to test IT infrastructure. This team knows how to design test cases for this type of testing.

- **When To Perform Infrastructure Testing?**
  There is an urgent need to perform this testing whenever any infrastructure-related changes are introduced.
  Examples of such changes are:

1. Any new patch in the system is developed.
2. Any new system updates are experienced.
3. Any update in Operating System.
4. The database version/structure is upgraded.
5. When there is memory up-gradation for servers.
6. Implementation of the new tool.
7. Security fixes.
8. Software update.

What Are The Benefits Of Infrastructure Testing?

- The planned and exhaustive approach of Infrastructure Testing gives many benefits to a software product as well as organizations.
- Few of the benefits are enlisted below:

1. Reduction in Production failures.
2. Improvement in defect identification before production execution. Upgrade the quality of infrastructure with zero defect slippage to production.
3. Quickened test execution, empowering early go live.
4. It helps in annual cost savings in operations as well as in business.
5. Confirm that software works in a systematic and controlled procedure.
6. Reduction in downtime.
7. Improvement in quality of service.
8. Availability of stable environments.
9. Reduction in the cost involved in risks.
10. Better user experience.

<u>Test people management</u>

- A managers most important, and most difficult job is to manage people. You must lead, motivate, inspire and encourage them.
- Sometimes manager will have to hire, fire, discipline or evaluate employees.
- Your job as a manager is to get things done. However it also means getting things done through others.
- People management is an integral part of any project management.
- People management also require the ability to hire, motivate and retain the right people.
- Testing projects present several additional challenges.

Test Process

Testing is not a single activity instead it's a set of number of processes.
It include following activities.
1.Base Lining a Test plan
2.Test Case Specification
3.Update of Traceability Matrix

Test processes are a vital part of Software Development Life Cycle (SDLC) and consist of various activities, which are carried out to improve the quality of the software product. From planning to execution, each stage of the process is systematically planned and require discipline to act upon them. These steps and stages are extremely important, as they have their own entry criteria and deliverable, which are combined and evaluated to get expected results and outcomes.
Therefore, we can say that the quality and effectiveness of the software testing is primarily determined by the quality of the test processes used by the software testers. Moreover, by following a fundamental test process, testers can simplify their work and keep a track of every major and minor activity.

Fundamental test process divided into following steps
1) Test Planning and Control
2) Test Analysis and Design
3) Test Implementation and Execution
4) Evaluating exit criteria and Reporting
5) Test Closure activities

Base Lining a Test plan

- A test combines all the points into a single document that acts as an anchor point for the entire testing project.
- An organization normally arrives at a template that is to be used across the board. Each testing project puts together a test plan based on the template.
- The test plan is reviewed by a designated set of component people in the organization.
- After this, test plan is baselined into configuration management repository.
- From then on, the baselined test plan becomes the basis for running the testing project.
- In addition, periodically any changes needed to the test plan templates are discusses among the different stakeholders and this is kept current and applicable to the testing teams.

Test Case Specification

- Test Case Specification document described detailed summary of what scenarios will be tested, how they will be tested, how often they will be tested, and so on and so forth, for a given feature. It specifies the purpose of a specific test, identifies the required inputs and expected results, provides step-by-step procedures for executing the test, and outlines the pass/fail criteria for determining acceptance.
- Test case specification has to be done separately for each unit. Based on the approach specified in the test plan first the feature to be tested for this unit must be determined.
- The overall approach stated in the plan is refined into specific test techniques that should be followed and into the criteria to be used for evaluation. Based on these the test cases are specified for testing unit.
- The two basic reasons test cases are specified before they are used for testing. It is known that testing has severe limitations and the effectiveness of testing depends very heavily on the exact nature of the test case. Even for a given criterion the exact nature of the test cases affects the effectiveness of testing.
- Constructing good test case that will reveal errors in programs is still a very creative activity that depends a great deal on the tester. Clearly it is important to ensure that the set of test cases used is of high quality. As with many other verification methods evaluation of quality of test case is done through "test case review" And

for any review a formal document or work product is needed. This is the primary reason for having the test case specification in the form of a document.

<u>What is Traceability Matrix? (TM)</u>

- A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.
- It is used to track the requirements and to check the current project requirements are met.

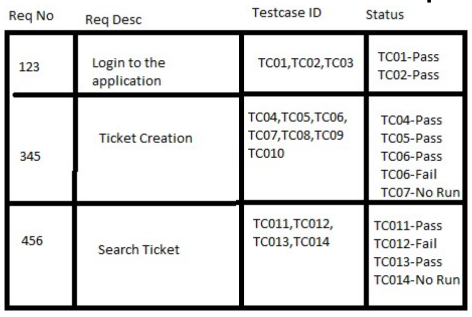
Requirement Traceability Matrix (RTM)

Requirement Traceability Matrix (RTM) is a document that maps and traces user requirement with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software devlopement life cycle. The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

In Requirement Traceability Matrix or RTM, we set up a process of documenting the links between the user requirements proposed by the client to the system being built. In short, it’s a high-level document to map and trace user requirements with test cases to ensure that for each and every requirement adequate level of testing is being achieved.

The traceability matrix is typically a worksheet that contains the requirements with its all possible test scenarios and cases and their current state, i.e. if they have been passed or failed. This would help the testing team to understand the level of testing activities done for the specific product.

Which Parameters to include in Requirement Traceability Matrix



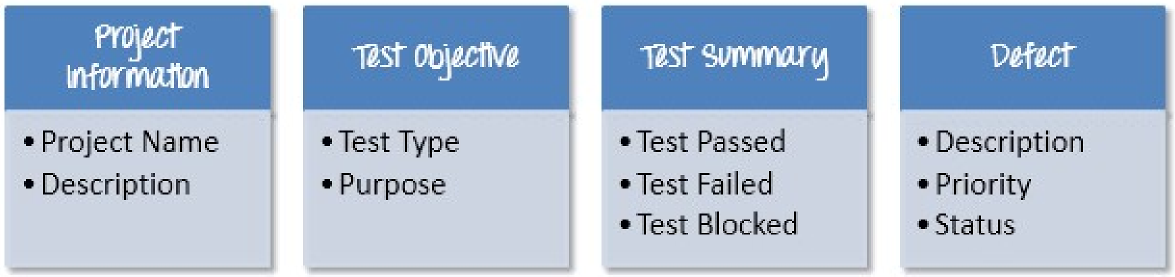
Req No	Req Desc	Testcase ID	Status
123	Login to the application	TC01,TC02,TC03	TC01-Pass TC02-Pass
345	Ticket Creation	TC04,TC05,TC06, TC07,TC08,TC09 TC010	TC04-Pass TC05-Pass TC06-Pass TC06-Fail TC07-No Run
456	Search Ticket	TC011,TC012, TC013,TC014	TC011-Pass TC012-Fail TC013-Pass TC014-No Run

Requirement ID
Requirement Type and Description
Test Cases with Status

<u>Test Report</u>

- Test Report is a document which contains a summary of all test activities and final test results of a testing project. Test report is an assessment of how well the Testing is performed. Based on the test report, stakeholders can evaluate the quality of the tested product and make a decision on the software release.
- For example, if the test report informs that there are many defects remaining in the product, stakeholders can delay the release until all the defects are fixed.
- A test report is an organized summary of testing objectives, activities, and results. It is created and used to help stakeholders (product manager, analysts, testing team, and developers) understand product quality and decide whether a product, feature, or a defect resolution is on track for release.
- Test reporting is essential for making sure your web or mobile app is achieving an acceptable level of quality.
- test report as a document containing information about the performed actions (run test cases, detected bugs, spent time etc.) and the results of this perfomance (failed/passed test cases, the number of bugs and crashes etc.).

What does a test report contain?



Project Information	Test Objective	Test Summary	Defect
• Project Name • Description	• Test Type • Purpose	• Test Passed • Test Failed • Test Blocked	• Description • Priority • Status

Project Information
All information of the project such as the project name, product name, and version should be described in the test report. For example, the information of XYZBank project

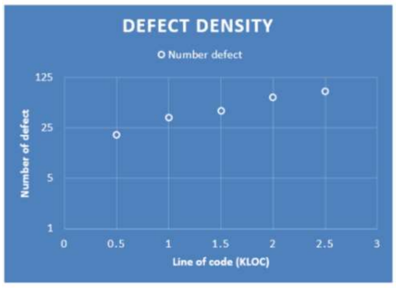
Test Objective
As mentioned in Test planning , Test Report should include the objective of each round of testing, such as Unit Test, Performance Test, System Test …Etc.

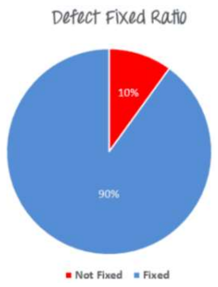
Test Summary
- This section includes the summary of testing activity in general. Information detailed here includes
- The number of test cases executed
- The numbers of test cases pass
- The numbers of test cases fail
- Pass percentage
- Fail percentage
- Comments

Defect
One of the most important information in Test Report is defect. The report should contain following information
- Total number of bugs
- Status of bugs (open, closed, responding)
- Number of bugs open, resolved, closed
- Breakdown by severity and priority
- Like test summary, you can include some simple metrics like Defect density, % of fixed defects.

The project team sent you the Defect information as followin
- Defect density is 20 defects/1000 lines of code average
- 90% defects fixed in total

You can represent the data as following graph





__**Tips to write a good test report**__
Test report is a communication tool between the Test Manager and the stakeholder. Through the test report, the stakeholder can understand the project situation, the quality of product and other things.
The following scenario shows you why we need a good Test Report
You co-operate with outsourcing company, its tester after having performed Performance Testing of the website Guru99 Bank, sends you a test report like this
The information of that report is too abstract. It does not have any detailed information. The stakeholder who will read it might be slightly puzzled when they get it. They might ask or have following sets of questions: -
- Why did they not execute 30 TCs that remains
- What are these failed Test Cases
- Doesn't have any bugs description

To solve that problem, a good Test Report should be:
- Detail: You should provide a detailed description of the testing activity, show which testing you have performed. Do not put the abstract information into the report, because the reader will not understand what you said.
- Clear: All information in the test report should be short and clearly understandable.
- Standard: The Test Report should follow the standard template. It is easy for stakeholder to review and ensure the consistency between test reports in many projects.

- Specific: Do not write an essay about the project activity. Describe and summarize the test result specification and focus on the main point.

Recommending Product Release

- A test report is a description or explanation or justification the status of a test project.
- Based on test summary report, an organization can take a decision on whether to release a product or not.
- Ideally, an organization would like to release a product with zero defects.
- However, market pressures may cause the product to be released with the senior management is convinced that there is no major risk of customer dissatisfaction.
- Such a decision should be taken by the senior manager after consultation with customer support team, development team and testing team.

Executing Test Cases

- The prepared test cases have to be executed at the appropriate times during a project. As test cases are executed during a test cycle, the defect repository is updated with,
  1. Defect from earlier test cycles that are fixed in the current build.
  2. New defect that get uncovered in the current run of tests.
  3.Follow test procedure to execute test suite and test cases.
  4. confirm test, re test re – execution to fix
  5.Log the result of test execution, status of test pass and fail.
  6. Comparison of actual results with expected results.
  7.Defect occurrences and differences should be updated in report
  8.Decide suspension and resumption of further test cases also update traceability matrix
- During test design and execution, the traceability matrix should be kept current.
- As when test gets designed and executed successfully, the traceability matrix should be updated.

Collecting and Analyzing Metrics:

- When test are executed, information about test execution gets collected in test logs and other files. The basic measurements from running the tests are then converted to meaningful metrics by the use of appropriate transformation and formulae
- Percentage test cases executed= (No of test cases executed/ Total no of test cases written) X 100
- Passed Test Cases Percentage = (Number of Passed Tests/Total number of tests executed) X 100
- Failed Test Cases Percentage = (Number of Failed Tests/Total number of tests executed) X 100
- Blocked Test Cases Percentage = (Number of Blocked Tests/Total number of tests executed) X 100
- Fixed Defects Percentage = (Defects Fixed/Defects Reported) X 100
- Accepted Defects Percentage = (Defects Accepted as Valid by Dev Team /Total Defects Reported) X 100
- Defects Deferred Percentage = (Defects deferred for future releases /Total Defects Reported) X 100
- Critical Defects Percentage = (Critical Defects / Total Defects Reported) X 100
- Number of tests run per time period = Number of tests run/Total time
- Test design efficiency = Number of tests designed /Total time
- Test review efficiency = Number of tests reviewed /Total time
- Bug find rote or Number of defects per test hour = Total number of defects/Total number of test hours

Test Summary Report

- Test Summary Report is an important deliverable which is prepared at the end of a Testing project, or rather after Testing is completed. The prime objective of this document is to explain various details and activities about the Testing performed for the Project, to the respective stakeholders like Senior Management, Client, etc.
- As part of Daily Status Reports, daily testing results will be shared with involved stakeholders every day. But the Test Summary Report provides a consolidated report on the Testing performed so far for the project.
- test summary report is a formal document that summarizes the results of all testing efforts for a particular testing cycle of a project / module or a sub module. Generally, test leads or test managers prepare this document at the end of testing cycle. Some test managers prepares it at the end of project.

Contents of a Test summary report:

1. **Purpose of the document**
2. **Application Overview**
3. **Testing Scope**
4. **Metrics**
5. **Types of testing performed**
6. **Test Environment & Tools**
7. **Lessons Learned**
8. **Recommendations**
9. **Best Practices**
10. **Exit Criteria**
11. **Conclusion/Sign Off**
    - ✓ **'Go Live'**
12. **Definitions, Acronyms, and Abbreviations**

UNIT 05 Defect Management

What is Bug?

A bug is the consequence/outcome of a coding fault.

Defect in Software Testing

- When a system gives a different output other than the actual business requirement i.e. when there is a deviation from the actual or original business requirement then we can say that there is a defect in the system/software.
- When the testing team executes the test cases, they come across a situation where the actual test result is different from the expected result. This variation is termed as a **Defect.**

Bug Report in Software Testing

- A Bug Report in Software Testing is a detailed document about bugs found in the software application. Bug report contains each detail about bugs like description, date when bug was found, name of tester who found it, name of developer who fixed it, etc. Bug report helps to identify similar bugs in future so it can be avoided.
- While reporting the bug to developer, your Bug Report should contain the following information
  Defect ID - Unique identification number for the defect.
  Defect Description - Detailed description of the Defect including information about the module in which Defect was found.
  Version - Version of the application in which defect was found.
  Steps - Detailed steps along with screenshots with which the developer can reproduce the defects.
  Date Raised - Date when the defect is raised
  Reference- where in you provide reference to the documents like. Requirements, design, architecture or maybe even screenshots of the error to help understand the defect
  Detected By - Name/ID of the tester who raised the defect
  Status - Status of the defect, more on this later
  Fixed by - Name/ID of the developer who fixed it
  Date Closed - Date when the defect is closed
  Severity which describes the impact of the defect on the application
  Priority which is related to defect fixing urgency. Severity Priority could be High/Medium/Low based on the impact urgency at which the defect should be fixed respectively

What is Defect Management Process?

- The defect management process is the core of software testing. Once the defects have been identified, the most significant activity for any organization is to manage the flaws, not only for the testing team but also for everyone involved in the software development or project management process.
- As we know, defect prevention is an effective and efficient way to decrease the number of defects. The defect prevention is a very cost-effective process to fix those defects discovered in the earlier stages of software processes.
- The Defect Management Process is process where most of the organizations manage the Defect Discovery, Defect Removal, and then the Process Improvement.
- **Defect Management Process (DMP)**, as name suggests, is a process of managing defects by simply identifying and resolving or fixing defects. If defect management process is done in more efficient manner with full focus, then less buggy software will be available more in the market.

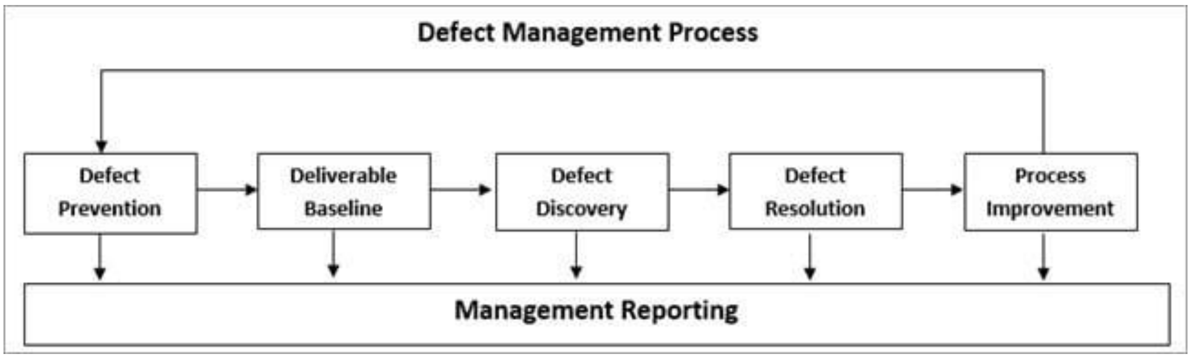
Goals of Defect Management Process (DMP):

- Prevent the defect
- Detection at an early stage
- Reduce the impact or effects of defect on software
- Resolving or fixing defects
- Improving process and performance of software

The main purpose of DMP for different projects or organization is given below:

- Operational support for simply resolving and retesting defects being found.

- To give input for status and progress report regarding defect.
- To give input for advice regarding release of defect.
- To identify the main reason that the defect occurred and how to handle it



#1) Defect Prevention:

Defect Prevention is the best method to eliminate the defects in the early stage of testing instead of finding the defects in the later stage and then fixing it. This method is also cost effective as the cost required for fixing the defects found in the early stages of testing is very low.
However, it is not possible to remove all the defects but at least you can minimize the impact of the defect and cost to fix the same.

The major steps involved in Defect Prevention are as follow:

- **Identify Critical Risk**: Identify the critical risks in the system which will impact more if occurred during testing or in the later stage.
- **Estimate Expected Impact**: For each critical risk, calculate how much would be the financial impact if the risk actually encountered.
- **Minimize expected impact**: Once you identify all critical risks, take the topmost risks which may be harmful to the system if encountered and try to minimize or eliminate the risk. For risks which cannot be eliminated, it reduces the probability of occurrence and its financial impact.

#2) Deliverable Baseline:

When a deliverable (system, product or document) reaches its pre-defined milestone then you can say a deliverable is a baseline. In this process, the product or the deliverable moves from one stage to another and as the deliverable moves from one stage to another, the existing defects in the system also gets carried forward to the next milestone or stage.
For Example, consider a scenario of coding, unit testing and then system testing. If a developer performs coding and unit testing then system testing is carried out by the testing team. Here coding and Unit Testing is one milestone and System Testing is another milestone.

So during unit testing, if the developer finds some issues then it is not called as a defect as these issues are identified before the meeting of the milestone deadline. Once the coding and unit testing have been completed, the developer hand-overs the code for system testing and then you can say that the code is **"baselined"** and ready for next milestone, here, in this case, it is "system testing".

Now, if the issues are identified during testing then it is called as the defect as it is identified after the completion of the earlier milestone i.e. coding and unit testing.
Basically, the deliverables are baselined when the changes in the deliverables are finalized and all possible defects are identified and fixed. Then the same deliverable passes on to the next group who will work on it.

#3) Defect Discovery:

It is almost impossible to remove all the defects from the system and make a system as a defect-free one. But you can identify the defects early before they become costlier to the project. We can say that the defect discovered means it is formally brought to the attention of the development team and after analysis of that the defect development team also accepted it as a defect.

Steps involved in Defect Discovery are as follows:

- **Find a Defect**: Identify defects before they become a major problem to the system.

- **Report Defect**: As soon as the testing team finds a defect, their responsibility is to make the development team aware that there is an issue identified which needs to be analyzed and fixed.
  Acknowledge Defect: Once the testing team assigns the defect to the development team, its the development team's responsibility to acknowledge the defect and continue further to fix it if it is a valid defect.

#4) Defect Resolution:
In the process, the testing team has identified the defect and reported to the development team. Now here the development team needs to proceed for the resolution of the defect.
The steps involved in the defect resolution are as follows:

- **Prioritize the risk**: Development team analyzes the defect and prioritizes the fixing of the defect. If a defect has more impact on the system then they make the fixing of the defect on a high priority.
- **Fix the defect**: Based on the priority, the development team fixes the defect, higher priority defects are resolved first and lower priority defects are fixed at the end.
- **Report the Resolution**: Its the development team's responsibility to ensure that the testing team is aware when the defects are going for a fix and how the defect has been fixed i.e. by changing one of the configuration files or making some code changes. This will be helpful for the testing team to understand the cause of the defect.
-

#5) Process Improvement:
Though in the defect resolution process the defects are prioritized and fixed, from a process perspective, it does not mean that lower priority defects are not important and are not impacting much to the system. From process improvement point of view, all defects identified are same as a critical defect.

Even these minor defects give an opportunity to learn how to improve the process and prevent the occurrences of any defect which may impact system failure in the future. Identification of a defect having a lower impact on the system may not be a big deal but the occurrences of such defect in the system itself is a big deal.

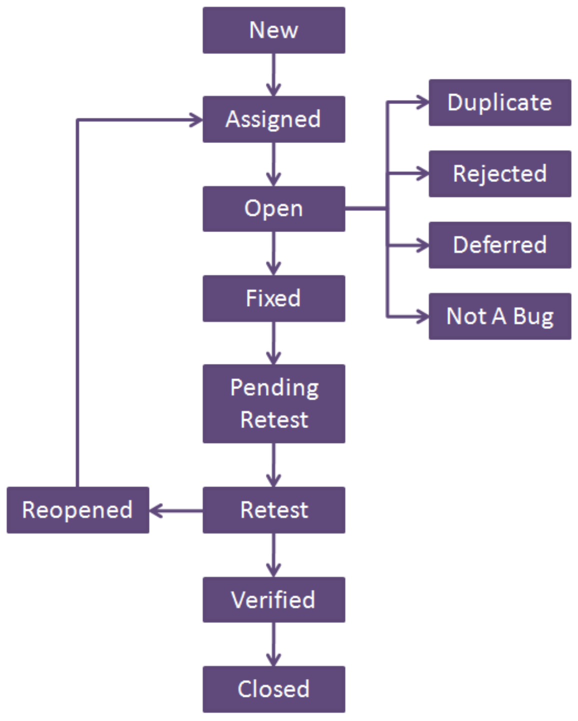
For process improvement, everyone in the project needs to look back and check from where the defect was originated. Based on that you can make changes in the validation process, base-lining document, review process which may catch the defects early in the process which are less expensive.

Defect life Cycle
A Defect life cycle, also known as a Bug life cycle, is a cycle of a defect from which it goes through covering the different states in its entire life. This starts as soon as any new defect is found by a tester and comes to an end when a tester closes that defect assuring that it won't get reproduced again.
Defect Life Cycle or Bug Life Cycle in software testing is the specific set of states that defect or bug goes through in its entire life. The purpose of Defect life cycle is to easily coordinate and communicate current status of defect which changes to various assignees and make the defect fixing process systematic and efficient.

- **Defect life cycle** is a cycle which a defect goes through during its lifetime. It starts when defect is found and ends when a defect is closed, after ensuring it's not reproduced. **Defect life cycle** is related to the bug found during testing.
- The defect life cycle can vary from organization to organization and also from project to project based on several factors like organization policy, software development model used (like Agile, Iterative), project timelines, team structure etc.

Defect Status
Defect Status or Bug Status in defect life cycle is the present state from which the defect or a bug is currently undergoing. The goal of defect status is to precisely convey the current state or progress of a defect or bug in order to better track and understand the actual progress of the defect life cycle.

The bug has different states in the Life Cycle. The Life cycle of the bug



Bug / Defect Lifecycle

- **New:** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
- **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team
- **Open:** The developer starts analysing and works on the defect fix
- **Fixed:** When a developer makes a necessary code change and verifies the change, he or she can make bug status as "Fixed."
- **Pending retest:** Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the software testing remains pending from the testers end, the status assigned is "pending retest."
- **Retest:** Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to "Re-test."
- **Verified:** The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is "verified."
- **Reopen:** If the bug persists even after the developer has fixed the bug, the tester changes the status to "reopened". Once again the bug goes through the life cycle.
- **Closed:** If the bug is no longer exists then tester assigns the status "Closed."
- **Duplicate:** If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to "duplicate."
- **Rejected:** If the developer feels the defect is not a genuine defect then it changes the defect to "rejected."
- **Deferred:** If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status "Deferred" is assigned to such bugs
- **Not a bug:** If it does not affect the functionality of the application then the status assigned to a bug is "Not a bug".

Classification of bug

Defects are classified from the QA team perspective as Priority and from the development perspective as Severity (complexity of code to fix it). These are two major classifications that play an important role in the timeframe and the amount of work that goes in to fix defects.

Severity

Bug/Defect severity can be defined as the impact of the bug on the application. It can be Critical, Major or Minor. In simple words, how much effect will be there on the system because of a particular defect?

For example: If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of application crashing is severe. So the severity is high but priority is low.

What is Priority?

Priority is defined as the order in which a defect should be fixed. Higher the priority the sooner the defect should be resolved.
Defect priority can be defined as an impact of the bug on the customers' business. Main focus on how soon the defect should be fixed. It gives the order in which a defect should be resolved. Developers decide which defect they should take up next based on the priority. It can be High, Medium or Low.
For example: If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.

Severity Listing

Severity can be categorized in the following ways −

- **Critical /Severity 1** − Defect impacts most crucial functionality of Application and the QA team cannot continue with the validation of application under test without fixing it. For example, App/Product crash frequently.
- **Major / Severity 2** − Defect impacts a functional module; the QA team cannot test that particular module but continue with the validation of other modules. For example, flight reservation is not working.
- **Medium / Severity** 3 − Defect has issue with single screen or related to a single function, but the system is still functioning. The defect here does not block any functionality. For example, Ticket# is a representation which does not follow proper alpha numeric characters like the first five characters and the last five as numeric.
- **Low / Severity 4** − It does not impact the functionality. It may be a cosmetic defect, UI inconsistency for a field or a suggestion to improve the end user experience from the UI side. For example, the background colour of the Submit button does not match with that of the Save button.

Priority Types

Types of Priority of bug/defect can be categorized into three parts:
Low − this defect can be fixed after the critical ones are fixed.
Medium − during the normal course of the development activities defect should be resolved. It can wait until a new version is created
High: The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed

Defect Report Template

In most companies, a defect tracking tool is used and the elements of a defect report can vary from one tool to the other. However, in general, a defect report can consist of the following elements.

ID	Unique identifier given to the defect. (Usually, automated)
Project	Project name.
Product	Product name.
Release Version	Release version of the product. (e.g. 1.2.3)
Module	Specific module of the product where the defect was detected.
Detected Build Version	Build version of the product where the defect was detected (e.g. 1.2.3.5)

Summary	Summary of the defect. Keep this clear and concise.
Description	Detailed description of the defect. Describe as much as possible but without repeating anything or using complex words. Keep it simple but comprehensive.
Steps to Replicate	Step by step description of the way to reproduce the defect. Number the steps.
Actual Result	The actual result you received when you followed the steps.
Expected Results	The expected results.
Attachments	Attach any additional information like screenshots and logs.
Remarks	Any additional comments on the defect.
Defect Probability	Probability of the Defect.
Defect Severity	Severity of the Defect.
Defect Priority	Priority of the Defect.
Reported By	The name of the person who reported the defect.
Assigned To	The name of the person that is assigned to analyse/ fix the defect.
Status	The status of the defect.
Fixed Build Version	Build version of the product where the defect was fixed (e.g. 1.2.3.9)

<u>Techniques to Identify Defects:</u>
Different techniques are used to identify defect. These techniques are categorized into three categories as given below:

1. **Static Techniques:**
   Static technique, as name suggests, is a technique in which software is tested without any execution or program or system. In this, software products are tested or examined manually, or with help of different automation tools that are available, but it's not executed.
   Different type of causes of defects can be found by this technique such as :

- Missing requirements
- Design defects
- Deviation from standards
- Inconsistent interface specification
- Non-maintainable code
- Insufficient maintainability, etc.

2. **Dynamic Techniques:**
   Dynamic technique, as name suggests, is a technique in which software is tested by execution of program or system. This technique can only be applied to software code as in this technique, testing is done only by execution of program or software code. Different types of defects can be found by this technique such as :

- **Functional defects –**
  These defects arise when functionality of system or software does not work as per Software Requirement Specification (SRS). Defects that are very critical largely affect essential functionalities of the system that in turn affect a software product or its functionality. Software product might not work properly and stop working. These defects are simply related to working of system.
- **Non-functional defects –**
  A defect in software products largely affects its non-functional aspects. These defects can affect performance, usability, etc.

3. **Operational Techniques:**
   Operational techniques, as name suggests, are a technique that produces a deliverable i.e. Product. Then user, customer, or control personnel identify or found defects by inspection, checking, reviewing, etc. In simple words, defect is found as a result of failure.

   Different techniques to find the defects are:
   a) Quick Attacks:
   b) Equivalence and Boundary Conditions
   c) Common Failure Modes
   D) State-Transition Diagrams
   e) Use Cases
   F) Code-Based Coverage Models
   g) Regression and High-Volume Test Techniques

   <u>Estimate Expected Impact:</u>
   Estimating expected impact of defect on cost is also important part of defect prevention. Whenever defect is encountered in system, main questions arises is what are measures that are needed to be taken to resolve it and amount required to resolve it. More critical defect is, more will be amount required to resolve it as more effort and resources will be required to resolve it. Therefore, it's better to estimate expected impact of defect on cost if defect somehow encounters or occurs. After that, some measures should be taken to minimize their occurrence.
   Minimize Expected Impact:
   After identifying and estimating impact of critical defects, one should take some measures actions to minimize or eliminate defects permanently before its occurrence. Minimizing expected impact is also important part of defect prevention. If one could not eliminate defect, then he/she should try to reduce possibility of its occurrence and its impact
   <u>Defect Reporting</u>
   Defect Reporting in software testing is a process in which test managers prepare and send the defect report to the management team for feedback on defect management process and defects' status. Then the management team checks the defect report and sends feedback or provides further support if needed. Defect reporting helps to better communicate, track and explain defects in detail.
   The management board has right to know the defect status. They must understand the defect management process to support you in this project. Therefore, you must report them the current defect situation to get feedback from them.

__UNIT 06 Testing tools and measurements__

Manual Testing

Manual Testing
Manual testing is a software testing process in which test cases are executed manually without using any automated tool. All test cases executed by the tester manually according to the end user's perspective. It ensures whether the application is working, as mentioned in the requirement document or not. Test cases are planned and implemented to complete almost 100 percent of the software application. Test case reports are also generated manually.

Limitations of manual testing

- Manual testing requires more resources and time as the complete process is manual.
- It's prone to human error for the cases where intensive data calculation is involved.
- Testing is a repetitive process. For each release you have certain test cases which are executed just to make sure that nothing is broken due to new features. These are called regression cases. To execute same cases again and again is boring.
- It's not suitable for large scale and time bounded projects.
- Verifying a large amount of data is not possible.
- Performance testing is impractical manually.
- GUI difference, component size are difficult to verify manually.
- GUI objects size difference and color combination etc is not easy to find out in manual testing.
- Load testing and performance testing is not possible in manual testing.
- Running test manually is very time consuming job.
- Regression Test cases are time consuming if it is manual testing.

Automation Testing

Automation testing is the application of tools and technology to testing software with the goal of reducing testing efforts, delivering capability faster and more affordably. It helps in building better quality software with less effort.
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.

Need for automation testing

Why Automation Testing?
Test automation has many benefits for app testing cycles. This allows you to build better apps with less effort. Also, it is less time-consuming.

1. Faster Feedback Cycle
Test automation helps you reduce the feedback cycle and bring faster validation for phases in the development of your product.

2. Team Saves Time
By automating your testing procedure, your team has to spend less time validating newly developed features. It also improves communication with other departments like marketing, design, or product owners who rely on the results of these tests. These departments can easily check the logs of the automated tests and see what’s happening.

3. Reduced Business Expenses

When using an automated test environment, your company will save money as fewer resources are spent on testing your product. The idea is that you should not be doing any manual testing. Over the course of a whole project, this can make a big difference.

4. Higher Test Coverage
Manual testing puts limits on how many tests you can verify. Automation allows you to spend time writing new tests and adding them to your automated test suite. This increases the test coverage for your product, so more features are properly tested resulting in a higher quality application.

5.Reusability: The scripts are reusable and you don't need new scripts every time. Also, you can redo the steps that are exactly as the previous ones.

6. Bugs: Automation helps you find bugs in the early stages of software development, reducing expenses and working hours to fix these problems as well.

7. Faster Time to Market
This reduces the feedback and testing cycle and allows companies to bring their products to the market faster.

8 Better Insights: Automated testing provides better insights than manual testing when some tests fail. Automated software testing not only gives insights into the application but also shows you the memory contents, data tables, file contents, and other internal program states. This helps developers determine what's gone wrong.

9 Improved Accuracy
Even the best testing engineer will make mistakes during manual testing. Especially when testing a complex use case, faults can occur. On the other side, automated tests can execute tests with 100-percent accuracy as they produce the same result every time you run them.

10 Automated Testing Provides More Features
An automated test suite can help you with more features—for example, simulating thousands of virtual users interacting with your web application in order to see how the application behaves. It's impossible to simulate this kind of behavior by doing manual testing. Features like this save developers a lot of time!

11. Less Stress on QA Team
By implementing an automated testing strategy, you allow your QA team to spend time on tasks other than manual testing.
For many QA engineers, testing automation creates the opportunity to build new tools to further optimize the current testing suite or extend it with new features.

12 Fewer Human Resources: You just need a test automation engineer to write your scripts to automate your tests, instead of a lot of people doing boring manual tests over and over again.

13. Eliminate Human Error
Manual testing opens up the opportunity for humans to make mistakes. Especially for complex scenarios, it makes sense to use test automation to avoid mistakes. You can still make mistakes, even with test automation. However, the rate of mistakes is significantly lower when using test automation for your test suite

Automation Testing vs Manual Testing

Automation Testing	Manual Testing
Automated testing is more reliable. It performs same operation each time. It eliminates the risk of human errors.	Manual testing is less reliable. Due to human error, manual testing is not accurate all the time.
Initial investment of automation testing is higher. Investment is required for testing tools. In the long run it is less expensive than manual. ROI is higher in the long run compared to Manual testing.	Initial investment of manual testing is less than automation. Investment is required for human resources. ROI is lower in the long run compared to Automation testing.
Automation testing is a practical option when we do regressions testing.	Manual testing is a practical option where the test cases are not run repeatedly and only needs to run once or twice.
Execution is done through software tools, so it is faster than manual testing and needs less human resources compared to manual testing.	Execution of test cases is time consuming and needs more human resources
Exploratory testing is not possible	Exploratory testing is possible
Performance Testing like Load Testing, Stress Testing etc. is a practical option in automation testing.	Performance Testing is not a practical option in manual testing
It can be done in parallel and reduce test execution time.	Its not an easy task to execute test cases in parallel in manual testing. We need more human resources to do this and becomes more expensive.
Programming knowledge is a must in automation testing	Programming knowledge is not required to do manual testing.
Build verification testing (BVT) is highly recommended	Build verification testing (BVT) is not recommended
Human intervention is not much, so it is not effective to do User Interface testing.	It involves human intervention, so it is highly effective to do User Interface testing.

Automation Testing tools

Katalon

Launched in 2015, Katalon is a free-licensed, cross-browser tool that enables running automation testing for APIs, Web interfaces, and mobile (Android and iOS). Additionally, this tool provides analysis reports and test recording.

Selenium

Selenium is a popular open-source (released under the Apache License 2.0) automation testing framework. Originally developed in 2004 by Jason Hugging, Selenium remains a widely-known and used tool for testing web applications. It operates across multiple browsers and platforms (macOS, Windows, and Linux) and can write tests in various programming languages, such as Python, Java, C#, Scala, Groovy, Ruby, Perl, and PHP.

Bugzilla?

Bugzilla is an open-source issue/bug tracking system that allows developers to keep track of outstanding problems with their product. It is written in Perl and uses MYSQL database.
Bugzilla is a Defect tracking tool, however, it can be used as a test management tool as such it can be easily linked with other Test Case management tools like Quality Center, Testlink etc.

QTP is an automated functional Testing tool that helps testers to execute automated tests in order to identify any errors, defects or gaps in contrary to the expected results of the application under test. It was designed by Mercury Interactive and later on acquired by HP and now MicroFocus. Full form of QTP is QuickTest Professional while UFT means Unified Functional Testing.

Why QTP is the best testing tool?

It is an icon-based tool that automates the regression and Functional Testing of an application
Both technical, as well as a non-technical tester, can use Micro Focus QTP
It provides both features- Record as well as Playback
We can test Desktop as well as the Web-based applications
It allows Business Process Testing (BPT)
QTP Testing is based on scripting language VB script

Selenium

Selenium is often used for regression testing. It offers testers a playback tool that allows them to record and playback regression tests. In fact, Selenium is not a single tool but rather a suite of software that includes various tools (or components):
Selenium IDE (Integrated Development Environment)
Selenium WebDriver
Selenium client API
Selenium Remote Control
Selenium Grid

MANTIS is an open source bug tracking software that can be used to track software defects for various software projects. You can easily download and install the Mantis for your use. Mantisbt now also provides a hosted version of the software. You can easily customize Mantis to map your software development workflow.
Some salient features of Mantis Bt are
Email notifications: It sends out emails of updates, comments, resolutions to the concerned stakeholders.
Access Control: You can control user access at a project level
Customize: You can easily customize Mantis as per your requirements.
Mobile Support: Mantis supports iPhone, Android, and Windows Phone Platforms.

LambdaTest

LambdaTest is a cloud-based automation testing tool for desktop and mobile applications. This tool allows for manual and automated cross-browser testing across more than 2000 operating systems, browsers, and devices.
LambdaTest allows testers to record real-time browser compatibility testing. Plus, it enables screen recording and automated screenshot testing on several combinations at a time.

Ranorex
Ranorex is a test automation tool for web, desktop, and mobile. This tool provides numerous benefits, such as codeless test creation, recording and replaying testing phases, and reusable test scripts.

Appium
Appium is an open-source test automation framework. This framework supports multiple programming languages (Python, Java, PHP, JavaScript, etc.) for writing tests and can integrate CI/CD tools (e.g., Jenkins).

Eggplant
Eggplant was developed by TestPlant to provide testers the possibility to execute different types of testing. Similar to Selenium, Eggplant is not a single tool but rather a suite of tools for automation testing, and each tool performs different types of testing.

7. Kobiton
Kobiton is a cloud-based platform that can perform both manual and automated mobile and web testing. Its AI-driven scriptless approach can automate performance, visual and UX, functional, and compatibility testing. In addition, Kobiton offers automated crash detection, which ensures comprehensive quality.

Features of test tools
Static Test Tools: These tools do not involve actual input and output. Rather, they take a symbolic approach to testing, i.e. they do not test the actual execution of the software. These tools include the following: ,
1) Flow analyzers: They ensure consistency in data flow from input to output.
2) Path tests: They find unused code and code with contradictions.
3) Coverage analyzers: It ensures that all logic paths are tested.
4) Interface analyzers: It examines the effects of passing variables and data between modules.

Dynamic Test Tools: These tools test the software system with 'live' data. Dynamic test tools include the following
1) Test driver: It inputs data into a module-under-test (MUT).
2) Test beds: It simultaneously displays source code along with the program under execution.
3) Emulators: The response facilities are used to emulate parts of the system not yet developed.
4) Mutation analyzers: The errors are deliberately 'fed' into the code in order to test fault tolerance of the system.

Selecting a testing tools
To select the most suitable testing tool for the project, the Test Manager should follow the below tools selection process

Step 1) Identify the Requirement for Tools
- How can you select a testing tool if you do not know what you are looking for?
  You to precisely identify your test tool requirements. All the requirement must
  be documented and reviewed by project teams and the management board.

Step 2) Evaluate the Tools and Vendors
- After baselining the requirement of the tool, the Test Manager should
- Analyze the commercial and open source tools that are available in the market, based on the project requirement.
- Create a tool shortlist which best meets your criteria
- One factor you should consider is vendors. You should consider the vendor's reputation, after sale support, tool update frequency, etc. while taking your decision.

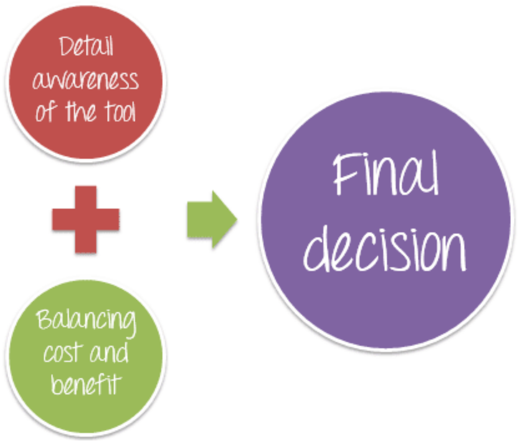
- Evaluate the quality of the tool by taking the trial usage & launching a pilot. Many vendors often make trial versions of their software available for download

Step 3) Estimate Cost and Benefit

- To ensure the test tool is beneficial for business, the Test Manager have to balance the following factors:
- Cost
- Value
- benefit

  However, after discussing with the software vendor, you found that the cost of this tool is too high compare to the value and benefit that it can bring to the teamwork.
- In such a case, the balance between cost & benefit of the tool may affect the final decision

Step 4) Make the Final Decision



To make the final decision, the Test Manager must have:

- Have a strong awareness of the tool. It means you must understand which is the strong points and the weak points of the tool
- Balance cost and benefit.
- Even with hours spent reading software manual and vendor information, you may still need to try the tool in your actual working environment before buying the license.
- You should have the meeting with the project team, consultants to get the deeper knowledge of the tool

Testing Metrics are the quantitative measures used to estimate the progress, quality, productivity and health of the software testing process. The goal of software testing metrics is to improve the efficiency and effectiveness in the software testing process and to help make better decisions for further testing process by providing reliable data about the testing process.

Why Test Metrics are Important?

"We cannot improve what we cannot measure" and Test Metrics helps us to do exactly the same.

- Take decision for next phase of activities
- Evidence of the claim or prediction
- Understand the type of improvement required
- Take decision or process or technology change

Types of Test Metrics

Process Metrics: It can be used to improve the process efficiency of the SDLC (Software Development Life Cycle)

Product Metrics: It deals with the quality of the software product

Project Metrics: It can be used to measure the efficiency of a project team or any testing tools being used by the team members