INDEX

Manual Testing Modules

Software Testing

Software testing is an activity to find the defect & bugs from the software the process to verifying that a software application does what it is supposed to do.

Software Testing Era

In 1951, Joseph Juran, who is now considered to be the father of software testing, for the first time marked the importance of software quality assurance in his book "Quality control handbook"
Testing gurus like Hetzel and Dave Gelprin divide testing into five significant

Eras:

1. **Debugging-oriented era:** This phase was during the early 1950s, when there was no distinction between testing and debugging. The focus was on fixing bugs. Developers used to write code, and when faced with an error would analyze and debug the issues. There was no concept of testing or testers.

   (In1957, Charles L Baker described the difference between debugging and testing in his book Digital Computer Programming)

2. **Demonstration-oriented era:** From 1957 to 1978, the distinction between debugging and testing was made and testing was carried out as a separate activity

3. **Destruction-oriented era**: From 1979 to 1982, the focus was on breaking the code and finding the errors in it

4. **Evaluation-oriented era:** From 1983 to 1987, the focus was on evaluating and measuring the quality of software

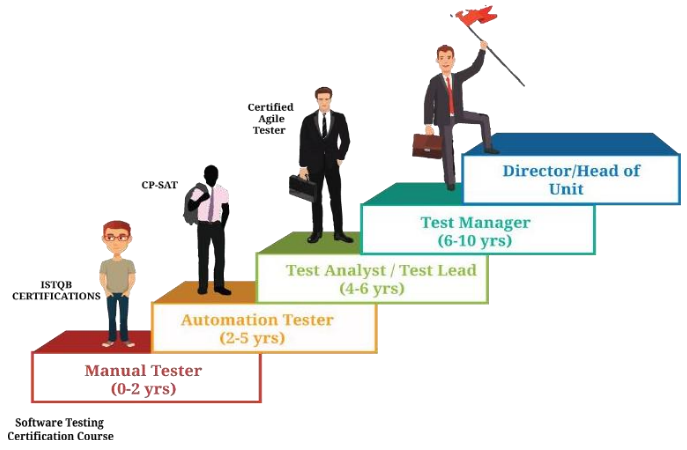
5. **Prevention-oriented era:** 1988 to 2000 saw a new approach, with tests focusing on demonstrating that software met its specification, detecting faults and preventing defects.

The year 2004 saw a major revolution in testing, with the advent of automation testing tools like Selenium.

Career in Software Testing

Choosing Software Testing as a career is always going to be a great deal for you. Choosing the best Software Testing Certification Course will help you to learn the subject clearly from scratch.
Software testing plays most significant role for application/product implementations. Businesses have realized the importance of structured testing of applications.

Testing is very interesting & innovative job profile. It is looked as a good professional career.

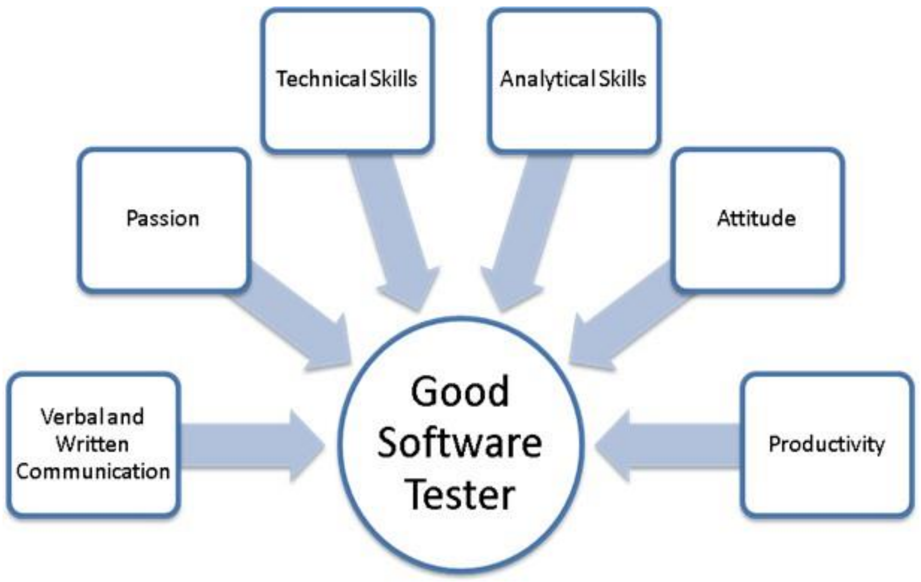


Software Testers are the only people who win the customer trust by performing good testing and making the application most potent. Your job is not only to find the bug from the application but is about ensuring customer satisfaction which is the top priority in the whole software industry. Don't worry if you don't have programming skills. You can always learn the things if you have interest for it.

If you've ever wondered how to become a software tester, you'll be glad to know that you don't need to have the experience of a developer, but you do have to be a quick learner who is very detail oriented. Software testing is roughly as old a profession as development, and it has a very similar career trajectory to that of the software developer.

You also get an early look at the latest in technology and have a hand in making sure, that a program is as defect free as possible. Finally, with software being an area that will always grow, software testers have one of the most secure jobs in the tech industry.

Skills required to become a Software Tester



Analytical skills: A good software tester should have sharp analytical skills. Analytical skills will help break up a complex software system into smaller units to gain a better understanding and create test cases.

Communication skill: A good software tester must have good verbal and written communication skill.

Time Management & Organization Skills: A software tester must efficiently manage workload, have high productivity, exhibit optimal time management, and organization skills.

GREAT Attitude: To be a good software tester you must have a GREAT attitude. An attitude to ‘test to break’, detail orientation, willingness to learn and suggest process improvements. Your attitude must reflect a certain degree of independence where you take ownership of the task allocated and complete it without much direct supervision.

Passion: To Excel in any profession or job, one must have a significant degree of the passion for it. A software tester must have a passion for his / her field.

Module 1: Introduction to Testing

1.1 What is S/W Testing:

"Testing is the process which is used to check completeness, correctness and quality of the application"

Software testing is the process of evaluating and analyzing whether the software product is working as expected and doing what it is supposed to do.

Prospectives of Software Testing:

Completeness-Check whether all the requirement that is mention by the client are fulfill.

Correctness-Check software is working correctly as per client requirement.

Quality-Check software is meeting the client requirement i.e. (expectation, security, performance, budget etc.).

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

"Software testing is a process of executing a program or application to finding the software defect. it is a process of "verifying" and "Validating" that a software program or application meets the business and technical requirement".

Q) Why there is need of software testing?

1. Software testing is done to make sure that the software that is created actually does what's supposed to do.

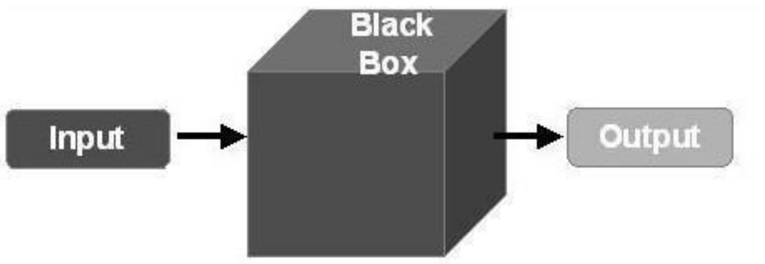
2. Software testing is required to point out the defect and error that were made during the development.
3. It's important to ensure that the application should not result into any failure because it can be very expensive in the future.
4. Every software is developed to support business, if software does not work properly there will be huge loss in the business to avoid this, we need to test software working as per need.
5. The first impression is really important and if you fail to give the same, users are going to find another product which will accomplish all the requirements.
6. Testing improves software reliability and high-quality applications are delivered with few errors. A system that meets or even exceeds customer expectations lead to potentially more sales and greater market share

1.2 Test Methods:

- **Black Box Testing**
- **White Box Testing**

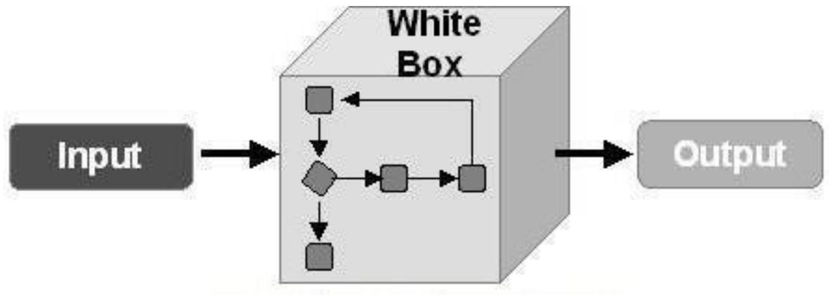
Black Box Testing:

1. **The main focus of Black Box Testing is on the functionality of the system as a whole. It is performed by the Software Tester.**
2. The Black Box testing method is used to test an application based on the requirements specified by the customer or user.
3. Black Box Testing does not focus on the internal structure/code of the application.
4. In this, focus on what is the output against the input?
5. Also Called as Behavioral Testing, Specification based testing, closed-box testing.



White Box Testing:

- White Box testing test the internal structure or working of the application.
- White box testing is used for verification of internal mechanisms i.e. how the output is achieved?
- Needs to have a look inside the source code and find out which code is behaving inappropriately. Done by Developer.
- Also Called as Structural Testing /Glass Box Testing/Open Box Testing/ Clear Box Testing.



1. 3 Strategies of Testing:

1. Negative Testing-

- Negative testing is the process of applying as much creativity as possible and validating the application against invalid data
- Motive is detecting possible application crashes in wrong situations.
- This is to test the application that does not do anything that it is not supposed to do so.

2. Positive Testing-

- Is testing process where the system is validated against the valid input data. Determines that your application works as expected.
- Motive to prove that a given product meets the requirements and specifications

1.4 Types of Testing:

- **Functional Testing**
- **Non- Functional Testing**

1. Functional Testing:

- Functional Testing tests a Function to be performed by the Software.
- The functions are “what” the system does.
- The main objective of functional testing is to test the functionality of the software is working correctly.

- The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output

Example (Consider banking application);

1. Check the functionality of login user should be logged into the session.
2. Check the functionality of transaction, Transaction detail should be shown.
3. Check the functionality of Account Balance & Amount transfer.

2. Non- Functional Testing

- **Non-Functional testing** is a type of testing in which the performance or usability and behavior of a software or application is tested under different circumstances
- Non-functional testing, is focused on testing the non-functional aspects of the software, such as **performance, security, usability, reliability, and compatibility.**
- Non-functional testing helps to ensure that the software meets the quality standards and performs well under different condition
- It means "How" well system responds.
- **Example;**
    1. Checking the UI.
    2. Checking the Color, Spelling, Border, Alignment.
    3. Checking the response time of application.

1.5 <u>Defect:</u>

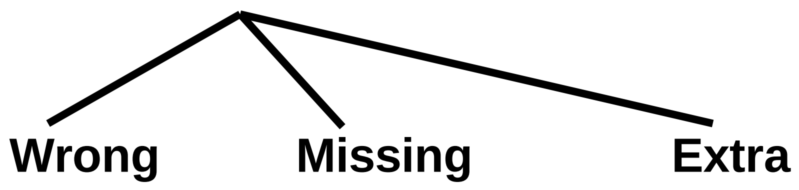
What is defect?

- **"Difference between the expected and actual behavior of the software".** When actual result deviates from the expected result while testing a software application then it results into a defect.

► <u>Types of Defects</u>

- **Functional Defect**
- **Non-Functional**

<u>Category of Defect</u>:



- **Functional Wrong Defect**
  - The Software does something that the software specification says it should not do.
- **Functional Missing Defect**

The Software does not do something that the software specification says it should do.

- **Functional Extra Defect**

The Software does something that is not mention in the software specification.

Example: -Please refer class notes.

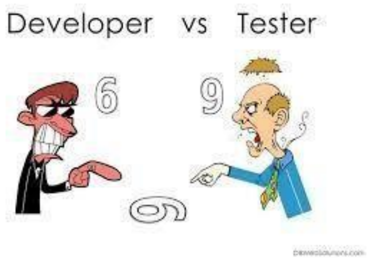
2. Non-Functional Defect:

Hard to use, Hard to Learn and Understand (GUI Related issue), Performance issue (Slow) etc.

► Synonyms of Defects

- Incident
- Bug
- Failure
- Error

1.6 Debugging and Testing:



Debugging:

- Done by the Developer.
- Debugging is an activity that finds, analyzes and removes the error that we found during the testing process.
- Debugging cannot be automated.
- The main objective is to find the exact root cause at code level to fix the errors during the testing.

Testing:

- Done by the Testers
- Testing is the process using which we find errors and bugs.
- Testing can be automated.
- The main objective is to find bugs and errors in an application which get missed during the unit testing by the developer.

1.6 Software Development Life Cycle:

SDLC is a process that defines the various stages involved in the development of software for delivering a high-quality product. It defines the steps involved in the development of software at each phase. It covers the detailed plan for building, testing, deploying and maintaining the software.

1.7 The Phases of SDLC:

- Requirement Gathering and Analysis
- Planning
- Designing
- Coding
- Testing
- Release and Maintenance

1. Requirement Gathering and Analysis:

- During this phase requirement is gathered by Business Analyst from customer.
- Analyze the requirements and BRS is prepared.
- After BRS document, SRS document is prepared by System Analyst.
- In analysis phase we are doing detail study of the doc i.e., Requirement specification (BRS and SRS).
- In Analysis phase we are doing feasibility study i.e. (Time feasibility, cost feasibility, technical feasibility, resource feasibility etc.).

2. Planning:

The planning phase includes resource allocation, capacity planning, project scheduling, set goals, and identify risks and cost estimation. Project Plan document will be prepared by project manager/senior member.

3. Design:

This phase of the SDLC starts by turning the software specifications into a design. Design act as blue print of Software.

Two Types of Design Documents Created as below.

1. HLD (High Level Design)/ Global Design for Entire System
2. LLD (Low Level Design)/ Detail Design for component.

4. Coding:

- In this stage of SDLC the actual development starts and the product is built.
- During this phase programs are written in selected Language. The programming language is chosen with respect to the type of software being developed.
- Source Code document is prepared.

5. Testing:

- In Testing Phase Testers Analise, the Requirements
- During this phase test cases are written and executed.
- Tester makes sure that system fulfils the customer needs.
- If any defect is found it is forwarded to the Developer to fix it.

6. Release and Maintenance:

- Once the product is tested and ready to be deployed it is released in live environment at Client side.
- The procedure where the care is taken for the released product is known as maintenance.
- If any Defect found by user while using that product it will be handled by Maintenance Team
- In the maintenance phase, among other tasks, the team fixes bugs, resolves customer issues, and manages software changes
- These changes are handled by configuration management team.

7. There are two types of changes:

Changes

Corrective

Any defect found at client side is fixed or correct by the development company.

Adaptive

Changes of requirement i.e. Addition of new feature, changing of platform, OS etc.

1.8 Software Testing Life Cycle:

STLC is a testing process which is executed in a sequence, in order to meet the quality goals, STLC is followed by the testing team

There are 6 phases in the Software Testing Life Cycle or STLC life cycle

1. **Requirement analysis**
2. **Test Planning**
3. **Test Analysis and Design**
4. **Environment Setup**
5. **Test Execution & Report**
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.

1. Requirement Analysis:

1. This is the very first phase of Software testing Life cycle (STLC).

In this phase testing team goes through the Requirement document with both Functional and non-functional details in order to identify the testable requirements.

Here BRS & SRS Documents are referred for the proper understanding about functional requirements or client expectation

2. In case of any confusion the QC team may setup a meeting with the clients and the stakeholders (Technical Leads, Business Analyst, System Architects and Client etc.) in order to clarify their doubts.

Once the QC team is clear with the requirements, they will document the acceptance Criteria and get it approved by the Customers.

Deliverables: List of all testable requirements, Automation feasibility report (if applicable)

2. Test Planning and Control:

1. Test planning activity is very important where testing strategy is defined.
2. Test Plan Document is Created.
3. Done by Test Manager / Test Lead / Project Manager.
4. SRS/BRS, Project plan and Design documents (Optional) documents are referred.
5. During Test Planning Scope of Testing, Schedule, Roles & Responsibilities etc. is decided.
6. Test control is the ongoing activity of comparing actual progress against the plan, and reporting the status, including deviations from the plan.
7. It involves taking necessary actions to meet the mission and objectives of the project.
8. Test Manager monitors, guides and controls.

Deliverables: (Outcome) of Test Planning phase are:

- ➢ Test Plan document

3. Test Analysis and Design:

1. Reviewing and understanding the test basis (BRS, SRS).
2. Evaluating testability of the test basis and test objects.
3. Writing and prioritizing test conditions for the creation of Test scenario document & Test Cases based on analysis.
4. Done by Tester.

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

- Creation of Test Scenario
- Creation of test cases
- Creation of test scripts if required
- Verification of test cases and automation scripts

Deliverables (Outcome) of Test Case Development phase are:

- Test Scenario document
- Test Cases
- Test scripts (for automation if required)

4. Test Environment setup

This phase includes the setup or installation process of software and hardware which is required for testing the application.

After setting up the required software and hardware the installation of build is tested.

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
- Installation of build.

Deliverables (Outcome) of Test Environment Setup phase are:

- Test Environment setup is ready

5. Test Execution & Report:

1. Test implementation and execution is the activity where test procedures or scripts are specified.
2. Environmental set up will be done and the tests are run.
3. Comparing Expected Result with Actual Result.
4. Defect Reporting and fixing will be done.
5. Checking test logs against the exit criteria specified in test planning.
6. Assessing if more tests are needed or if the exit criteria specified should be changed.
7. Writing a test summary report for stakeholders against the defined objectives.

6. Test Closure Activities:

1. The Test Cycle Closure phase is the end goal of execution of the testing phase that includes the summary of all the tests conducted during the software testing life cycle.
2. Finalizing and archiving test ware, the test environment and the test infrastructure for later reuse.
3. Handing over the test ware to the maintenance organization.
4. Store and Share Learning's /Experience.

1.9 Quality:

Quality is meeting the requirement, expectation and needs of the customer being free from defect, lacks and substantial variants. There are standards that needed to follow to satisfy the customer needs.

➢ Quality Management System (QMS)

- These are the people they will decide which standard to be followed through the project standard such as IEEE, ISO etc.
- The management of every software development organization has to decide which quality system has to be implemented.
- They will establish the process.

➢ Quality Assurance (QA)

- The process that has established by the QMS team is checked by the QA.
- QA ensures that the approaches, techniques, methods and process are designed for the project are implemented correctly.

➢ Quality Control (QC)

- The main purpose of the Quality Control process is to ensure that the product (software) meets the actual requirements of the users.
- It aims to identify and fix the defects.

1.10 Software Testing Principles:

Principle 1: Testing Shows Presence of Defect

Testing can show that defects are present but cannot prove that there are no defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.

Principle 2: Exhaustive Testing is impossible

Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases.

Principle 3: Early Testing

To find defects early, testing activities should start as early as possible in the software or system development life cycle and should be focused on defined objectives.

Principle 4: Defect Clustering

A small number of modules usually contain most of the defects or is responsible for the most failures. Approximately 80% of the defects are found in 20% of the modules.

Principle 5: Pesticide Paradox

If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects. To overcome this “pesticide paradox”, the test cases need to be regularly reviewed and revised, and new and different tests need to be written.

Principle 6: Testing is Context Dependent

All software is not developed the same. You can use a different approach, method, technique, and type of test depending on the type of application For example, safety-critical software is tested differently from an ecommerce site.

Principle 7: Absence of Error fallacy

Finding and fixing defects does not help if the system built is unusable and does not fulfil the users’ needs and expectations. Means if system is not fit for use, there is no point in testing, finding defects and fixing it.

Module 2: Software Development Model

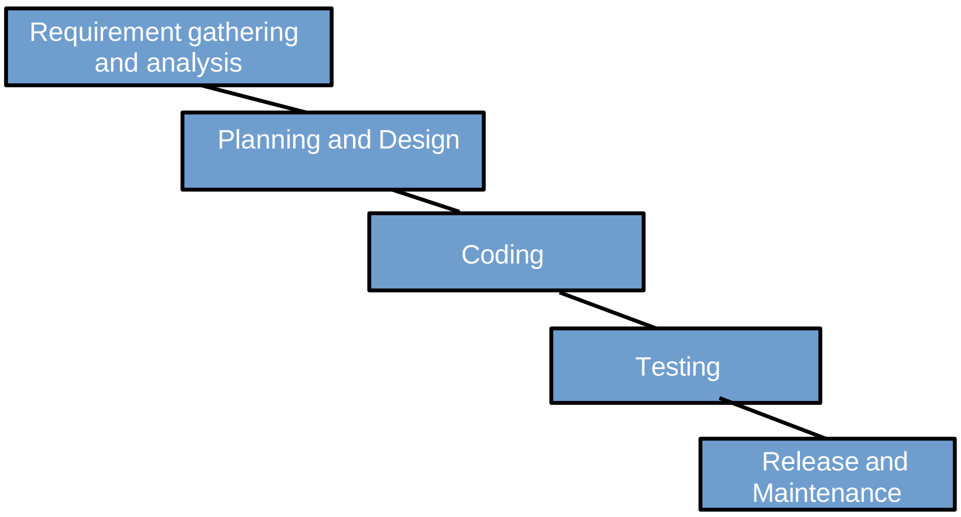
2.1 Different Software Development Models

2.2 Sequential Model Incremental/ Iterative Model

· Waterfall model

· Agile model
- RAD model
- Spiral model

2.2 Sequential Model (Waterfall):



The Waterfall Model was 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.

1. In a waterfall model, each phase must be completed fully before the next phase can begin. This type of software development model is basically used for the project which is small and there are no uncertain requirements.

2. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In this model software testing starts only after the development is complete. In waterfall model phases do not overlap.
3. Recommended for small scale, Low budget and short-term projects.

2.3 Iterative /Incremental Model:

- Software is developed step by step.
- First increment is the core product.
- After the first increment is ready it is released for the use of the client Asa result of user the plan is prepared for the next increment.
- Modification is made to the core product to meet the needs of the customer.
- Client gets Early working software Example: Spiral and Agile model

2.4 Agile (Iterative /Incremental Model)

Agile Process / Agile Methodology: It is an iterative and Incremental SDLC model. Which is focused on quick delivery.

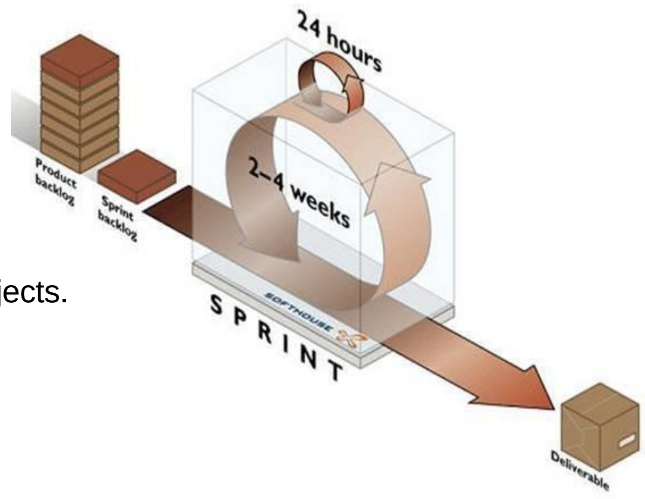
Agile Principle:

- Clients no need to wait for long time for product delivery.
- Requirements can accept/accommodate between the development process.
- Software will divide in small piece, that piece of software will develop, test and release to the customer.

In agile process will be good communication between customer, BA, Developers & testers.
Agile process is the latest development process, followed by lot of famous companies like Facebook, Google etc.
More focused on meeting and discussions, not on documentation

Advantages:

- ➢ Customer no need to wait for long time.
- ➢ Requirement changes allowed in any stage of development
- ➢ Good communication between team.
- ➢ Recommended for medium to large size projects.
- ➢ Agile works for quick deliveries.
- ➢ Focuses on Quality as well as Time.
- ➢ Weekly Interaction with the Client is done.

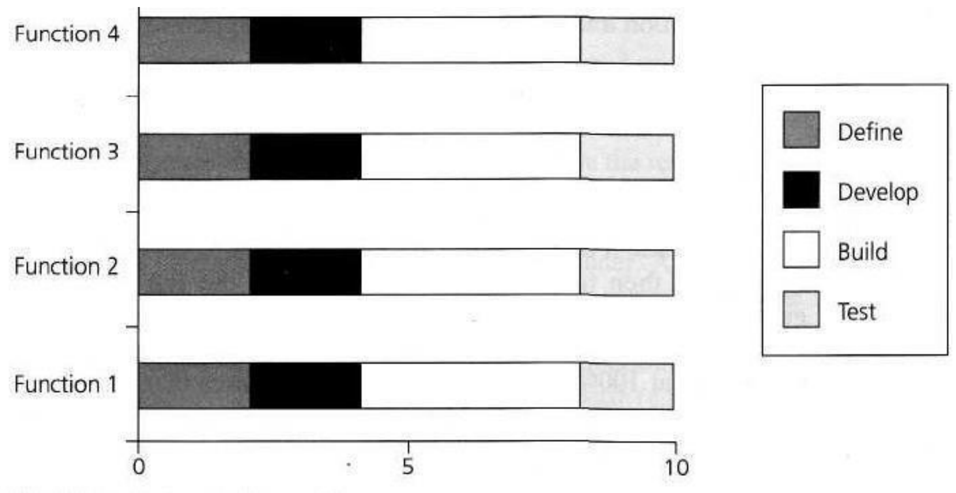


Additional requirements can be added during the development

Scrum Agile Testing:

Scrum is an iterative and incremental framework through which we build software product by following Agile principles that helps teams deliver high-quality products in a timely manner.
It is based on the principles of transparency, inspection, and adaptation, and provides a flexible and collaborative approach to project management.

2.5 RAD (Iterative /Incremental Model):



1. Rapid Application Development (RAD) is formally a parallel development of functions and subsequent integration.

2. It is a type of incremental model.

3. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype.

4. There are some advantages of RAD model like... Product can develop soon, can take initial reviews quickly, Integration from very beginning solves a lot of integration issues.

5. Some disadvantages like it required strong team and good individual performance to understand business requirements. Only system that can be modularized can be built using RAD.

6. Rad model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

2.6 Spiral (Iterative /Incremental Model)

1. The spiral model is a risk-driven process model generator for software projects.

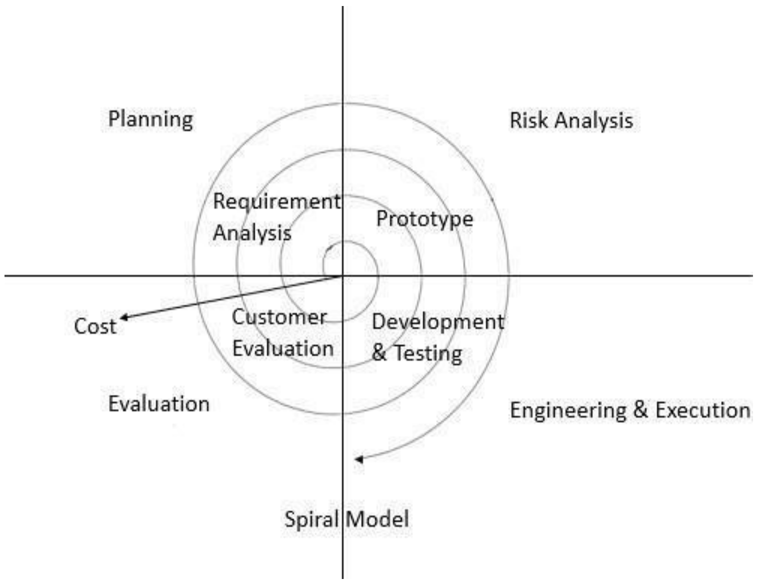
2. This model is best used for large projects which involves continuous enhancements.

3. There are specific activities which are done in one iteration (spiral) where the output is a small prototype of the large software.

4. The same activities are then repeated for all the spirals till the entire software is build.

5. Some advantages of Spiral model is It is use for Larger projects, More and more features are added in a systematic way.

6. Some Disadvantage is that It is costly for smaller projects, is not beneficial for smaller projects, and time consuming also.



2.6 Prototype Model)

Software prototype is an early-stage deliverables that are built to showcase how requirements must be implemented.

1. Prototypes is a representation of requirements.
2. Prototypes help to clarify complicated areas of products in development.
3. Represent how the solution will work and how users will interact with it to accomplish their tasks.
4. The latter can even become the early versions of the product that already have some pieces of the final code.

Module 3: Verification & Validation

1.1 Verification:

1. Verification is the process of evaluating work products (Documents) of s/w development phases to determine whether they meet the specified requirements.
2. Verification Also called as Static Testing.
3. Verification ensures that the product is built according to their requirements and design specifications.
4. Verification relies on the manual examination(reviews) of Work products (Documents).
5. Are we building the product, right?
6. Defects found costs less to fix in static testing as compare to Dynamic Testing.
7. Verification can happen on Any software work product including requirements specifications, design specifications, code, test plans, test specifications, test cases, test scripts, user guides etc.
8. Benefits of verification include early defect detection and it avoids defect multiplication.
9. Compared to dynamic testing, Verification finds causes of failures (defects) rather than the failures themselves.
10. Done by QA (Quality Assurance) Team.

3.2 Verification Testing Techniques:

1. Review
2. Walk-through
3. Inspection

3.2.1 Review: - Review is conducted on documents to ensure that document is correct and complete or not.

- To check if the content written is correct.
- Review can be performed by anyone having knowledge about the document.

- **Different type of review is there:**

Requirements review
Design review
Code review
Test plan review
Test Case review

3.2.2 Walkthrough: -

1. Lead by author.
2. It is a step-by-step re-presentation by the author of a document in order to gather information and to establish a common understanding of its content.
3. Done between colleagues.
4. It may be informal or semiformal process.
5. Main purposes: learning, gaining understanding, finding defects.

3.2.3 Inspection: -

- It is formal process.
- Led by trained moderator (not the author).
- Done at Each phase to have opinion, is phase completed & can we move to the next phase?
- Main purposes: Finding defects.

5 Major Roles in Inspection Meeting

1. **Moderator:** Activity Leader.
2. **Author**: Who's document under inspection.
3. **Reader:** Reads documents for all.
4. **Recorder:** Notes Minutes of meeting.
5. **Inspector**: Domain Expert.

3.3 Validation:

1. Validation is the process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.

2. Validation, which requires the execution of software.
3. Also Called as Dynamic Testing.
4. It requires Execution of Software.
5. Ensures Are we building the right product?
6. Dynamic Testing(validation) finds just failures not causes of failures.
7. Checks the Product, not the process.
8. Done by QC (Quality Control) team.
9. Advantages of Validation if During
10. If in verification some defects are missed then during validation process it can be caught as Defect. Defect Fixing cost in validation is more as Compare to Verification.
11. Verification and Validation have the same objective of identifying defects.

3.4 Validation Testing Technique/ Levels of Testing

1. Unit Testing
2. Integration Testing
   2.1. Unit Integration testing
   2.2. System Integration testing
3. System Testing
4. User Acceptance Testing
   4.1 Alpha Testing
   4.2 Beta Testing

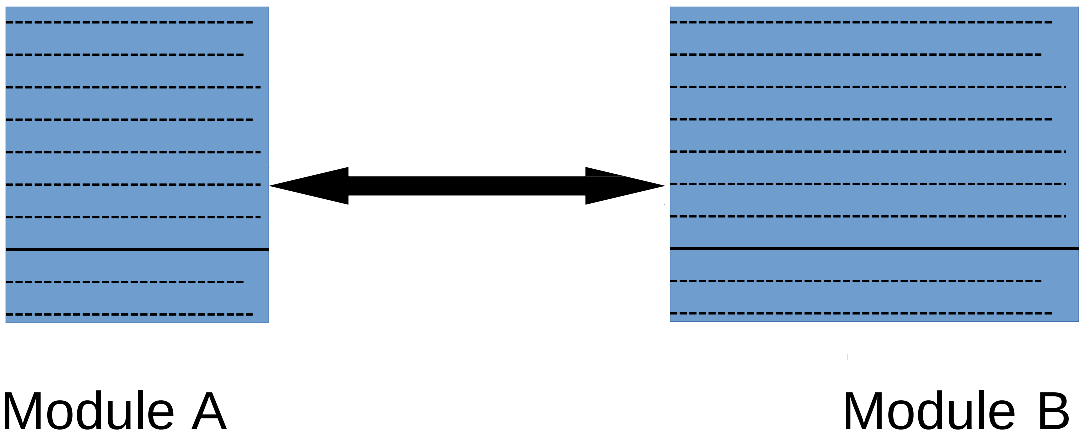
3.4.1 Unit /Component Testing:

- Component testing is also known as unit, module or program testing.
- Testing small piece of executable code.
- The unit testing purpose is to ensure that the unit satisfies its functional specification.
- If Unit is working properly further testing (i.e., Integration Testing will be simplified.
- Typically done by Programmer since it requires access to code.
- Defects found & fix during Unit testing typically not recorded (Reported).

3.4.2 Integration Testing:

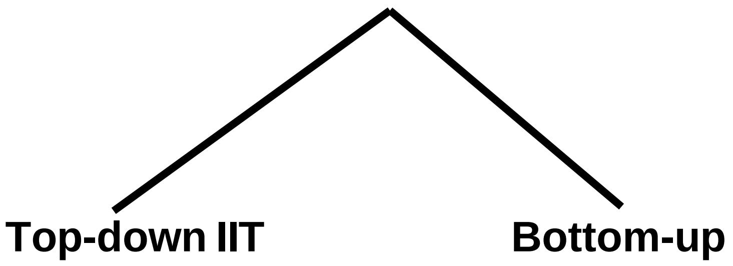
- The purpose of integration testing is to expose defects in the interfaces and in the interactions between integrated components or systems.
- There may be more than one level of integration testing as follows.
- Component integration testing tests the interactions between software component and is done after Component testing...

- At each stage of integration, testers concentrate solely on the integration itself.
- For example, if they are integrating module A with module B, they are interested in testing the communication between the modules, not the functionality of the individual module as that was done during component testing. Both functional and structural approaches may be used.



- System integration testing tests the interactions between different systems or between hardware and software and may be done after system testing.

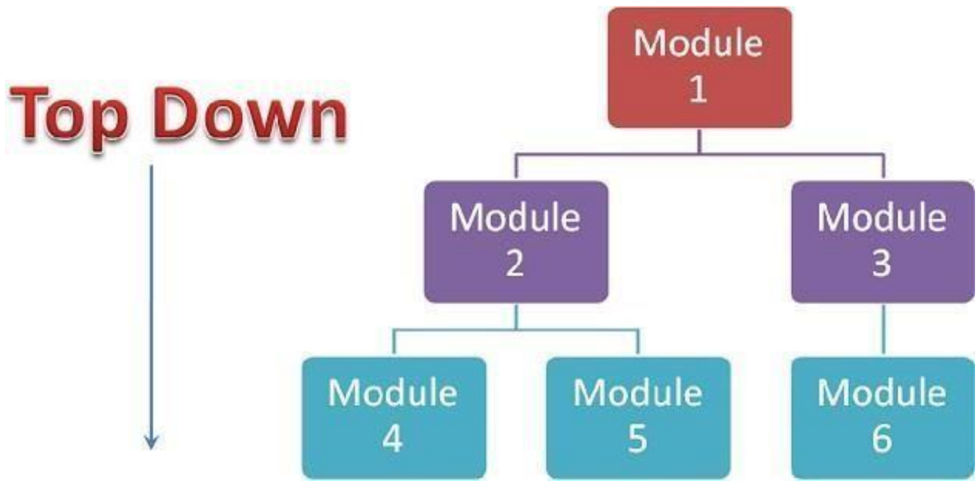
Integration Testing Approaches



IIT Top – Down Approach:

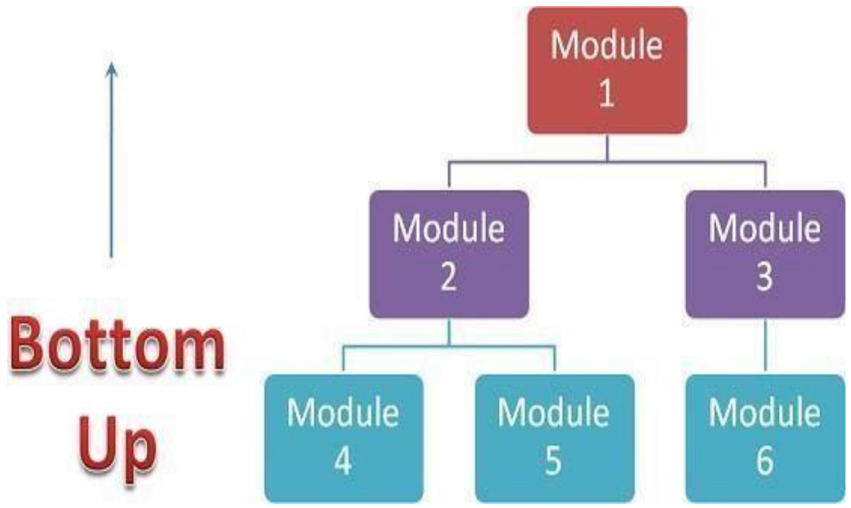
1. This technique starts from the top most module and gradually progress towards the lower modules. Only the top module is unit tested in isolation.
2. After this, the lower modules are integrated one by one. The process repeated until all the modules are integrated and tested.
3. we would need some program or a “simulator”. These simulator programs are called **“STUBS”.**

4. “Stubs” can be referred to as code a snippet which accepts the inputs/ requests from the top module and returns the results/ response.

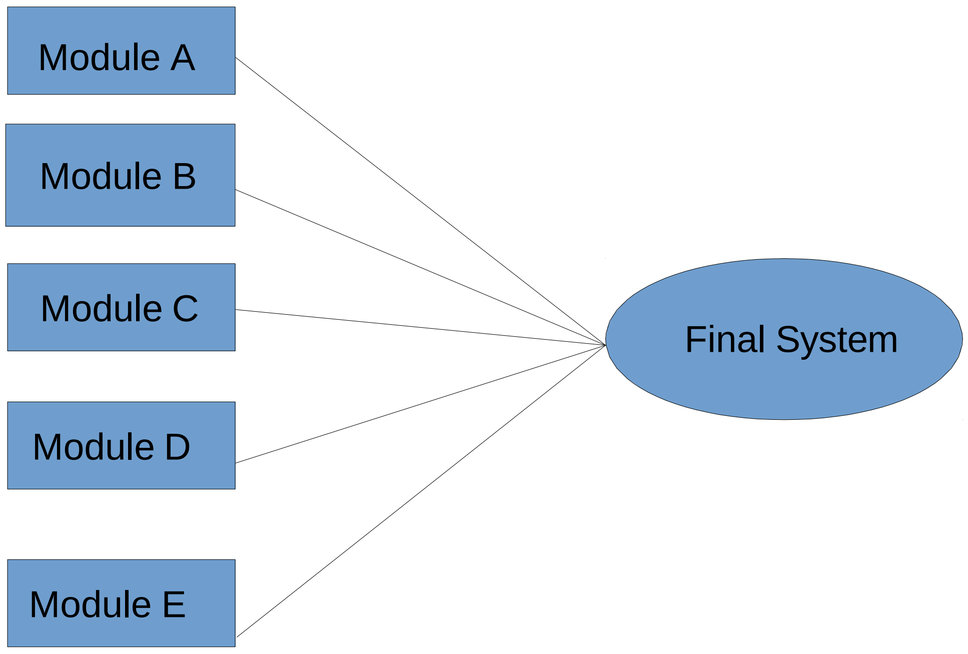


Bottom-Up Approach:

1. Bottom-up testing, as the name suggests starts from the lowest or the innermost unit of the application, and gradually moves up. Integration testing starts from the lowest module and gradually progresses towards the upper modules of the application.
2. This integration continues till all the modules are integrated and the entire application is tested as a single unit.
3. We would need some program or a “simulator”. These simulator programs are called “**DRIVERS”.**
4. In simple words, “**DRIVERS”** are the dummy programs which are used to call the functions of the lowest module in case when the calling function does not exist.



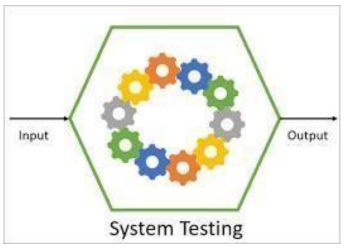
Big-Bang Approach:



- Big-Bang approach is used when all the component i.e. (hardware, software) are combined.
- In **Big Bang** all components or module are integrated simultaneously, after which everything is tested as a whole. In this approach individual modules are not integrated until and unless all the modules are ready.

3.4.3 System Testing:

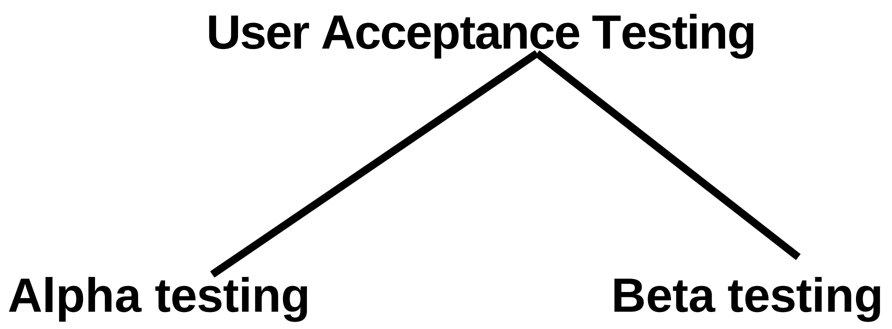
- System testing is concerned with the behavior of a whole system/products defined by the scope of a development project or product.
- System testing is most often the final test to verify that the system to be delivered meets the customer need or requirement.
- Its purpose is to find as many defects as possible.
- System testing should investigate both functional and non-functional requirements of the system.



- In system testing, the test environment should correspond to the final target or production environment as much as possible in order to minimize the risk of environment- specific failures not
- being found in testing. Most often it is carried out by independent test team.

3.4.4 User Acceptance Testing:

1. Acceptance testing is most often the responsibility of the user or customer.
2. User acceptance is a type of testing performed by the Client to certify the system with respect to the requirements that was agreed upon.
3. The acceptance test should answer questions such as: 'Can the system be released?'
4. The main purpose of this testing is to validate the end-to-end business flow & not to find defects.
5. This testing happens in the final phase of testing before rolling out the software application to Market or Production environment.



Alpha testing:

- A type of UAT in which customer or client do testing in production environment.
- Developers and Testers monitor Alpha Testing.
- It is always performed within the organization, and not open for market or public.
- Alpha Testing is done before the launch of software product into the market.

Beta testing:

- A type of UAT in which customer or client do testing in live environment.
- It is performed at the user’s premises in the absence of the

development team.
- It is always performed by users.
- Beta Testing is always performed at the time when software product and project are marketed.
- Beta Testing is also known by the name Field Testing.

<u>3.5 Testing within a Life Cycle Model (V-Model):</u>

1. For every development activity there is a corresponding testing activity.
2. The analysis and design of tests for a given test level should begin during the corresponding development activity.
3. V-model is also known as <u>*verification & validation*</u> model.
4. When development documents (BRS, SRS, HLD, LLD) are prepared, finalized and reviewed in Verification process simultaneously test documents (test cases) are created (unit, integration, system, acceptance) by referring development documents in validation process.
5. V model is a sequential model so it starts from BRS to coding that comes in verification process and after that unit to UAT is followed by executing test cases that comes under validation process. i.e.
6. According to BRS test cases for acceptance testing created and executed.
7. According to SRS test cases for system testing created and executed.
8. According to HLD test cases for integration testing created and executed.
9. According to LLD test cases for unit testing created and executed.

<u>3.6 V model advantages and disadvantages</u>:

V- model advantages:

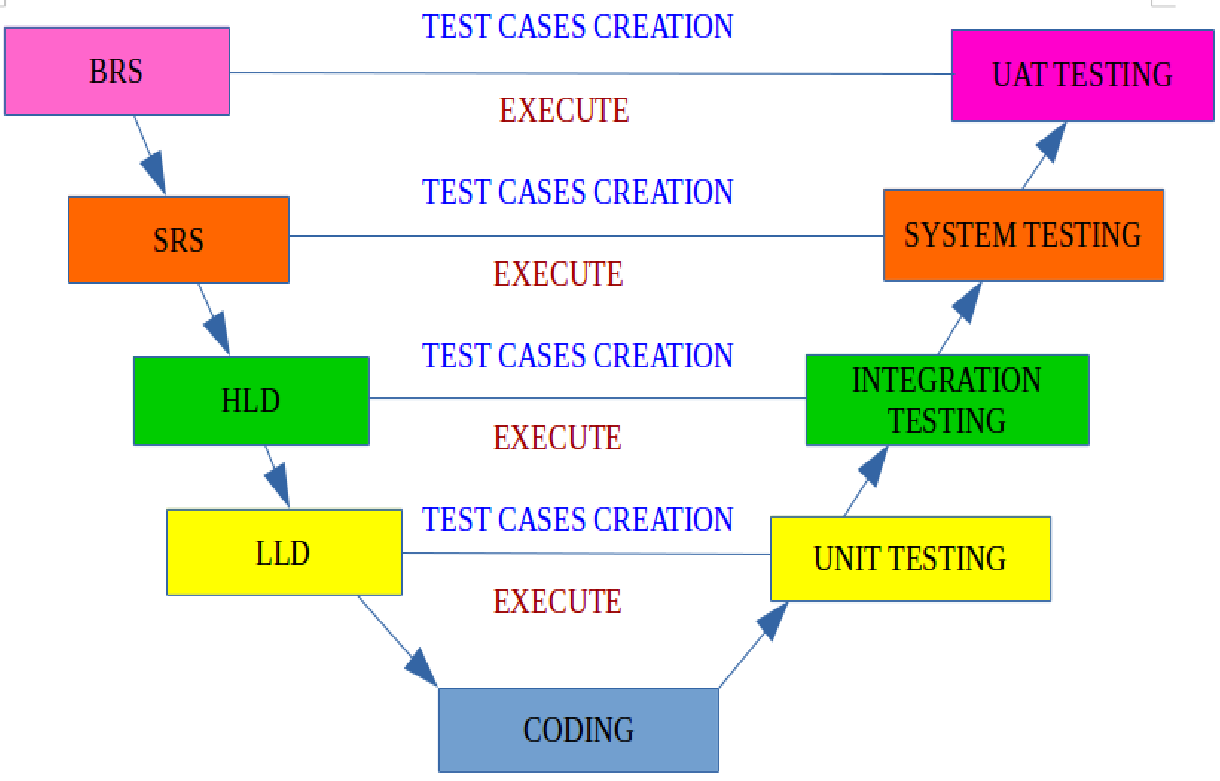
- It is simple and easy to use.
- It has major changes to find defects in the early stage.
- In this phase, test planning and test designing like test activities perform before the coding.
- It suited for small projects where requirements not complex.
- It is a high discipline model to develop software.

V- model disadvantages:

- The risk is high and unpredictable.
- This model does not suite for complex projects.
- It is difficult to go back to the previous stage after completing the testing phase.
- It is a poor model for long and ongoing projects.

3.7 V Model:

V model means verification and validation model. V model life cycle has sequential execution process.
It is a step-by-step procedure to develop software and overcome problems compared to the waterfall model.



Module 4: Types of Testing

4.1 Introduction:

- A test type is focused on a particular test objective, which could be any of the following.
- A function to be performed by the software.
- A non-functional quality characteristic, such as reliability or usability, the structure or architecture of the software or system.
- Change related, i.e., confirming that defects have been fixed (Retesting/Confirmation testing) and looking for unintended changes (regression testing).
- All functional and non-functional type of testing comes under system testing.

4.2 Testing Related to Change:

4.2.1 Retesting:

- After a defect is detected and fixed, the software should be re-tested to confirm that the original defect has been successfully removed.
- This is also called confirmation testing.
- It is mandatory to re-execute the failed test case with same input data other data may be optional.

4.2.2 Regression Testing:

- Regression testing is the repeated testing of an already tested program, after modification, to discover any new defects introduced as a result of the change(s).
- **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.
- It is performed after any kind of changes are made like when requirements change, environments changed or legislation changed.

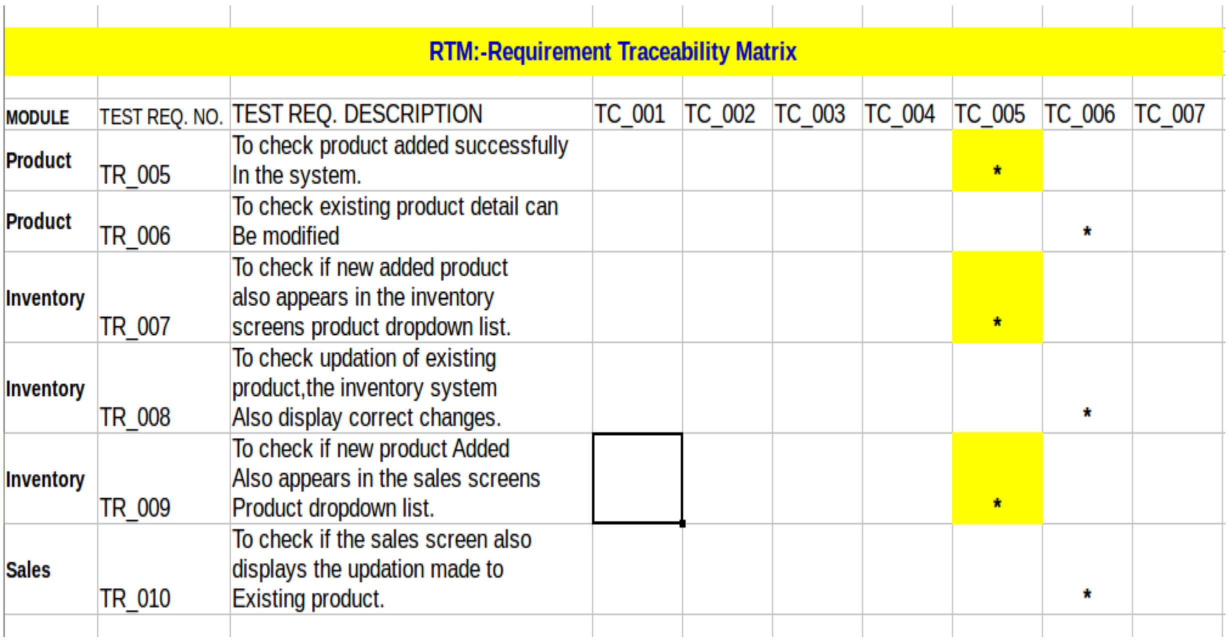
- Impact Analysis is done with the help of RTM (Requirement Traceability Matrix) to decide how much Regression should be done.
- Automation Testing Tools should be used for Regression Testing since it is repetitive in nature.

Requirement Traceability Matrix

In simple words, a testing requirements traceability matrix is a document that traces and maps user requirements. The purpose of this document is to make sure that all requirements are covered in test cases so that nothing is missed

RTM with impact analysis

S.N	**Requirement ID**	**Test Cases**	**Requirement version**	**Defects**
1	REQ_12	TC 01, TC 02, TC 03	1.0	D_21



RTM:-Requirement Traceability Matrix

MODULE	TEST REQ. NO.	TEST REQ. DESCRIPTION	TC_001	TC_002	TC_003	TC_004	TC_005	TC_006	TC_007
Product	TR_005	To check product added successfully In the system.					*		
Product	TR_006	To check existing product detail can Be modified						*	
Inventory	TR_007	To check if new added product also appears in the inventory screens product dropdown list.					*		
Inventory	TR_008	To check updation of existing product,the inventory system Also display correct changes.						*	
Inventory	TR_009	To check if new product Added Also appears in the sales screens Product dropdown list.					*		
Sales	TR_010	To check if the sales screen also displays the updation made to Existing product.						*	

4.3 Testing Related to Function:

4.3.1 Security testing:

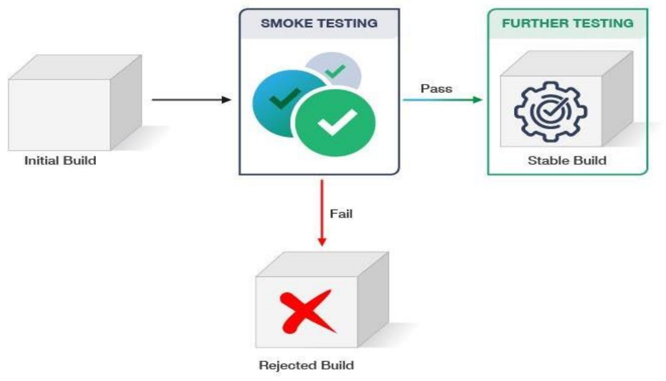
- A type of functional testing, investigates the functions (e.g. a firewall) relating to detection of threats, such as viruses, from malicious outsiders.

For Security testing tester will check.

- Access Control
- Cookies
- Encryption etc.
- Authentication / Authorization
- Password testing

4.3.2 Smoke Testing (BAT/BVT):

- Smoke testing is a testing technique that is used to check the basic functionality of a software application or system after a build is received.
- This testing is done to ensure that the build is stable enough for further testing.
- Smoke testing involves a quick and shallow check of the software application to verify that it is functioning properly and that there are no critical defects that could prevent further testing.
- Smoke testing is typically performed by testers or developers before any detailed testing is performed. It's a decision-making point to go for depth testing.
- It is also known as Build verification testing or Build acceptance testing.



4.3.3 Sanity testing:

Sanity testing is a testing technique that is used to check that specific functionality or components of a software application are working as expected after making changes or fixing defects.

- The main objective of sanity testing is to verify that the changes made to the application have not introduced new defects.

- The functionality of sanity testing is to verify that the critical functionality of a software build is working Correctly.

4.4 <u>Testing Related to Software</u> Characteristics (Non- Functional Testing):

- Non-functional testing includes performance testing, load testing, stress testing, usability testing, maintainability testing, reliability testing and portability testing .it is the testing of "how" the system works.

<u>Types of Non-Functional Testing</u>

- **Performance Testing**
- **Compatibility Testing**
- **Configuration Testing**
- **Usability Testing**
- **User Interface Testing**
- **Internationalization Testing**
- **Localization Testing**

4.4.1 <u>Performance Testing:</u>

- **Performance Testing** is a type of software testing that ensures software applications to perform properly under various loads and conditions, and ensure that the system can handle the expected number of users or transactions.

- While doing performance testing on the application, we will concentrate on the various factors like **Response time, Load, and Stability** of the application

Types of performance testing:

- Load Testing
- Stress Testing
- Spike Testing

- Endurance Testing

Load Testing:

- Load testing is a kind of performance testing which determine a system's performance under real-life load conditions.
- This testing helps determine how the application behaves when multiple users access it simultaneously.

Why load testing:

- Load testing gives confidence in the system & its reliability and performance.
- Load Testing helps identify the bottlenecks in the system under heavy user stress scenarios before they happen in a production environment.

Stress Testing:

- It validates an application's behavior when it is pushed beyond normal or peak load conditions.
- The goal of stress testing is to reveal application bugs that surface only under high load conditions. These bugs can include such things as memory leaks.

Why Stress Testing:

- 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.
- To find the crash point.

Spike Testing:

- It is a subset of Stress testing, tests the software's reaction to sudden large spikes in the load generated by users (Load increments and decrements).

Endurance Testing:

- It is a subset of Load testing, is done to make sure the software can handle the expected load over a long period of time.

Performance Testing Tool:

- Since Performance Testing requires lots of user's which is impossible to have physically. Hence tools are used in performance Testing.

Like:

- JMeter
- LoadRunner
- RPT (Rational Performance Tester)

4.4.2 Compatibility Testing:

- Compatibility Testing is performed to check that the application functions properly across various hardware and Software environment.
- More Important in web applications since user can use any browser or OS for accessing.
- This testing is necessary to check whether the application is compatible with the client's environment.

4.4.3 Configuration Testing:

- Configuration Testing is also known as Hardware compatibility testing and is performed to check that the application functions properly across various hardware.

4.4.4 Usability Testing:

- Usability means how easy the application is to use/learn.
- Ease of use should be balance between fresh user and Experience User.
- It should be done by user of Product.

4.4.6 User interface Testing:

- User interface means how user Friendly the application is for the customer.
- Deals with Graphical User Interface.
- Spelling Check.
- Grammatical Check.
- Font and Size.
- Color etc.

4.4.7 Internationalization Testing:

- Internationalization means to test the application with respect to various languages and regions without any changes.

4.4.8 Localization Testing:

- Localization means to test that the internationalized software for specific language or region by adding local specific components, Key words and translating the text.

Important types of testing used for web-based application:

1. Form testing

- Check labels, field navigation, data transmission meaningful error message.

2. Link testing

- In link testing we have to check the working of all the links i.e. (External, internal, orphan, broken).

3. Client-side pop-up window

- In this we have to Check the content of pop-up, control appearance, etc.

4. Client-side scripting

- The scripting which run in browser i.e., client side is known as client-side scripting. It run whenever any activity perform at client side i.e., page load, pop- up, click, mouse over etc.

5. Reliability testing

- Testing the functionality of an application continuously over a period of time is known as reliability testing. We are doing reliability testing to check whether software is trust worthy or not. Reliability is depended on availability.

6. Availability testing

- In this we are checking software is available 24/7
  with the same performance.

7. Security testing

- The process involves testing, analyzing, and reporting every security aspect of your application

8. Compatibility testing

- Here we check does the application functions properly in different browsers and their versions, which is why it is essential to perform cross browser compatibility tests.

9. Usability testing etc.

- checks the user-friendliness of the elements in the web app. It tests the flow of the web app and how the user can navigate through it seamlessly

Module 5: Use Case & Requirement Analysis

5.1 Use Case:

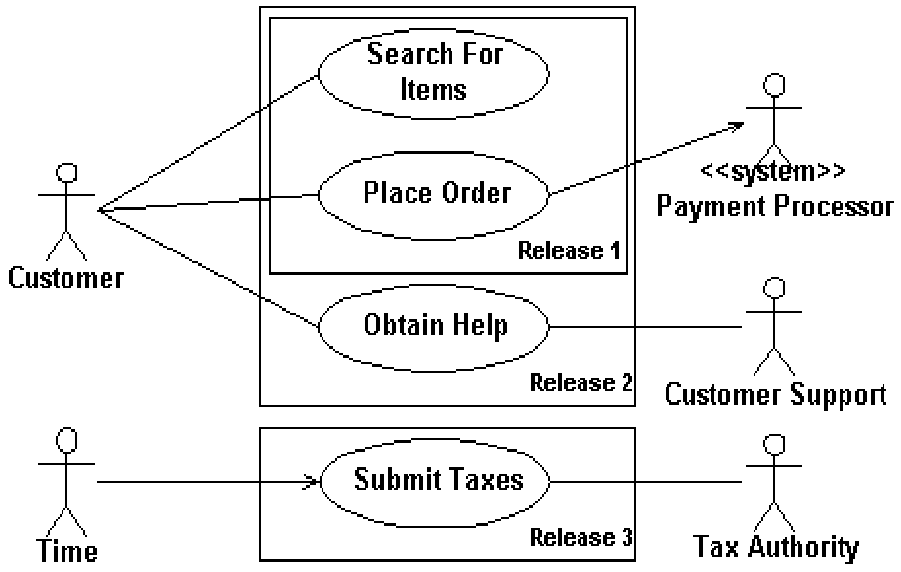
A use case is a list of actions in which a user interacts with a system.

What is Use Case:

- A Use case diagram is a behavior diagram that visualizes the observable interactions between user and the system.
- It is Document that exercise the whole system on a transaction-by-transaction basis from start to finish.
- It consists of business flow i.e., how user interact with the system (based on its most likely use).
- They use language and terms of business rather than technical terms.
- Written only for Functional requirements.

Remember:

- Use cases are requirements but are not all of the requirements.
- Use Cases are Not Good for defining User interfaces, Data formats and Non-functional requirements.
- Use cases are diagrammatically representations of flow of system.



Common templates for use cases are.

1. Use case Id
2. Goal
3. Pre-condition
4. Actors
5. Steps
6. Use case flow/path
7. Post-condition

Use case Id: These attributes consist Unique Id for uniquely identifying the use case template.

Use case Goal: Goal is what user want to achieve from the system.

Use Case Pre-Condition: Environmental and state conditions that must be fulfilled before Use Case execution.

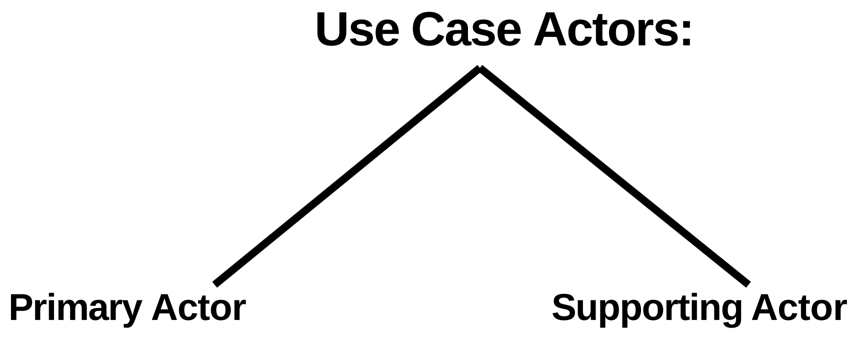
- Each use case must specify any preconditions that need to be met for the use case to work.

Example:

To identify the square root of a number, the precondition is that the number should be greater than zero. Upon executing the pre-condition, the square root of the number is displayed on the console.

Use Case Actors:

- Actors are entities which interact with a system.
- An actor might be a person, a company or organization, a computer program, or a computer system — hardware, software, or both. They are not part of the system and are situated outside of the system boundary.
- Actor has responsibility toward the system (inputs), and Actor have expectations from the system (outputs).
- A use case defines the interactions between external actors and the system under consideration to accomplish a goal.



Primary Actor:

The use case is initiated by the primary actor. Primary Actor Tries to achieve goal.

Supporting Actor:

Helps to Primary Actor to achieve its Goal.

Use Case Path/Flows:

- Normal/ Basic
- Alternate
- Error/Exception.

Normal/ Basic: The main scenario described in the use case which a user will most likely take to achieve the goal.

Alternate Flow: An Alternate Flow is a step, or a sequence of steps that achieves the use case’s goal following different steps

Exception Flow: An Exception is anything that leads to NOT achieving the use case’s goal.

Post-Condition of Use Case: Use cases must also specify post conditions that are observable results and a description of the final state of the system after the use case has been executed successfully.

Advantage of Use Case:

- Since it is written in language of user. They serve as the foundation for developing test cases mostly at the system and acceptance testing levels.
- use cases particularly good for finding defects in the real-world use of the system (i.e., the defects that the users are most likely to come across when first using the system).
- use case consists mainly of narrative text, it is easily understandable by all stakeholders, including customers, users and executives, not just developers but also testers.

Note: - For use case template refer class notes.

Requirement Analysis

5.2.1 What is Requirement Analysis:

It is the first phase of STLC. It involves the testing team understanding and analyzing all requirements specified by stakeholders and end users. Testing professionals determine whether the requirements are testable or not. If any requirements are ambiguous, missing, or not testable, they communicate the same to stakeholders.

- ➢ Requirements analysis is a process used to determine the needs and expectations of a new product.
- ➢ It involves frequent communication with the stakeholders and end-users of the product to define expectations, resolve conflicts, and document all the key requirements..
- ➢ In this step Testing team understands the requirement to know what to test & figure out the testable requirements.
- ➢ In this Requirements are analyzed in terms of below factors.

a. Are requirements complete, and Correct?

b. Are requirements achievable and Testable?

- ➢ If any conflict, missing or not understood any requirement, then QA team takes follow up with the various stakeholders.

Two types of Requirements

Functional Requirement **Non-Functional Requirement**

Functional Requirement:

- A Requirement that specifies a function that a component or System must perform (What the System should do?).
- A Functional Requirement is a description of the service that the software must offer
- Functional requirement is the 100% need of the client. Example:
  - Login to the System.
  - Search option given to user to search.

Non-Functional Requirement:

- A Requirement that is does not related to functionality to functionality, but to attribute such as Performance Usability, Reliability, Portability, UI etc...
- It is the expectation of the client.
- How system should do.

Example:

- Page of the system should be open within 5 seconds.
- Size of the button, color, font etc.

Note: -For more example refer class notes

Categories of requirement

Explicit Requirement **Implicit Requirement**

Explicit Requirement:

- The requirement which are mention by the client explicitly

is called as “**Explicit requirement**”.
Example: Client wants DOB field in the registration form.

Implicit Requirement:

- The requirement which are not mention by the client which are derived from the explicit requirement is called as “**Implicit Requirement**”.
- Implicit requirement is in the hand of development company how they implement. Example: DOB field can be implemented either by (drop down, manually entering, search etc.).

The primary goal of Requirement Analysis phase is that the testing team should clearly understand requirements and identify potential risks in the testing process.

Module 6: Test Planning and Strategy

6.1 Introduction:

- Test Plan is a document that includes details of scope, Objective and method of testing such as resources and schedule of intended test activities.
- 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.
- Test Strategy is the document that tell us what type of technique to follow to test the different modules of software. It is High level document prepared by the project manager.

Importance of Test Plan:

- It helps to determine the needed efforts.
- Avoids Random Testing.
- Proper Utilization of resources.
- The test planning process and the plan itself serve as the means of communication with other members of the project team, testers, managers and other stakeholders (clients & executives).

Guidelines for test plan:

- Be specific, for example when you specify an operating system as a property of a test environment, mention the OS Edition/Version as well, Make use of lists and tables wherever possible. Avoid lengthy paragraphs.
- Test plan must review a number of times prior to baseline it or sending it for approval.

  Update the test plan as and when necessary.

6.2 Test Plan Attributes/ Template IEEE 829:

1. Test plan ID (Name) Identifier.
2. Introduction
3. References
4. Test Items (Inclusions)
5. Features to be tested
6. Features not to be tested
7. Test Environment/Test bed
8. Test deliverables
9. Approach
10. Suspend & resume criteria
11. Entry & Exit Criteria
12. Risk & Contingencies
13. Roles and Responsibilities
14. Schedule
15. Staffing and Training needs
16. Approvals

6.2.1 Test Plan Identifier:

- Unique ID given to plan so that we can identify different project and plan.

6.2.2 Introduction:

- Detail description about the plan if it’s a Master plan (overall activity of STLC is defined) or Generic Plan (for particular test level, test types), and introduction about project.

6.2.3 References:

- Different documents and version that we have referred to prepared the plan like...
- BRS
- SRS
- Development plan
- previous versions documents.

6.2.4 Test Items:

- All the items (Module names) of the current version of the software.

6.2.5 Features to be tested:

- All the Functionalities that have to be tested.
- This is a listing of what is to be tested from the Users viewpoint of what the system does. This is not a technical description of the software, but a user's view of the functions.

6.2.6 Features not to be tested:

- All the functionalities that may not be released in the current version. If there is less time than the low priority test cases may not be executed.

6.2.7 Test Environment/ Test Bed:

- It describes Require Hardware and software for setting-up Test Environment or Test Lab.
- Describe the environment in which the application is going to be accessed.

6.2.8 Test Deliverable:

- All the documents that will be generated as a result of testing activities (Test plan, Test cases, Execution log).
-

6.2.8 Approach/Strategy:

- This is your overall test strategy for this test plan; it should be appropriate to the level of the plan (Master, Generic etc.).
- Are any special tools to be used and what are they?
- Will the tool require special training?
- How is Configuration Management to be handled? Hardware, Software.

6.2.9 Suspend/Resume criteria (Pause Criteria):

- Defines under what circumstance we may not be able to continue further testing.
- Resume criteria defines under what circumstance we can continue the testing activates.

6.2.10 Entry & Exit Criteria:

- Entry Criteria describes when to start Testing.
- Exit Criteria describes when to stop testing.

6.2.11 Risk & Contingencies:

- Identify the high-risk assumptions of the test plan related testing process.
- Specify contingency plans for each.

6.2.12 Roles & Responsibilities:

- Roles and Responsibilities of Team Lead or Test.
- Lead and Team members who are involved in project.

6.2.13 Schedule:

- Schedule for all Test activities in this Software Test Process.

6.2.14 Staffing and Training:

- Hiring new Employee as per Project Need.
- Training on the application/system (Domain Training).
- Training for any test tools to be used.

6.2.15 Approvals:

- Specify the names and titles of all persons who must approve the plan.
- Provide space for signatures and dates.

Module 7: Test Design & Techniques

7.1 Introduction:

The objective of test design technique is to identify the test conditions and test scenarios so that effective test cases can be written. Test design technique is best approach then writing the test cases randomly.

Test Case: A test case is a set of actions to perform that validates a specific aspect of an application functionality.

Test Case Development Process:

- During test analysis, the test basis (SRS or BRS) documentation is analyzed in order to determine what to test.
- During test design the test conditions, test data & RTM are created and specified.
- During test implementation the test cases are developed, implemented, prioritized.
- During Execution Test Cases are Executed & any deviation is reported as Defect.

Why to Design Test Case:

- It helps us in achieving maximum test coverage and to write test cases it will help to test the insight quality of system.

7.2 Black Box Test Design Techniques:

1. Specification base Technique:

1.1 Equivalence Class Partitioning
1.2 Boundary Value Analysis

2. Experience base Technique:

2.1 Error Guessing
2.2 Exploratory Testing

<u>Equivalence class partitioning</u>: Equivalence Class Partitioning (ECP), also known as Equivalence Class Testing, is a software testing technique used to simplify the creation of test cases by grouping inputs or test conditions into equivalence classes.

- In equivalence class partitioning, inputs to the software or system are divided into groups that are expected to exhibit similar behavior, so they are likely to be processed in the same way.
- Equivalence partitions (or classes) can be found for both valid data i.e., values that should be accepted and invalid data, i.e. values that should be rejected.
- Tests can be designed to cover all valid and invalid partitions.
- Equivalence partitioning is applicable at all levels of testing.
- Choose one data from each Partition.

Equivalence class partition Example:

- If we take data range between 15-60 Yrs. of age then. We take one value which is less than 15, one value which is greater than 60 it can be any random value and any one value which is between 15 to 60. Group the inputs/ outputs according to their behavior i.e. They are giving same outputs.

14|15 - 60 |61

Invalid | Valid | invalid

- So, there are 3 values 13, 42,62

<u>Boundary Value Analysis</u>: Boundary Value Analysis (BVA) is a software testing technique that focuses on testing values that are on the boundary or extreme limits of input ranges. It is used to identify potential errors or defects that may occur at the edges

- Behavior at the edge of each equivalence partition is more likely to be incorrect than behavior within the partition, so boundaries are an area where testing is likely to yield defects.
- The maximum and minimum values of a partition are its boundary values.
- A boundary value for a valid partition is a valid boundary value; the boundary of an invalid partition is an invalid boundary value.
- Tests can be designed to cover both valid and invalid boundary values.
- When designing test cases, a test for each boundary value is chosen.

Boundary Value Analysis Example:

- If we take data range between 15-60 Yrs. Then we need to take values which are at the boundary and value which is just above and below the boundary value. Group the inputs/ outputs according to their behavior i.e. They are giving same outputs.

14 | 15 – 60 | 61

invalid | Valid | invalid

- Just +1 and -1 to Edges of valid partition.
- So overall 6 values in BVA that is 14,15,16,59,60,61

Error Guessing: Error Guessing is an informal experienced based software testing technique that relies on the experience, intuition, and creativity of the tester to identify and guess potential areas of error or defects in a software application

- A commonly used experience-based technique is error guessing.
- Generally, testers can guess defects based on experience.
- Anyone can do error guessing but it gives varying degree of effectiveness depends on testers experience or skill.

Exploratory Testing: Exploratory Testing is a dynamic and unscripted experienced based software testing approach where testers design and execute test cases while actively exploring an application. This testing method relies on the tester's creativity, intuition, and domain knowledge to discover defects, and usability issues in real-time.

- Exploratory testing is concurrent test design, test execution, test logging and learning, based on a test charter containing test objectives, and carried out within time-boxes.
- It is an approach that is most useful where there are few or inadequate specifications and severe time pressure, or in order to augment or complement other, more formal testing.

Test Scenario: A Test Scenario is a statement describing the functionality of the application to be tested. It is used for end-to-end testing of a feature and is generally derived from the use cases.

7.4 Test Scenario Template:

Test scenario is the document which consist of the test condition, having following attributes.

- Test scenario id
- Reference
- Test name
- Test objective
- Pre-condition
- Test condition

7.5 Test Case Template:

Tester who writes Test Case fills following:

- Test Scenario Id
- Test case ID
- Test Objective/Test Condition
- Pre-condition
- Step Id
- Step description
- Test data
- Expected Result

Tester who Executes Test Case fills following:

- Actual result
- Status
- Remark
-

A good test case has several characteristics:

- **Clear objective:** The test case should have a clear and refined objective**.**
- **Accurate:** The test case should be accurate and specific about its purpose**.**
- **Traceable:** The test case should be traceable to requirements**.**
- **Reusable:** The test case should be reusable, meaning it can be reused to perform the test again in the future.
- **Independent:** The test case should be independent from other test cases while testing one thing.
- **Simple and clear:** The test case should be simple and clear, so that any tester can understand it by reading it once.

Module 8: Incident Management

8.1 Introduction:

Defect management is a process to identify the defect of the software. Defect management will work like a backbone to developing a team in finding the defect in the early stage in a very easy way.

Defect management, records all the defect of the software these records can be seen later as well if you want to review or want to check that how you have fixed that.

What is Defect: A Software DEFECT / BUG, this is a condition in a software product which does not meet with customer requirement. A defect can occur at any stage of the software development lifecycle**.**

Types of Defects:

- Functional
- Non-functional.

Functional Defect Type:

1. Arithmetic Defects:

It includes the defects in some arithmetic expression. This type of defects is basically made by the programmer.

2. Logical Defects:

Logical defects are mistakes done regarding the implementation of the code by the developers.

3. Syntax Defects:

Syntax defect means mistake in the writing style of the code, there might be some small symbols escaped

Types of Non-Functional Defect:

1. Interface Defects:

Interface defects mean the defects in the interaction of the software and the users.

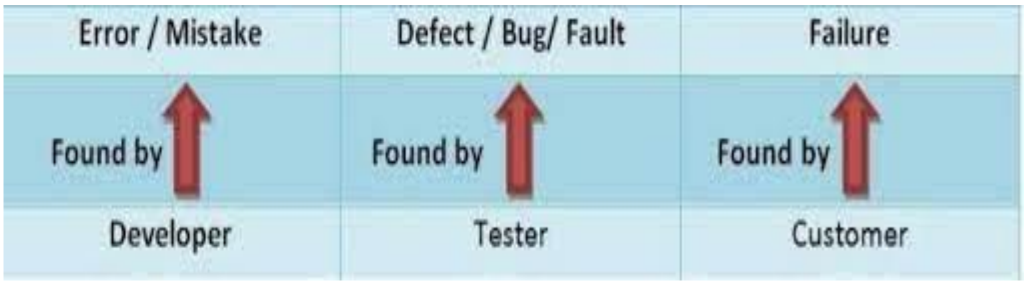
2. Performance Defects:

Performance defects may include the response of the system with the varying load on the system.

What is Incident/Defect Management:

Defect Reporting, Defect Tracking, and Status Tracking is called as Defect Management. Some companies use Manual Process (Excel workbook), and some companies use Tool based process for Defect Management.

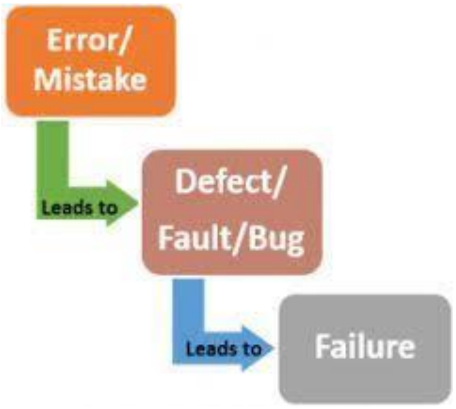
8.2 Error vs Defect vs Failure:



Error / Mistake	Defect / Bug/ Fault	Failure
Found by ↑	Found by ↑	Found by ↑
Developer	Tester	Customer

8.3 Reasons of Defect:

- Incomplete and incorrect requirement.
- Wrong Design.
- Wrong Coding.
- Wrong Testing.
- Error at the time of Defect fixing.
- Error because of Impact of Defect Fixing.



8.4 Defect Management Tools:

- Bugzilla
- Mantis BT
- QC (ALM)
- Jira

8.5 Status of Defect:

- New
- Assign/Re-assign
- Open/Re-open
- Reject
- Not A defect
- Duplicate
- Deferred
- Fixed/Resolved
- Retest
- Close

8.6 Goals of Defect Management Process (DMP)

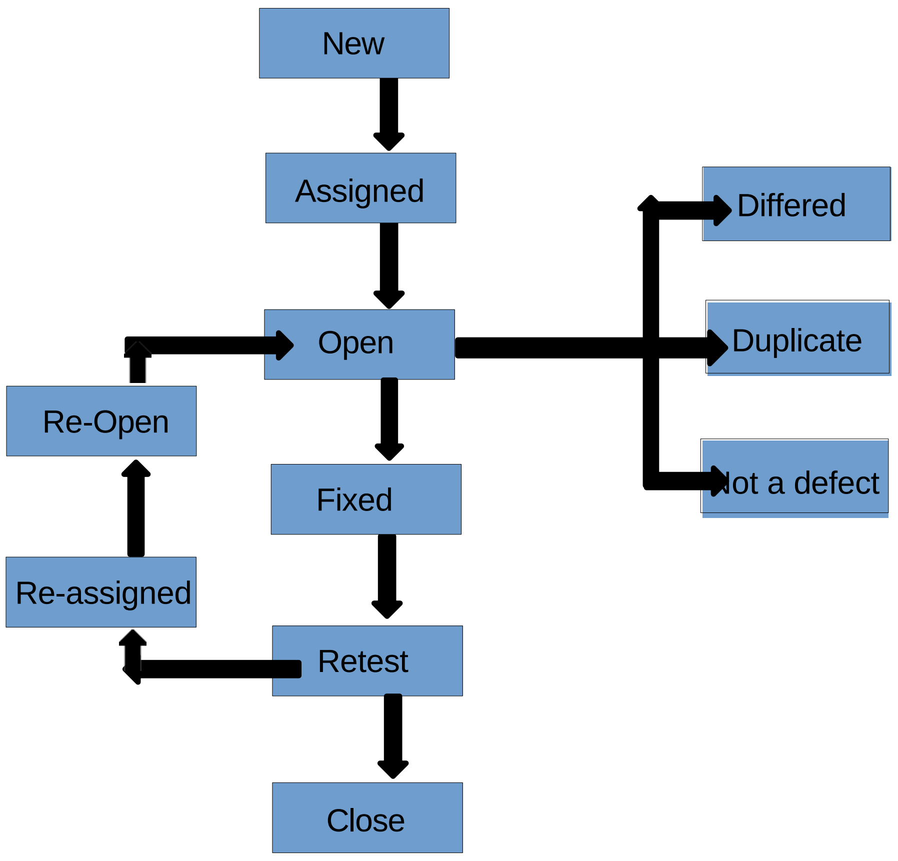
- Prevent the defect
- Detection at an early stage
- Reduce the impact or effects of defect on software
- Resolving or fixing defects
- Improving process and performance of software

Advantages of DMP:

- Identifies and fixes issues early, reducing errors in the final product.
- Logged in a tracking system with details like severity and priority.
- Evaluated for priority and resource needs during triage.
- Assigned to team members based on expertise and availability.
- Resolved through fixes, documentation updates, or other actions.
- Verified by testers to ensure correctness and no new issues.
- Closed after verification and status is updated in the system.
- Tracks, prioritizes, and reports defects accurately.
- Enhances user experience, boosting customer loyalty.
- Fosters collaboration among teams.
- Supports continuous improvement through root cause analysis.

4. Defect Life Cycle:

Defect life cycle, also known as Bug Life cycle is the journey of a defect 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



8.6 Defect Reporting Attributes/ Template:

1. Defect Id
2. Defect Reporting Date
3. Project Name
4. Project Version
5. Module/Sub Module Name
6. Test Case Id
7. Type of Defect
8. Phase in which defect is identified
9. Summary of defect
10. Step Description
11. Severity
12. Priority
13. Status

14. Reported By
15. Reported To

8.7 Levels of defect:

Severity:

- Severity is assigned to the Defect by the tester on the basis of Impact of the defect on the functionality of the Software.
- There are different Severity levels like.
    1. **High**
    2. **Medium**
    3. **Low**
- But Severity Levels changes tool to tool & company to Company.

Priority:

- Priority is assigned by the developer to the defect on the basis of how urgent that bug is to be fixed.
- Decided by Developer / BA / Client as Urgency is decided based on business needs.
- Priority Levels also changes tool to tool & Company to Company.

8.8 Categories of levels of defect:

High Severity	**High Priority**
Low Severity	**Low Priority**

Note: For example, refer class notes.

8.9 Why to Report Defect:

- To get the Defect Fix.
- To improve the Quality of Software.
- Management Reporting.
- Preventive Measures.
- Process Improvement.

Module 9: Configuration Management

9.1 Introduction:

- Configuration management is the management of features and assurance through control of changes made to hardware, software, documentation and test documentation of a system, throughout the development life cycle.
- During test planning, the configuration management

  procedures and infrastructure (tools) should be chosen, documented and implemented for the tester, configuration management helps to uniquely identify (and to reproduce) the tested item, test documents, the tests and the test harness.
- Used primarily when the software requirement get change.
- The main goal of configuration management is to manage the integrity between the work product and the end product.

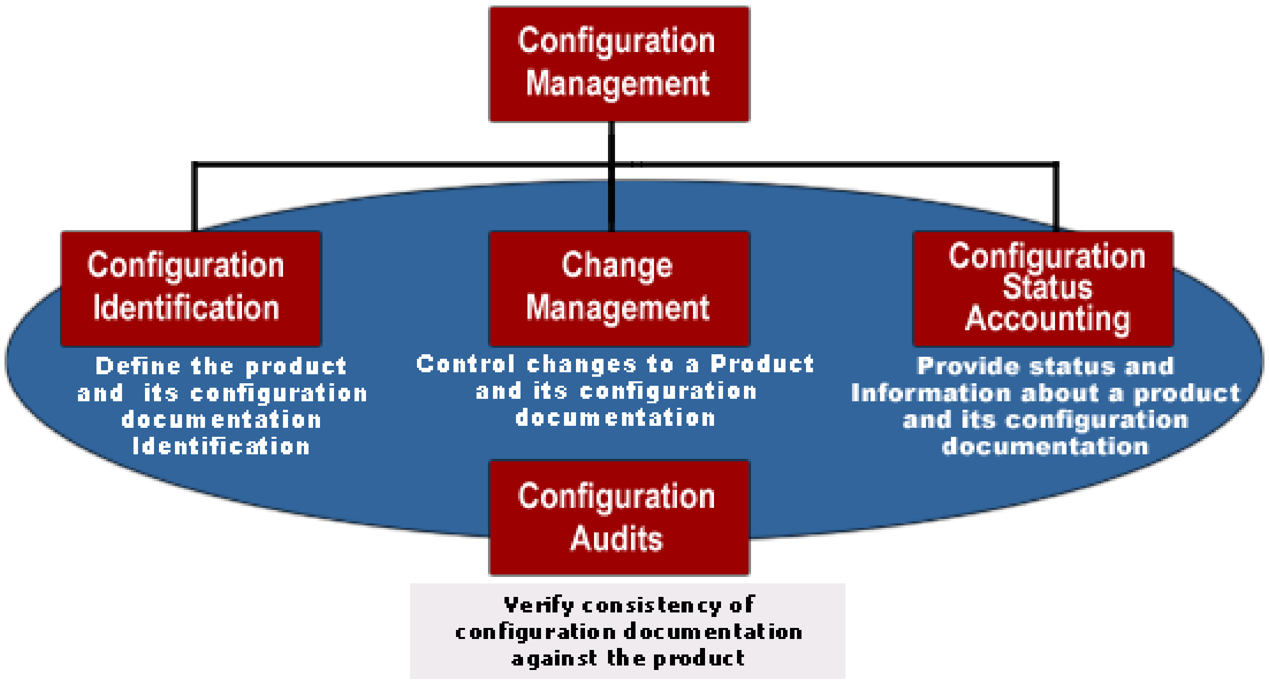
9.2 Need of configuration:

- Manage Uncontrolled changes.
- Help in traceability of correct document.
- Help in monitoring.
- Manage the Simultaneous update.
- Version management.

9.3 Configuration Management Includes:

- Change management- Manage changes to the specifications of a potential software system.
- Documentation management-Manage all documentation including defects, specification, testing, purchasing, emails, agendas every single documentation detail.
- Hardware/firmware configuration management manage all the hardware and firmware.
- Source code management –Manage changes to the source code including the application code, operating system, compilers, utilities, database management system, communication system, middleware, etc.

9.4 Activities in Configuration Management:


Configuration Management
Configuration Identification
Define the product and its configuration documentation Identification
Change Management
Control changes to a Product and its configuration documentation
Configuration Status Accounting
Provide status and Information about a product and its configuration documentation
Configuration Audits
Verify consistency of configuration documentation against the product

Configuration Item identification:

- Identify which software configuration item has to be change such as:
- software requirements
- Software Design descriptions
- source code
- Project plan
- Development plan
- Software test documents etc.

Change Control:

- Controlling the changes that has been mentioned by the client this control is done by **CCB team** i.e., Evaluating and approving or disapproving changes, Scheduling those changes etc.

CCB team include people such as:

- Project manager.
- User Representative.
- Funding manager.
- Contract manager.
- Configuration manager

- Quality manager.
- System Engineers

Status Accounting:

- Recording and reporting of the status of work component and changes.
- Configuration status information is maintained in a **Configuration Item Register**.
- A list of specification that describe each configuration item.

Configuration Audit:

- Configuration audit is conducted by configuration management team or manager in this they are validating the completeness and correctness of the software- configuration management procedure is being followed.

9.5 Change control Process:

- Customer submit change request.
- BA performs Impact Analysis.
- BA prepares Recommendations.
- BA submits report to CCB team.
- CCB deliberates and decides.
- Approved changes are entered in the configuration management documents.
- Traceability matrices are updated.

9.6 Why Requirement keep on changing:

- External factor (Legislation)
- Advance in technology
- Additional capabilities requested by users

Module 10:

Manual Testing VS Automation Testing

10.1 Introduction:

- Software testing is a huge domain, but it can be broadly categorized into two areas: manual testing and automated testing.
- Both manual and automated testing offer benefits and disadvantages.
- It's worth knowing the difference, and when to use one or the other for best results.
- In manual testing (as the name suggests), test cases are executed manually (by a human, that is) without any support from tools or scripts.
- But with automated testing, test cases are executed with the assistance of tools, scripts, and software.

The type of testing (manual or automated) depends on various factors including

- Project requirements
- Budget
- Timeline
- Expertise
- Suitability

10.2 Manual Testing

Manual Testing is a type of software testing in which test cases are executed manually by a tester without using any automated tools.

10.3 Advantage & Disadvantage

Manual Testing Advantage:

- Manual testing can be used in big as well as small projects.
- Test cases can be updated easily.
- Easy to learn for the new people who entered into testing.
- Manual testing allows for human observation, which may be more useful if the goal is user-friendliness or improved customer experience.
- If you’re only testing a simple application once and don’t expect lots of updates, manual testing doesn’t require you to invest in expensive tools or software.
- Automation cannot replace human intuition and reasoning.

Manual Testing Disadvantage:

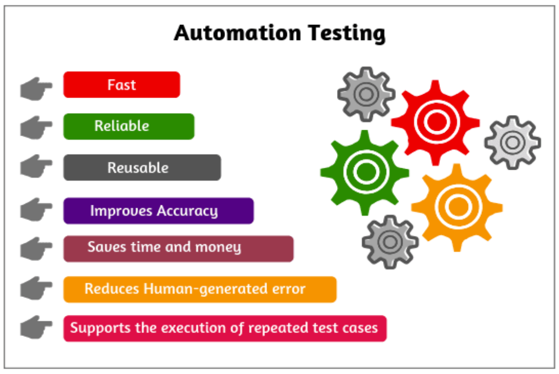
- Manual testing is not accurate at all times due to human error; hence itis less reliable.
- Manual testing is time-consuming, taking up human resources.
- Repetition of the Task is not much.
- Tiredness.
- Simultaneous actions are not possible (Parallel Testing) GUI Objects size difference & colour combination etc. is not easy to find out in manual testing.
- Load testing & performance testing is not possible in manual testing.

10.4 When to use Manual Testing:

- As Soon as Possible Testing
- Exploratory Testing
- Usability Testing
- Ad-hoc Testing

10.5 Automation Testing:

Automation testing uses some specific tools to execute the test scripts without any human interference.



10.6 Automation Testing Advantage

1. **Reliable**: Tests perform precisely the same operations each time they are run, thereby eliminating human error.
2. **Repeatable:** You can test how the software reacts under repeated execution of the same operations.
3. **Programmable:** You can program sophisticated tests that bring out hidden information from the application.
4. **Comprehensive:** You can build a suite of tests that covers every feature in your application.
5. **Reusable:** You can reuse tests on different versions of an application, even if the user interface changes.
6. **Economical:** As the number of resources for regression test are reduced.
7. **Better Quality Software:** Because you can run more tests in less time with fewer resources
8. **Fast:** Automated Tools run tests significantly faster than human users.
9. **Gives your team a break:** There's a better chance your development team will get fatigued and weary if they're manually testing an app after coding it. Automating testing will give them a chance to focus more on the big picture.

Automation Testing Disadvantage:

- Proficiency is required to write the automation test scripts.
- Debugging the test script is major issue. If any error is present in the test script, sometimes it may lead to deadly consequences.
- Test maintenance is costly in case of playback methods.
- Even though a minor change occurs in the GUI, the test script has to be re-recorded or replaced by a new test script.
- Maintenance of test data files is difficult, if the test script tests more screens, It is expensive.

  We cannot automate all areas.

Difference Between Manual Testing & Automation:

Nature of Testing:

- **Manual Testing:** Human testers perform testing by manually executing test cases.
- **Automation Testing:** Testing is automated using scripts and tools.

Reliable:

- **Manual Testing:** Manual testing is not accurate all the time due to human generated errors. Therefore, it is less reliable.
  Automation Testing: Automation testing is performed by tools or scripts. So it is more reliable.

Speed and Efficiency:

- **Manual Testing:** Slower, especially for repetitive tasks, which can lead to longer testing cycles.
- **Automation Testing:** Faster and more efficient, capable of executing tests quickly and repeatedly.

Error-Prone:

- **Manual Testing:** Manual testing can have more mistakes during test execution because humans can make errors while doing tasks like clicking buttons or checking software
- **Automation Testing:** This precision and consistency are why automation testing is less likely to have errors. It's like having a robot do a task that doesn't get tired or distracted.

Coverage:

- **Manual Testing:** Coverage depends on testers' expertise and thoroughness.
- **Automation Testing:** Can achieve higher test coverage by running a large number of test cases.

Initial Setup Time:

- **Manual Testing:** Quick to start as it doesn't require script development.
- **Automation Testing:** Requires initial time for script creation and setup.

Cost:

- **Manual Testing:** Typically, lower initial cost but can be expensive in the long run due to labor costs.
- **Automation Testing:** Higher initial investment in script development and for automation tools setup but lower ongoing costs.

Module 11 : Cloud Computing & API

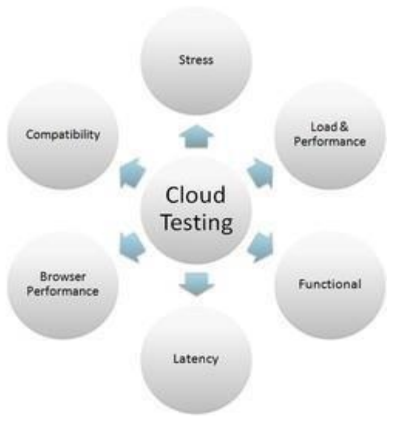
What is CLOUD?

"A cloud refers to a different IT environment. We can say it's like repository a location where we can store and manage our data".

What is cloud Computing?

Cloud Computing provides us means by which we can access the applications as utilities over the internet. It allows us to create, configure, and customize the business applications online.

- Cloud computing is a means of using the Internet and remote servers for software applications, data access, data management and storage resources.
- Cloud computing allows consumers and businesses to use applications without installing software or accessing local files on their computers and hence can be used on any computer with Internet access.



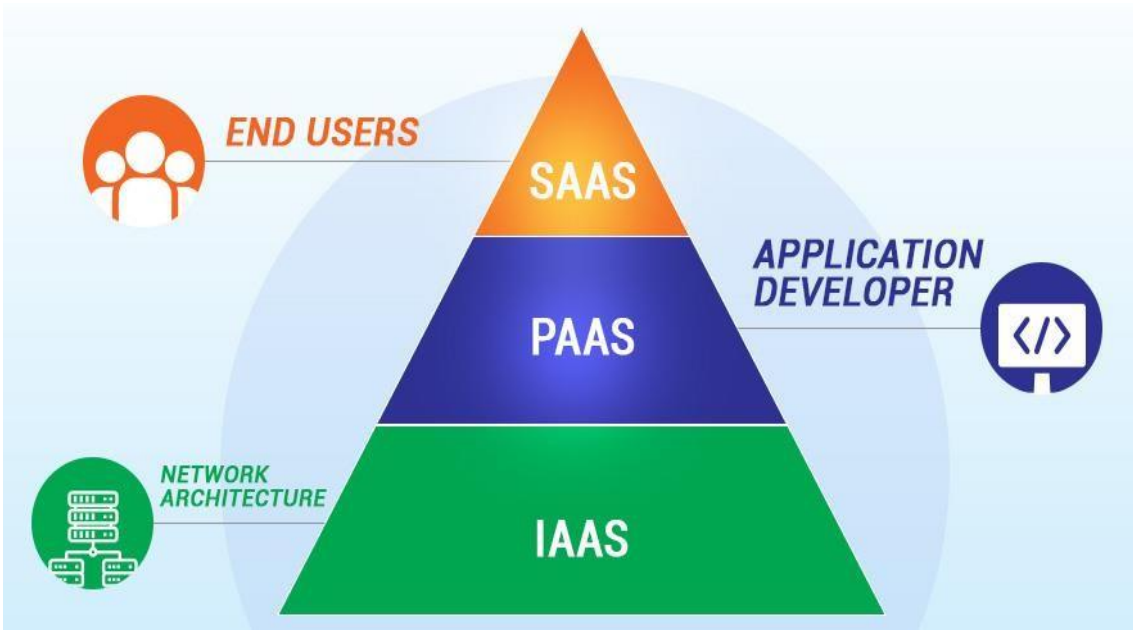
What is Cloud Testing:

- Cloud Testing is a means of testing cloud-based applications that use resources found in the cloud.
- By resources, we mean any element (hardware, software and infrastructure) necessary to carry out the tests.
- Cloud testing provides an end-to-end solution that transforms the way testing is done and can help an organization boost its competitiveness by reducing the cost of testing without negatively impacting mission-critical production applications.

Why This Cloud Technology

The small, as well as large IT companies, follow the old traditions of managing IT infrastructure, i.e., server room to keep all the details and maintaining that server. In one word it is a server room, but actually, it consists of database servers, mail server, firewalls, routers, switches, QPS (Query per second) & Load handler and other networking devices along with server engineers. To provide such IT infrastructure, a huge amount of money has to spend. So, to reduce the IT infrastructure cost, Cloud Computing technology came into play.

Types of Models in Cloud Computing



- **IaaS (Infrastructure as a Service:** provides you the computing infrastructure, physical or virtual machines and other resources like cloud-based services, pay-as-you-go for services such as storage, networking, and virtualization.

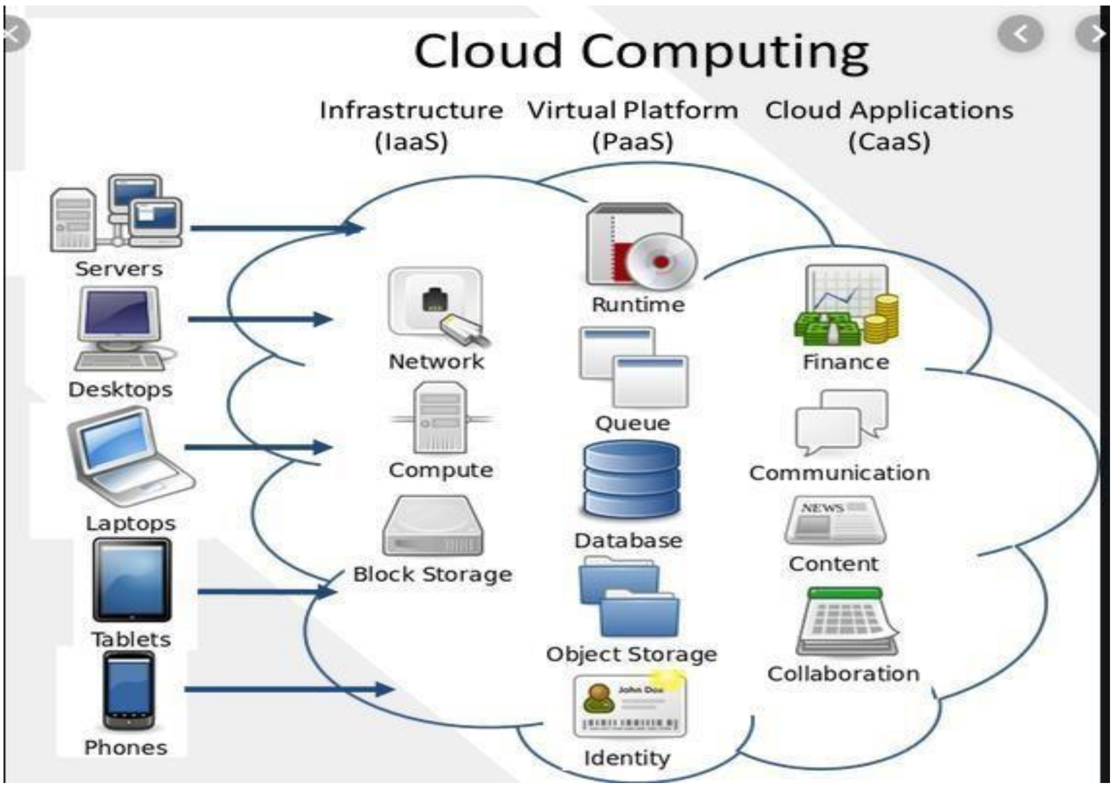
Examples: Amazon EC2, Windows Azure, Rackspace, Google Compute Engine.

- **PaaS (Platform as a Service):** provides you computing platforms which typically includes operating system, programming language execution environment, database, web server etc. (hardware and software tools available over the internet.)

Examples: AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App, Engine, Apache Stratos.

- **SaaS (Software as a Service):** model you are provided with access to application software often referred to as "on-demand software ("paid service by third party application or software provider) including the installation, setup and running of the application.

  Examples: Google Apps, Microsoft Office 365.



Cloud testing focuses on the core components like

1. **Application:** It covers testing of functions, end-to-end business workflows, data security, browser compatibility, etc.
2. **Network**: It includes testing various network bandwidths, protocols and successful transfer of data through networks.
3. **Infrastructure**: It covers disaster recovery test, backups, secure connection, and storage policies. The infrastructure needs to be validated for regulatory compliances

Other Testing types in Cloud includes

- Performance
- Availability

- Compliance
- Security
- Scalability

API

API stands for **Application Programming Interface**. An **API** is a set of programming code that enables data transmission between one software product and another. It also contains the terms of this data exchange.

It has been described as a “contract” between the client and the server, such that if the client makes a request in a specific format, it will always get a response in a specific format or initiate a defined action.
Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you’re using an API.

“In technical words, an application program interface (API) is a set of routines, protocols, and tools for building software applications”.

- In API the interface is used by not a human but by a program. Developers will write the code to talk with API.

There are different types of APIs for operating systems, applications or websites for example

Example of API:

1. Google Maps API: Google Maps APIs lets developers embed Google Maps on webpages using a JavaScript or Flash interface. The Google Maps API is designed to work on mobile devices and desktop browsers.

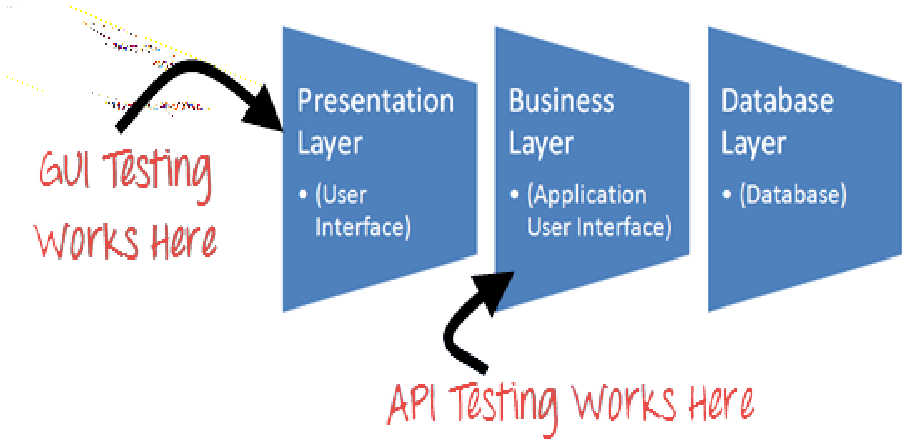
2. YouTube API: Google's APIs lets developers integrate YouTube videos and functionality into websites or applications. YouTube APIs include the YouTube Analytics API, YouTube Data API, YouTube Live Streaming API, YouTube Player APIs and others.

3. Amazon Product Advertising API: Amazon's Product Advertising API gives developers access to Amazon's product selection and discovery functionality to advertise Amazon products to monetize a website.

Why API Testing is important?

APIs are what gives value to an application. It’s what makes our phones “smart”, and its what streamlines business processes. If an API doesn’t work efficiently and effectively, it will never be adopted, regardless if it’s a free or not. Also,

If an API breaks because errors weren’t detected, there is the threat of not only breaking a single application, but an entire chain of business processes hinged to it.



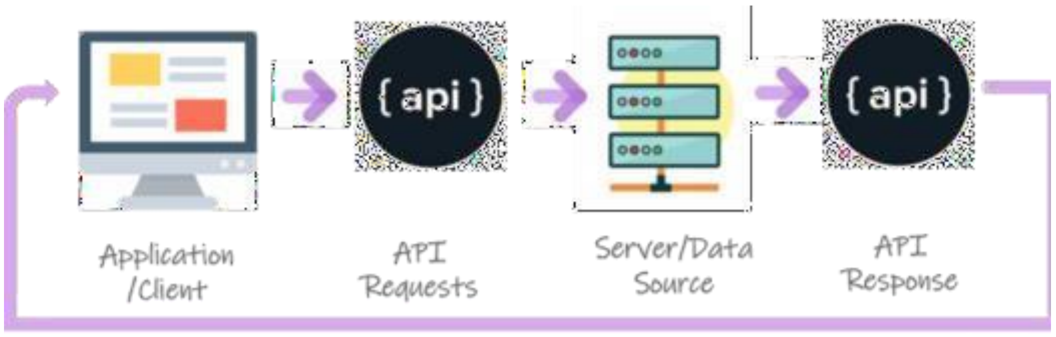
What is API testing?

API testing is a type of software testing that involves testing application programming interfaces (APIs) directly and as part of integration testing to determine if they meet expectations for functionality, reliability, performance, and security.

- **API testing** is a type of testing that involves testing application programming interfaces (APIs) directly and as part of integration testing to determine if they meet expectations for functionality, reliability, performance and security.
- **API Testing** is entirely different from **GUI Testing** and mainly concentrates on the business logic layer of the software architecture. This testing won't concentrate on the look and feel of an application. Instead of using standard user inputs(keyboard) and outputs, in API Testing you use software to send calls to the API get output and note down the system's response. API Testing requires an application to interact with API.

In order to test an API, you will need to:

- Use Testing Tool to drive the API
- Write your own code to test the API



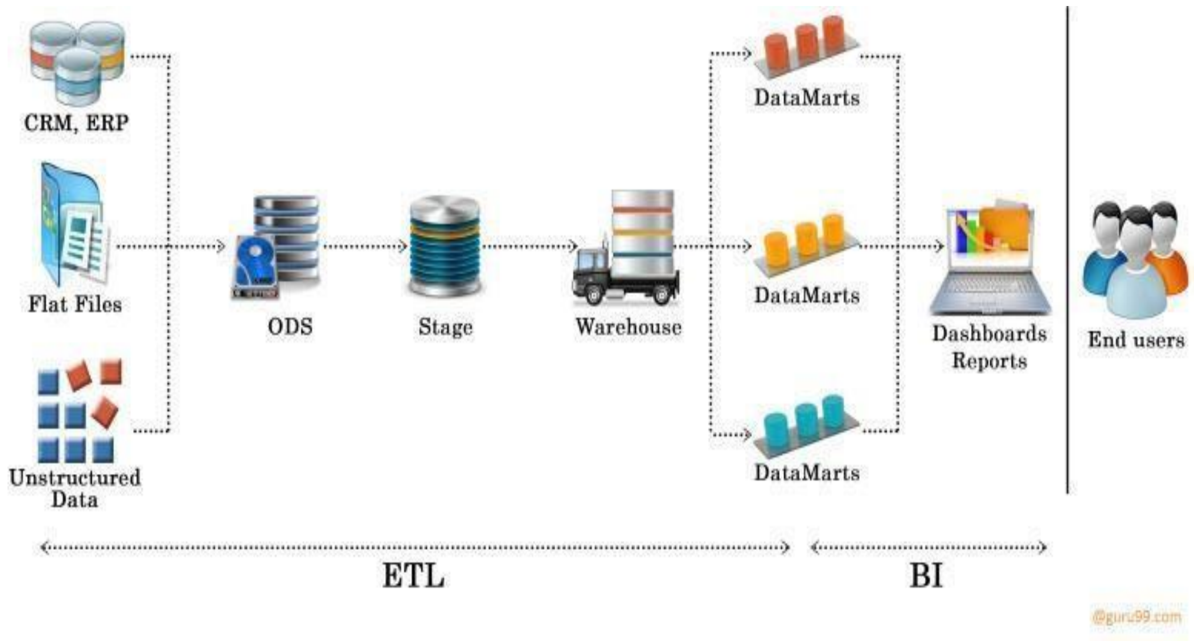
API Testing Types?

- **Load testing:** To test the functionality and performance under load
- **Runtime/Error Detection:** To monitor an application to identify problems such as exceptions and resource leaks
- **Security testing:** To ensure that the implementation of the API is secure from external threats
- **UI testing: It** is performed as part of end-to-end integration tests to make sure every aspect of the user interface functions as expected
- **Penetration testing:** To find vulnerabilities of an application from attackers

Tools used for API Testing:

1. Parasoft SOAtest,
2. TestGrid
3. ReadyAPI
4. Postman
5. SOAP UI
6. Vrest
7. Rest-Assured
8. RunScope

Module 12: ETL



Extract -

Extract relevant data

Transformation -

Transformation is the second step of ETL process where all collected data is been transformed into same format.

Loading -

Final step of ETL process, the big chunk of data which is collected from various sources and transformed then finally load to our data warehouse.

ETL (Extract/Transform/Load) - is a process that extracts data from source systems, transforms the information into a consistent data type, then loads the data into a single depository.
ETL testing refers to **the process of validating, verifying, and qualifying data while preventing duplicate records and data loss**.

The skills required for ETL testing

- PL/SQL Skills
- Manual Testing Skills
- ETL/Data Warehousing Knowledge
- Unix/Linux
- Python/shell scripting language
- ETL tools

ETL Testing Process -

- Business requirement understanding.
- Test planning and estimation.
- Design test cases and preparing test data.
- Test execution, bug reporting & closer

Data Mart -

A Data Mart is focused on a single functional area of an organization and contains a subset of data stored in a Data Warehouse. Data Mart usually draws data from only a few sources compared to a Data warehouse. Data marts are small in size and are more flexible compared to a Datawarehouse.

Data Warehouse -

As name implies Data warehouse, It is warehouse for database to store large aggregated data collected from wide range of sources within an organization. Source can be soft files, database files or some excel files.

Things we should take care for ETL Testing

- During data extraction step should be design in such a way that it should not have any negative affect on the source system.
- All data should be in general formation before uploading in warehouse
- Data extraction take time so the second step of transformation take place parallel.

Given below is the list of the top ETL Testing Tools:

1. RightData
2. Integrate.io
3. Informatica Data Validation
4. Datagaps ETL Validator
5. Data Centric Testing
6. TestBench
7. DataQ

Two terms are very important:

- **OLTP (Online transaction processing)** –It's called Database.
  its store the daily transactions of data, this is a temporary data. Here data changes frequently.
- **OLAP (Online analytical processing)** – Data warehouse process consists of OLAP process. Its in the form of organized data.

Module 13: Robotic Process Automation

What is Robotic Process Automation?

Mimicking human actions to perform a sequence of steps that lead to a meaningful activity, without any human intervention is known as **Robotic Process Automation**.

You could just configure a computer software or a robot to interpret the human actions and imitate them.

So, you could just configure a robot to publish the articles every day at the mentioned time.

That would not only cost you less but would also be less tiresome.

Automation v/s RPA:

Automation existed even before RPA came into the picture.
Though multiple overlaps exist between these two, unlike RPA, Automation is the invention of a new technology to solve real-life problems with the need for human intervention.

Refer to the table below to look into the differences between Automation and RPA.

Parameter	Automation	RPA
What does it Reduce?	Reduces execution time	Reduces the number of people working on something.
What does it automate?	Automates repetitive test cases i.e., a product	Automates the repetitive business process i.e., product as well as business
Programming Knowledge	Programming knowledge required to create test scripts	Programming knowledge is mostly not needed as it is wizard-driven
Software Environment	Limited software environment	A wide range of software environments
Application	Used for QA, Production, Performance, UAT environments	Usually used in production environments

Lifecycle of RPA:

A typical life cycle of RPA has 4 phases. Analysis, Bot Development, Testing, and Support & Maintenance.

- **Analysis –** Business teams & RPA Architects work together to analyze a business process for RPA development.

- **Bot Development** – Developer teams start working on developing the automated workflows for the requirements in a distinct development environment.
- **Testing** – Run testing cycles such as STLC to analyze the quality and correct defects.
- **Support & Maintenance** -After the development & testing phases, a bot enters the maintenance phases in which it provides continuous support and helps in the immediate defect resolution.

After knowing the methodology, you should also know how to implement RPA. Well, the answer to this is by using the various RPA tools available in the market. RPA Tools
There are numerous tools available in the market, each providing various functionalities according to your need.
But, the top 3 tools in today's market are the trio (**UiPath**, **Blue Prism**, and **Automation Anywhere**).

Advantages of RPA:

The listed below are a few benefits of RPA.

- Multiple processes can be automated at once.
- Cost cutting technology and enhances resource optimization.
- Doesn't require prior programming knowledge.
- Supports and allows regular compliance process, with error-free auditing.
- Easy to model, scale, and deploy the automation process.
- Makes it easy to track defects.
- Continues builds & release management.
- No training period is required as it works without human intervention

Usage of RPA:

Apart from imitating human actions, repeating high volume tasks and performing multiple tasks at once. RPA can also be used to do the following:

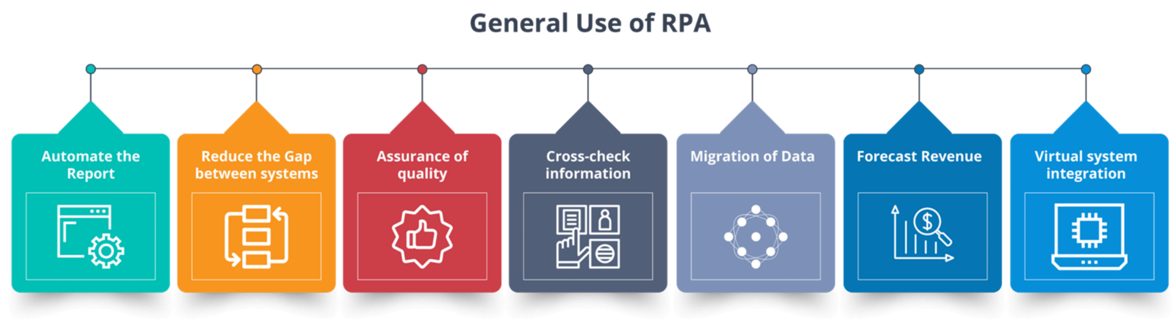


Fig 1: Usage of RPA in various fields – RPA Tutorial

- **Automate the Report Generation** – Makes accurate and timely reports by automating the process of extracting data.
- **Reduce the Gap between systems** – Reduces the gap between systems by preventing custom implementations.
- **Assurance of quality** – Delivers quality product by performing testing and automating customer use case scenarios.
- **Cross-check information** – Data across various systems is cross- verified to validate the information.
- **Migration of Data** – Unlike traditional systems, RPA allows automated data migration through systems.
- **Forecast Revenue** – Updates financial statements to predict revenue forecast automatically.
- **Virtual system integration** – Automated systems transfer data between disparate and legacy systems by connecting them at user interface level.

Industries using RPA

1. Healthcare
2. Hr
3. Telecom
4. insurance
5. Retail Industries
6. travel
7. Supply chain management

Software Testing Domains-For Reference

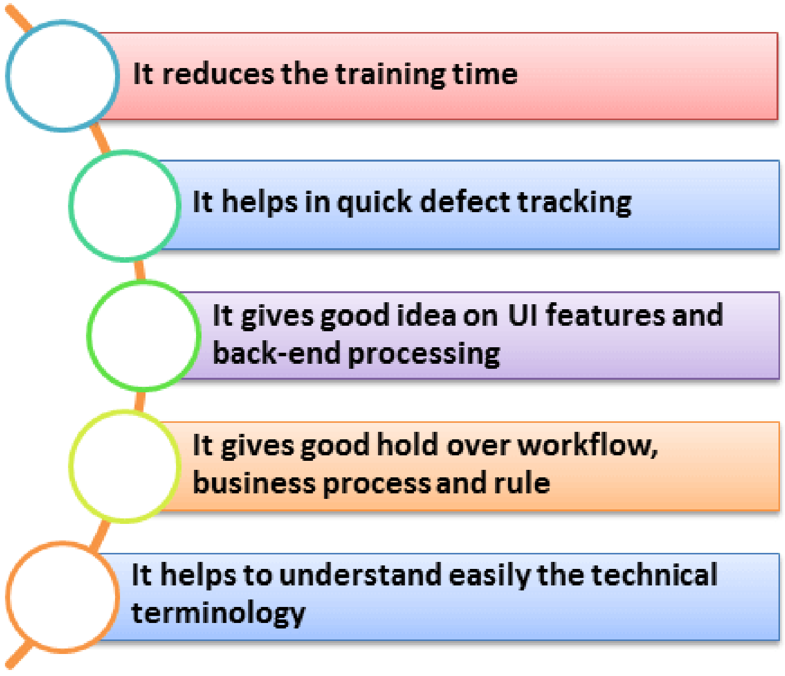
11.1 What is domain:

Domain is an area in which business is done on specialization. In software testing there are different domain such as

- Banking, Financial service & Insurance
- Health care
- Telecom and Mobile
- E-commerce
- Retail
- Airline
- Gaming
- ERP/CRM

11.2 Why in Testing Domain Knowledge Matters?

Domain knowledge is essential for testing any software product and it has its own benefits like...


It reduces the training time
It helps in quick defect tracking
It gives good idea on UI features and back-end processing
It gives good hold over workflow, business process and rule
It helps to understand easily the technical terminology

11.3 DIFFERENT IT DOMAINS

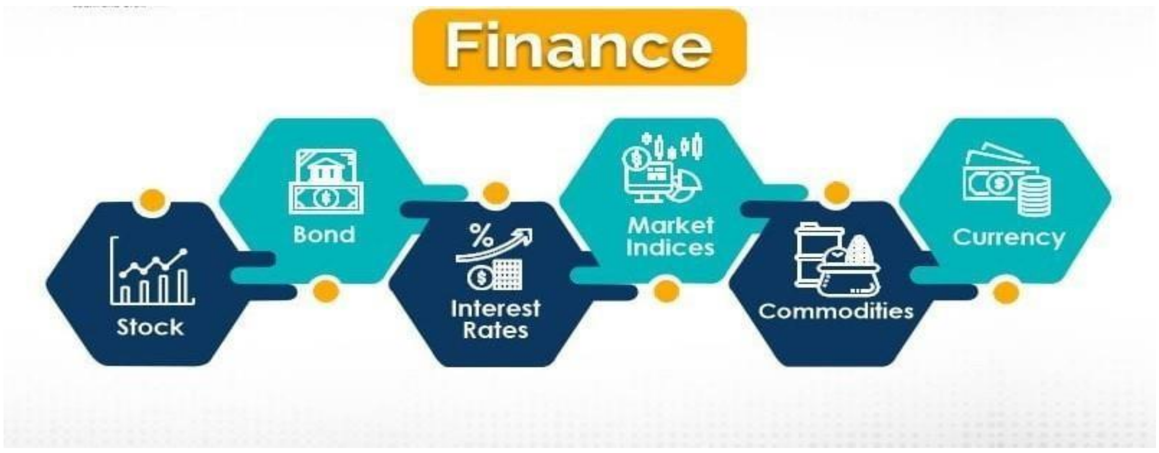
11.3.1 Banking and financial service are included:

“A bank is a financial institution that provides banking and other financial to their customers. Banks are a subset of the financial services industry and play an important role in the global economies”.

There are generally five types of banks:

- Retail Banks
- Commercial Banks/corporates Banks/Merchants Bank
- Investment Banks
- Private Banks
- Universal Banks
- Credit Domain software
- Capital Market software etc.

11.3.2 Finance Domain:



“Finance domain” term is generally used to refer to the skills and jobs that fall under finance industry or financial services. It is used as a collective term to refer to a broad range of economic services provided by the finance industry which encompasses a broad range of organizations that manage money, including credit unions, banks, credit card companies etc.

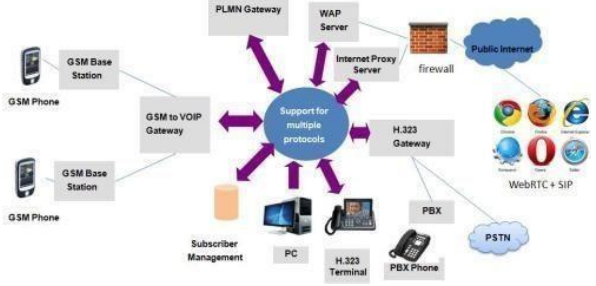
11.3.3 Health Care Domain:

Health care industry is one of the largest industries in the world and it has a direct effect on the quality of life of people. Health care area may include

- Diagnostic Imaging
- Health Care IT
- Life science
- Clinical Care areas
- Medical Diagnostics
- Interventional & surgery

11.3.4 Telecom Domain:

“Since the shift of telecom sector to computer network and digital medium, telecommunication sector has undergone a major software transformation. Telecom started depending on the various types of software components to deliver many services like routing and switching, VoIP broadband access etc.”.



Hence, that telecom testing
has become an
inevitable process the area include are as follows

- Application Testing
- Security testing
- Telecom Switch Testing.
- Telecom Billing Testing.
- OSS-BSS Testing
- Bluetooth Testing

11.3.5 E-commerce Domain:

E-commerce electronic commerce or EC is the buying and selling of goods and services, or the transmitting of funds or data, over an electronic network, primarily the internet. These business transactions occur either as business-to-business, business-to-consumer, consumer-to-consumer or consumer-to-business.

Types of Ecommerce Applications available in the IT Industry are as follows...

- Business to Business (B2B).
- Business to Customers (B2C).
- Customers to Customers (C2C).
- Customers to Business (C2B) etc...

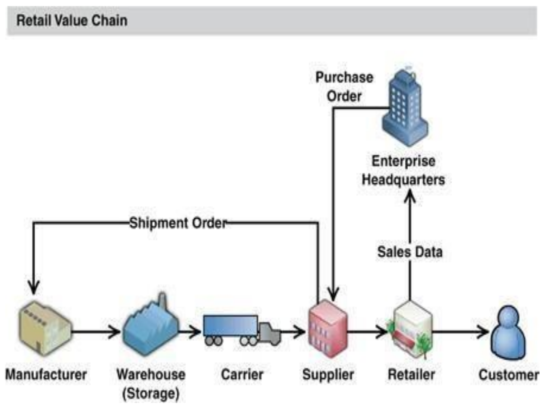
shutterstock.com · 531055792

E-commerce Website are may include:

- Homepage testing.
- Search Result page.
- Order Form page.
- Order conformation page.
- product detail page.
- Shopping Cart Testing (for storage).
- Sorting and filtering checking.
- Payments etc.

11.3.6 Retail Domain:

Retail Domain Retail is the sale of goods and services from individuals or businesses to the end-user. The **retail** industry provides consumers with goods and services for their everyday needs. Retailers are part of an integrated system called the supply-chain.



Retail area include:

- Point of Service(pos) Testing.
- Inventory planning.
- Warehouse Management System.
- Business Intelligence and Data warehouse.
- E/M commerce application testing.

11.3.7 Gaming Domain:

Gaming area may include:

- Pc based Games
- Nintendo wii
- Playstation (PS2 & PS3)
- Flash based games
- Mobile based games

Types of Game Testing:

1) Non-Functional Testing:

“Functionality testers look for the generic problems within the game or its user interface & graphics, such as game mechanic issues, stability issues, and game asset integrity. User interface testing ensures user-friendliness of the game”.

Example: Checking colors and backgrounds, menu structure, screen resolution, font size, alignment errors, usability, system navigation such as loading time, timeout and display, sorting, confirmation messages, animations and audio elements aspects of the game, instructions, and dialogue messages. User Interactions, User Interfaces, Transactions testing, Calibration and accuracy testing of mobile phone cameras, Screen resolutions, Mobile responsive design testing, Audio quality Testing.

2) Compatibility Testing:

"Checking if the game is compatible across different devices, and on different configurations of hardware and software".

Example: Install and Uninstall the game on all supported consoles/desktops/mobiles

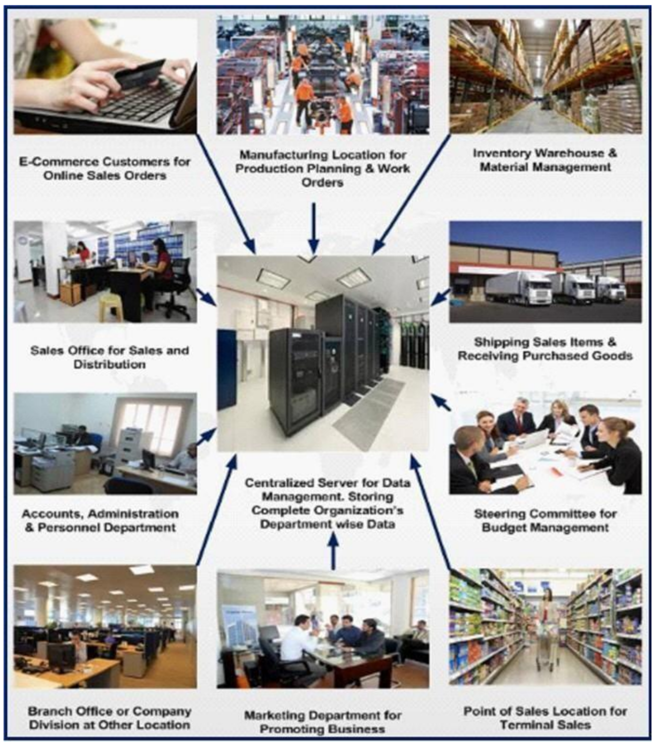
3) Performance Testing

4) Recovery testing

5) Sound Testing

6) Soak testing etc.

11.3.8 Enterprise resource planning (ERP) Domain:


E-Commerce Customers for Online Sales Orders
Manufacturing Location for Production Planning & Work Orders
Inventory Warehouse & Material Management
Sales Office for Sales and Distribution
Shipping Sales Items & Receiving Purchased Goods
Accounts, Administration & Personnel Department
Centralized Server for Data Management. Storing Complete Organization's Department wise Data
Steering Committee for Budget Management
Branch Office or Company Division at Other Location
Marketing Department for Promoting Business
Point of Sales Location for Terminal Sales

ERP is business process management software that allows an organization to use a system of integrated application to manage the business and automate many back-office functions related to technology, services and human resources.

Enterprise resource planning combines all business process into one unified system. It's purpose to make the best possible use of resources of the business to give the company a competitive edge.

ERP area may include:

a) **Finance/accounting:** General ledge payable, cash management, fixed assets, budgeting etc.

b) **Human resources:** payroll, training, diversity management

c) **Manufacturing:** Engineering, bill of material, costmanagement, manufacturing flow, Product life-cycle management.

d) **Supply chain management:** Order to cash, inventory, purchasing, supply chain planning, commissions etc.

e) **Access control:** Management of user privileges for various processes.

f) **Project management:** Costing, billing, time and expense, performance units, activity management.

11.4 Other Types of Testing

11.4.1 Database Testing:

Introduction:

What is Data?

Data can be information about entity in consideration. For example, your name, age, height, weight, etc are some data related to you.
A picture, image, file, pdf etc. can also be considered data.

What is a Database?

Database is a systematic collection of data. Databases support storage and manipulation of data. Databases make data management easy.
For example: Your electricity service provider is obviously using a database to manage billing, client related issues, to handle fault data etc.

What is Database Testing?

Database Testing is checking the schema, tables, triggers, etc. of the database under test.

It may involve creating complex queries to load/stress test the database and check its responsiveness.

It Checks data integrity and consistency.

What to test in Database testing– different components

1) Transactions:

When testing transactions, it is important to make sure that they satisfy the ACID properties.
These are the statements commonly used:

- BEGIN TRANSACTION#
- END TRANSACTION #

The Rollback statement ensures that the database remains in a consistent state.

- ROLLBACK TRANSACTION#

After these statements are executed, use a Select to make sure the changes have been reflected.

SELECT * FROM TABLENAME <tables which involve the transactions>

2) Database schema:

"A database schema is nothing more than a formal definition of the how the data is going to be organized inside a DB".

- To test database schema, we have to Identify the requirements based on which the database operates such as:
- Primary keys to be created before any other fields are created.
- Foreign keys should be completely indexed for easy retrieval and search.
- Field names starting or ending with certain characters.
- Fields with a constraint that certain values can or cannot be inserted.

3) Trigger:

When a certain event takes places on a certain table a piece of code (a trigger) can be auto-instructed to be executed.
For example: A new student joined a school. The student is taking 2 classes math and science. The student is added to the "student table". A trigger could add the student to the corresponding subject tables once he is added to the student table.

4) Field constraints – Default value, unique value and foreign key.

- Perform a front-end operation which exercises the database object condition.
- Validate the results with a SQL Query.

Conclusion:

The database is the core and critical part of almost every software application. So, DB testing of an application demands keen attention, good SQL skills, proper knowledge of DB structure of AUT and proper training.

11.4.2 MOBILE TESTING:

"**Mobile application testing** is a process by which an **application software** developed for handheld mobile devices is tested for its functionality, usability, and consistency. Mobile application testing can be automated or manual type of testing."

What is Mobile Application Testing?

- Currently Android & i OS are the most popular mobile operating system present on major mobile devices. There are millions of applications designed for these platforms that need to be tested before introducing them to the actual end users.

- Mobile application testing is a process to test mobile applications for its functionality, usability and consistency.

- Any mobile application will either come pre-installed or can be installed from the mobile software distribution platform.

Below are the types of testing that can be done for any Mobile Application:

1. Functional Testing

Functional testing is the most basic test for any application to ensure that it is working as per the defined requirements.

2. Compatibility Testing

The purpose of a mobile app compatibility test, in general, is to ensure an app's key functions behave as expected on a specific device.

3. Localization Testing

Most of the apps are designed for global use and it is very important to care about regional trails like languages, time zones, etc. It’s important to validate the app’s functionality when someone changes the time zone.

4. Performance Testing

Mobile performance test covers client application performance, server performance, and network performance.

5. Stress Testing

Stress testing is a must to find exceptions, hangs, and deadlocks that may go unnoticed during functional and user interface testing.

6. Security Testing

Verifying to hacking, authentication, and authorization policies, data security, session management and other security standards.

7. Power Consumption Testing

While we focus on power consumption testing, we are required to measure the state of the battery at each activity level

8. Interrupt Testing

An application, while functioning, may face several interruptions like incoming calls or network coverage outage and recovery.

9. Usability Testing

It should satisfy all user requirements and should check how good the application is user friendly to operate for the success of any application.

10. Installation & Uninstallation Testing

Installation testing verifies that the installation process goes smoothly without the user having to face any difficulty. Uninstallation testing can be as “Uninstallation should sweep out data related to the App in just one go”.

11. Updates Testing

It is very important that under the update testing, we qualify that the App is working properly after the update.

12. Certification Testing

To get a certificate of compliance, each mobile device needs to be tested against the guidelines set by different mobile platforms.