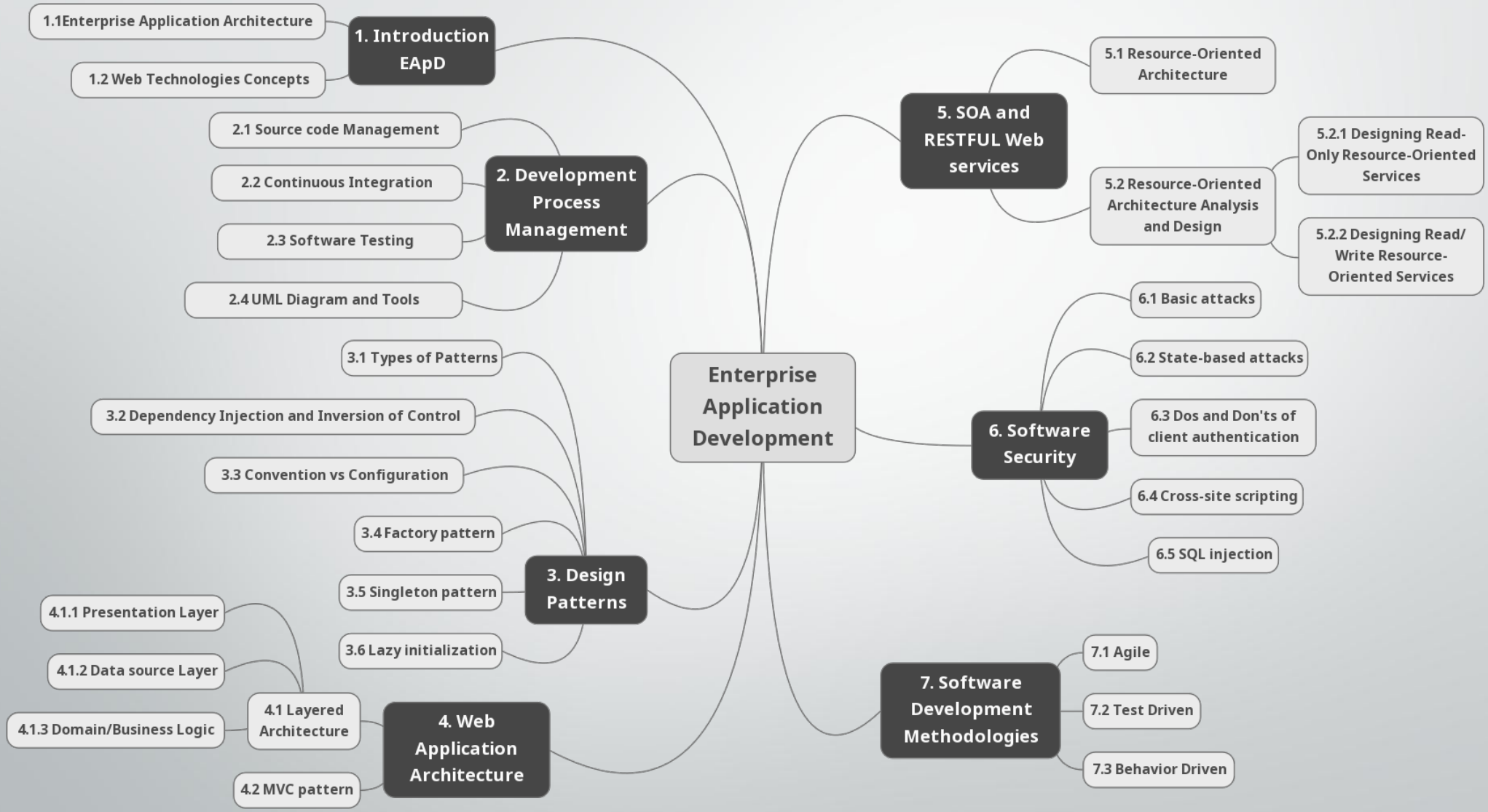
Enterprise Application Development

[BE SE-7th Semester]

Nepal College of Information Technology

POKHARA UNIVERSITY

Enterprise Application Development



2.3 Software Testing

1. Software Testing Fundamentals
2. Test Cases and Levels of Testing
3. Different Types of Testing
4. Security testing, Black box and white box testing

Software Testing Fundamentals

- The engineer creates a series of test cases that are intended to "demolish" the software that has been built.
- In fact, testing is the one step in the software process that could be viewed as destructive rather than constructive.
- Testing requires that the developer discard preconceived notions of the "correctness" of software just developed and overcome a conflict of interest that occurs when errors are uncovered.
- Therefore, testing and test case design is an admission of failure, which instills a goodly dose of guilt.

Software Testing

- Testing is an essential part of software development process.
- Testing is used to identify the correctness, completeness and quality of developed software.
- Testing is the process of executing a program or application with the intent of finding the software bugs.
- It can be stated as the process of validating and verifying software.
- *Software Testing is the process of evaluating the system or its components with the intent to find errors whether it satisfies the specified requirements or not It is the process of executing a system in order to identify gaps, errors or missing requirements in contrary to the actual desire or requirements.*

Software Testing Objectives

- To ensure that customer will be able to get his/her work done
- To find defects that may be created by the programmer while developing the software.
- To gain confidence in and providing information about the level of quality.
- To prevent defect.
- To make sure that the end result meets the business and user requirements
- To gain confidence of the customer by providing them a quality product.
- To find and remove errors that lead to software failure.

Who perform testing?

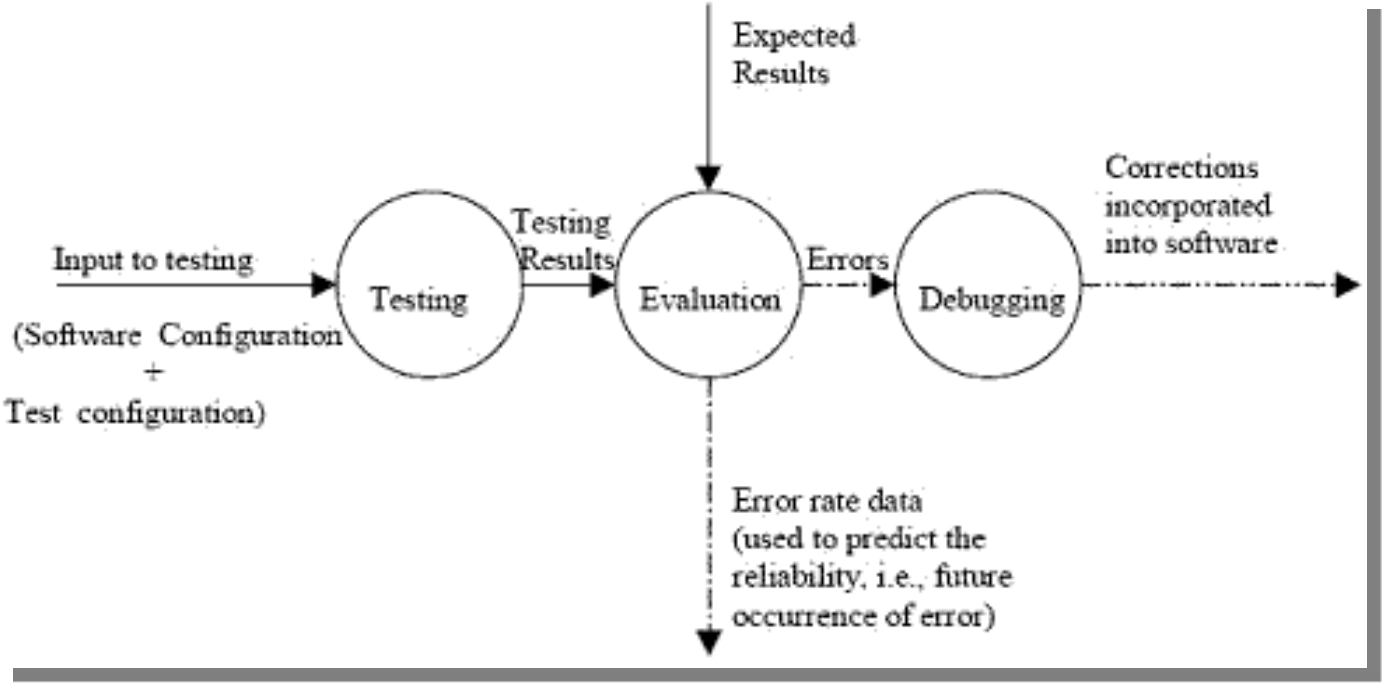
- Testing is performed either by the software developer (who originally developed the software) or by Independent Test Group(ITG), which comprises of software testers.
- ITG form a part of the software development project team.
- This is because the group becomes involved during the specification activity and strays involved (Planning and specifying test procedures) throughout the development process

How test information flow?

- Testing is a complete process. For testing, we need two types of inputs (Software Configuration and Test configuration)
- Software configuration includes software requirement specification, design specifications and source code.
- Software configuration is required so that the testers know what is to be expected and tested.
- Test configuration is a testing plan that is, the way how the testing will be conducted on the system.
- Testing configuration specifies test case and their expected value also tools for testing are to be used.
- Test cases are required to know what specific situations need to be tested.

How test information flow?

- When tests are evaluated, test results are compared with actual results and if there is some error, then debugging is done to correct the



- Testing is a way to know about quality and reliability. Error rate that is the occurrence of errors is evaluated. This data can be used to predict the occurrence of errors in future

Software Testing Life cycle

- Software Testing Life Cycle (STLC) is the testing process which is executed in systematic and planned manner.
- In STLC process, different activities are carried out to improve the quality of the product.

  [Requirement Analysis → Test Planning → Test Case Development → Environment Setup → Test Execution → Test Cycle Closure]

- Ideally, the next step is based on previous step or we can say next step cannot be started unless and until previous step is completed.
- It is possible in the ideal situation, but practically it is not always true.

STLC → 1. Requirement Analysis

- Analyzing the System Requirement specifications from the testing point of view
- Preparation of RTM that is Requirement Traceability Matrix
- Identifying the testing techniques and testing types
- Prioritizing the feature which need focused testing
- Analyzing the Automation feasibility
- Identifying the details about the testing environment where actual testing will be done

STLC → 2. Test Planning

- Analyzing the System Requirement specifications from the testing point of view
- Preparation of RTM that is Requirement Traceability Matrix
- Identifying the Estimation of testing effort
- Selection of Testing Approach
- Preparation of Test Plan, Test strategy documents
- Resource planning and assigning roles and responsibility to them
- Selection of Testing tool

STLC → 2. Test Planning

Deliverables (Outcome) of Test Planning phase are:

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

STLC → 3. Test Case Development

- In this phase the QA team writes test cases. They also write scripts for automation if required. Activities to be done in Test Case Development phase are given below:
  - Creation of test cases
  - Creation of test scripts if required
  - Verification of test cases and automation scripts
  - Creation of Test Data in testing environment
- Deliverables (Outcome) of Test Case Development phase are:
  - Test cases
  - Test scripts (for automation if required)
  - Test Data

STLC → 4. Test Environment setup

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

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

- Deliverables (Outcome) of Test Environment Setup phase are:

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

STLC → 5. Test Execution

- Activities to be done in Test Execution phase are given below:
  - Execution of Test Cases
  - Reporting test results
  - Logging defects for the failed test cases
  - Verification and retesting of the defect
  - Closure of defects
- Deliverables (Outcome) of Test Execution phase are:
  - Test execution Report
  - Updated test cases with results
  - Bug Report

STLC → 6. Test Cycle Closure

- Activities to be done in Test Cycle Closure phase are given below:
  - To evaluate the test completion on the basis of Test Coverage and Software Quality
  - Documentation of the learning from the project
  - Analyzing the test results to find out the distribution of severe defects
  - Test Closure Report preparation
- Deliverables (Outcome) of Test Cycle Closure phase are:
  - Report of Test Closure

Test Cases

- We now know, test cases are **integral part of testing**.
- So we need to know more about test cases and how these test cases are designed.
- The most desired or obvious expectation from a test case is that it should be able to find most errors with the least amount of time and effort.
- Test cases are a set of conditions or variable under which a tester will determine whether a system under test satisfies requirements or works correctly.

Writing Good Test Cases

- Write test cases in such a way that you test only one thing at a time. Do not overlap or complicate test cases.
- Attempt to make test cases —atomic
- Ensure all positive and negative scenarios are covered.
- Writing in simple and easy to understand.
- Use active voice: Do this, Do that.
- Use exact and consistent names (of fields, forms, etc.).

STLC → Test Case Example

Test Scenario ID	Login-1	Test Case ID	Login-1B
Test Case Description	Login – Negative test case	Test Priority	High
Pre-Requisite	NA	Post-Requisite	NA

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Launch application	https://www.facebook.com/	Facebook home	Facebook home	IE-11	Pass	[Priya 10/17/2017 11:44 AM]: Launch successful
2	Enter invalid Email & any Password and hit login button	Email id : invalid@xyz.com Password: ******	The email address or phone number that you've entered doesn't match any account. Sign up for an account.	The email address or phone number that you've entered doesn't match any account. Sign up for an account.	IE-11	Pass	[Priya 10/17/2017 11:45 AM]: Invalid login attempt stopped
3	Enter valid Email & incorrect Password and hit login button	Email id : valid@xyz.com Password: ******	The password that you've entered is incorrect. Forgotten password?	The password that you've entered is incorrect. Forgotten password ?	IE-11	Pass	[Priya 10/17/2017 11:46 AM]: Invalid login attempt stopped

Characteristics of good test case

1) Accurate: Exacts the purpose.
2) Economical: No unnecessary steps or words.
3) Traceable: Capable of being traced to requirements.
4) Repeatable: Can be used to perform the test over and over.
5) Reusable: Can be reused if necessary.

Levels of testing

Unit testing

- To test each module (unit, or component) independently
- Mostly done by developers of the modules

Integration and system testing

- To test the system as a whole
- Often done by separate testing or QA team

Acceptance testing

- To validate system functions for (and by) customers or user

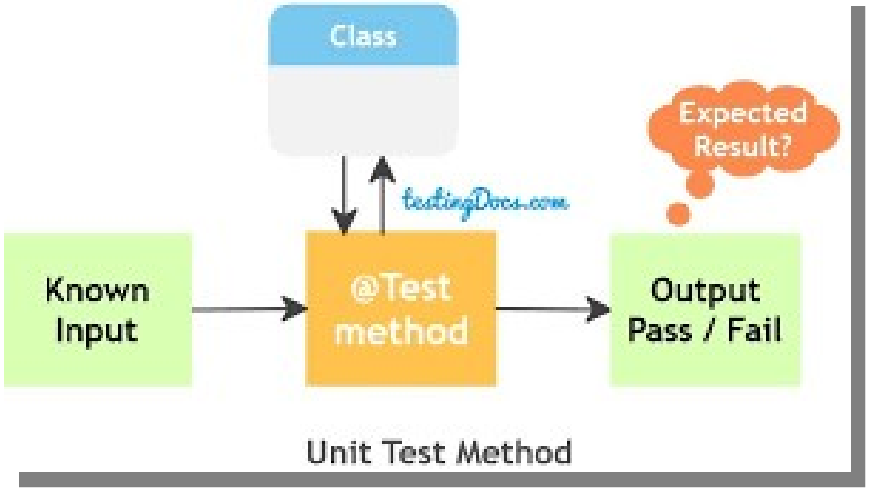
Unit Testing

A Unit testing is a Level of Testing where smallest part of individual unit / component (called unit) is tested to determine if they are fit for use.

- The unit test cases writing and execution is done by the developer (not the tester) to make sure that individual units are working as expected.
- The smallest part of individual components like functions, procedures, classes, interfaces etc.
- If we take a example of functions, when we pass input parameters to functions then check if the function should return a expected values. The main intention of this activity is to check whether units are working as per design and handling error and exception more neatly.

Unit testing

- The both positive and negative conditions should handle properly.
- The white box testing is used to test the unit testing.
- This is very first stepping in level of testing and started before doing integration testing.

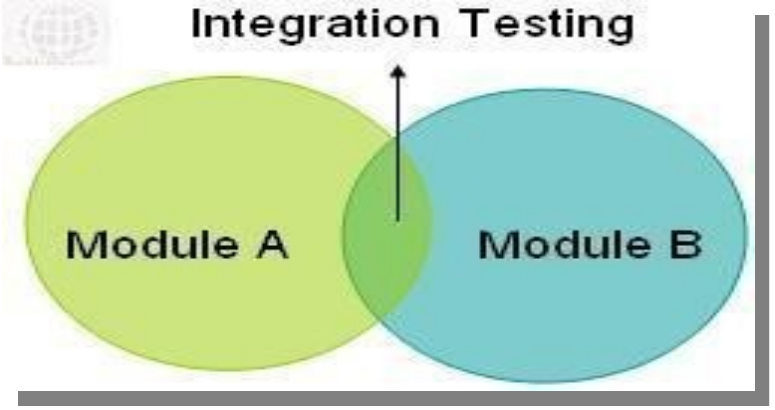


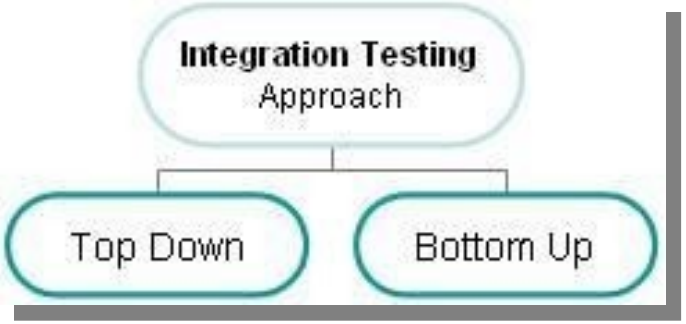
Unit Testing Test Case Preparation Guidelines:

1) Unit Test Plan/Cases should be made a separate deliverable. It should not be merged with other artifacts. Try to document all the probable test scenarios which encompass uncommon and alternative flows.
2) Construction and Unit testing need to be made distinct phases and the deliverable need to be scheduled accordingly.
3) If Construction and UT are scheduled as a single phase, Unit testing results need to be made as a separate deliverable
4) Make use of the count of test cases planned, executed, passed, and failed to apprehend the progress and replicate Unit testing if required.
5) Try to include on-the-fly test cases that are developed while executing a pre-defined set of test cases.

Integration Testing

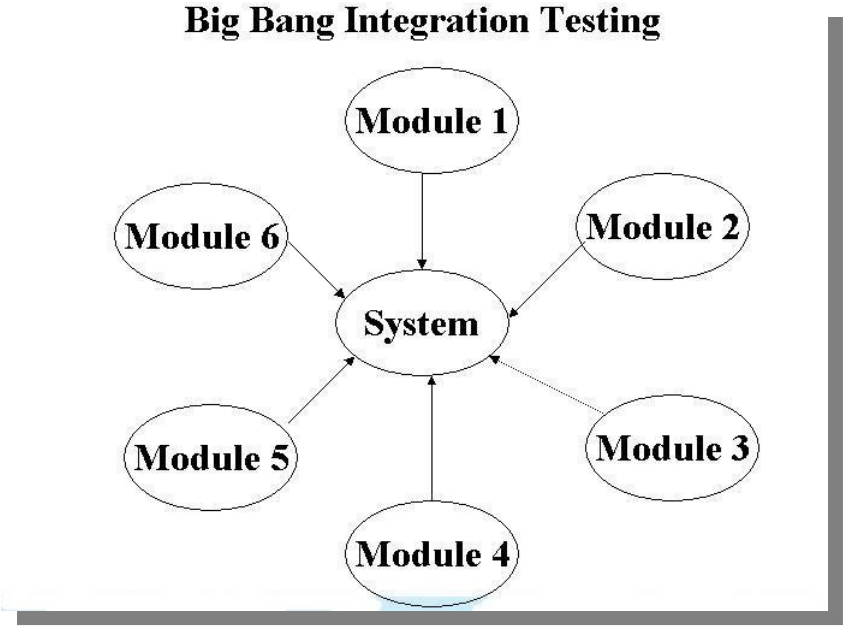
- Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.
- Also after integrating two different components together we do the integration testing.
- Integration testing is done by a specific integration tester or test team.





1. Big Bang integration testing:

all components or modules are integrated simultaneously, after which everything is tested as a whole. As per the below image all the modules from Module 1‘ to Module 6‘ are integrated simultaneously then the testing is carried out.



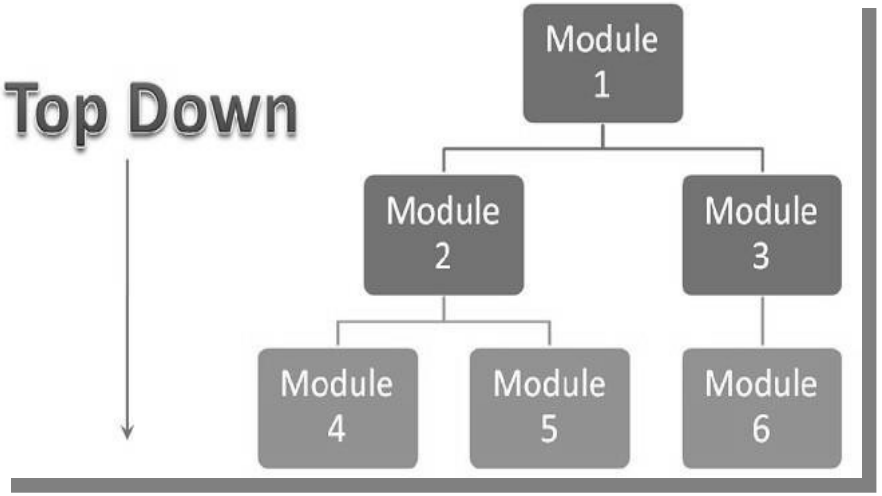
- **Advantage**: Big Bang testing has the advantage that everything is finished before integration testing starts.

- **Disadvantage**:

  The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.

2. Top-down integration testing:

- Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu).
- Components or systems are substituted by stubs. Below is the diagram of _Top down Approach‘:



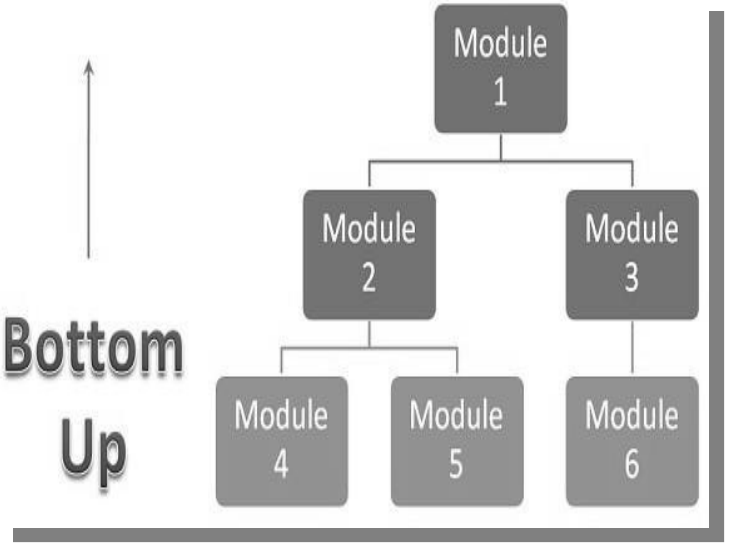
Advantages :
The tested product is very consistent because the integration testing is basically performed in an environment that almost similar to that of reality

- **Disadvantages**:

Basic functionality is tested at the end of cycle

3. Bottom-up integration testing

- Testing takes place from the bottom of the control flow upwards.
- Components or systems are substituted by drivers.
- **Advantage**:
  In this approach development and testing can be done together so that the product or application will be efficien and as per the customer specifications.
- **Disadvantages**:
  We can catch the Key interface defects at the end of cycle
- It is required to create the test drivers for modules at all levels except the top control



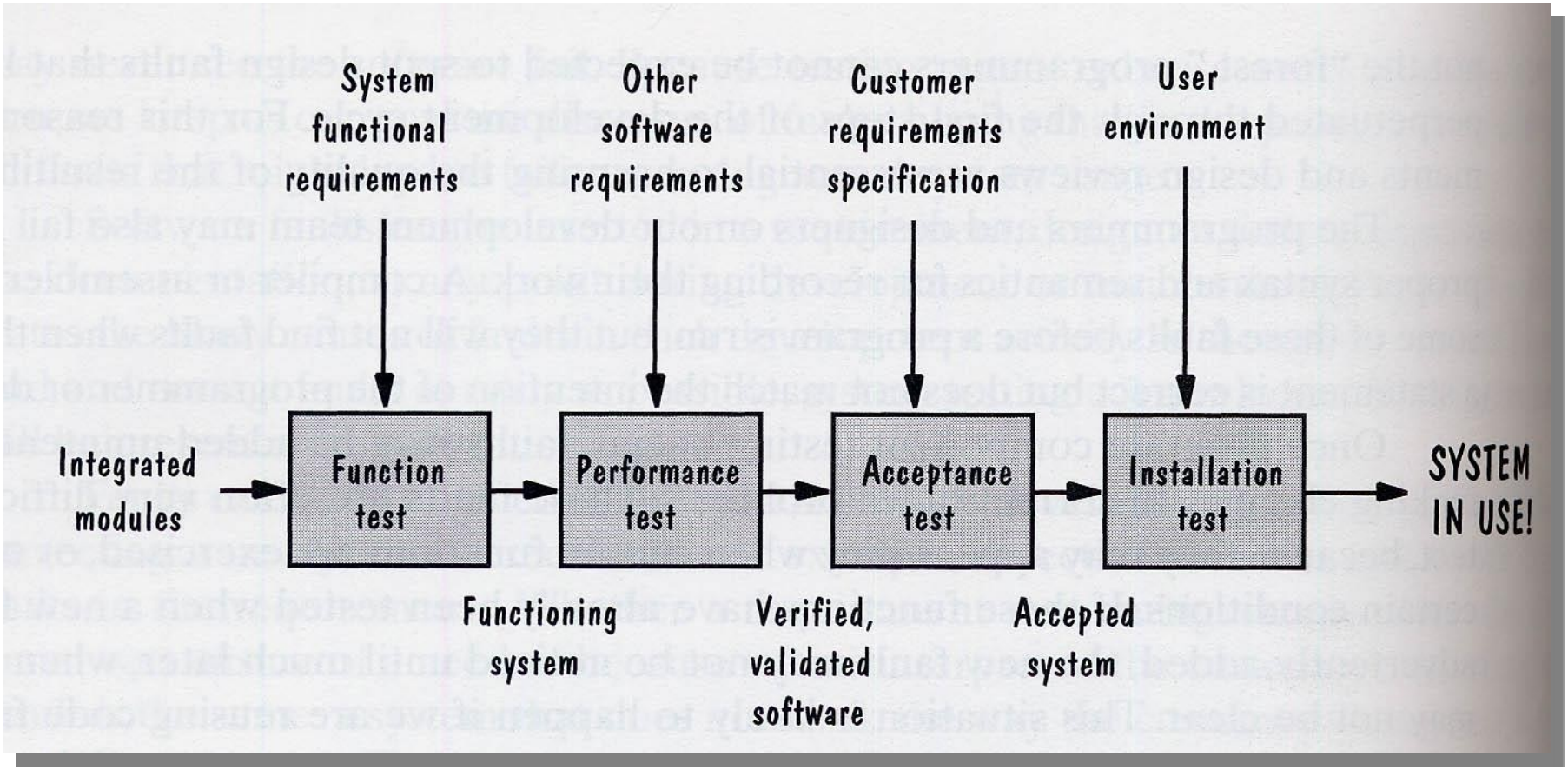
Stubs and drivers

- Stubs and drivers are dummy programs written while integration testing. Stubs are used during top down testing.
- In this type highest level components are created first. To test if component written will function correctly when integrated with lower level components a dummy program for lower level component is written as a substitute of actual code which is called **Stubs**.
- Stubs will contain only functionality needed to be successfully called by a higher level component. It will simulate the behavior of a lower level component.
- In bottom up approach, lower level components are created first.
- Temporary components called 'drivers' are written as substitutes for the missing code. Then the lowest level components can be tested using the test driver.

System testing

- In system testing the behavior of whole system/product is tested as defined by the scope of the development project or product.
- It may include tests based on risks and/or requirement specifications, business process, use cases, or other high level descriptions of system behavior, interactions with the operating systems, and system resources.
- System testing is most often the final test to verify that the system to be delivered meets the specification and its purpose.
- System testing is carried out by specialist‘s testers or independent testers.
- System testing should investigate both functional and non-functional requirements of the testing.

System testing



System Testing Process

- **Function Testing:** the system must perform functions specified in the requirements.
- **Performance Testing**: the system must satisfy security, precision, load and speed constrains specified in the requirements.
- **Acceptance Testing**: customers try the system (in the lab) to make sure that the system built is the system they requested.
- **Deployment Testing**: the software is deployed and tested in the production environment.

Acceptance testing

- After the system test has corrected all or most defects, the system will be delivered to the user or customer for Acceptance Testing or User Acceptance Testing (UAT).
- Acceptance testing is basically done by the user or customer although other stakeholders may be involved as well.
- The goal of acceptance testing is to establish confidence in the system.
- Acceptance testing is most often focused on a validation type testing.
- Acceptance testing may occur at more than just a single level, for example:
- A Commercial Off the shelf (COTS) software product may be acceptance tested when it is installed or integrated.
- Acceptance testing of a new functional enhancement may come before system testing

The types of acceptance testing:

- **The User Acceptance test:** focuses mainly on the functionality thereby validating the fitness-for-use of the system by the business user.
- **The Operational Acceptance test**: also known as Production acceptance test validates whether the system meets the requirements for operation. The operational acceptance test may include testing of backup/restore, disaster recovery, maintenance tasks and periodic check of security vulnerabilities.
- **Contract Acceptance testing:** It is performed against the contract‘s acceptance criteria for producing custom developed software. Acceptance should be formally defined when the contract is agreed.
- **Compliance acceptance testing:** It is also known as regulation acceptance testing is performed against the regulations which must be adhered to, such as governmental, legal or safety regulations.

Levels of testing

Unit testing

- To test each module (unit, or component) independently
- Mostly done by developers of the modules

Integration and system testing

- To test the system as a whole
- Often done by separate testing or QA team

Acceptance testing

- To validate system functions for (and by) customers or user

Alpha Testing and Beta Testing

- Alpha testing is one of the most common software testing strategies used in software development. It‘s specially used by product development organizations.
- This test takes place at the developer’s site. Developers observe the users and note problems.
- Alpha testing is testing of an application when development is about to complete. Minor design changes can still be made as a result of alpha testing.
- Alpha testing is typically performed by a group that is independent of the design team, but still within the company, e.g. in-house software test engineers, or software QA engineers.

Alpha Testing and Beta Testing

- Alpha testing is final testing before the software is released to the general public. It has two phases:
- In the **first** phase of alpha testing, the software is tested by in-house developers. They use either debugger software, or hardware-assisted debuggers. The goal is to catch bugs quickly.
- In the **second** phase of alpha testing, the software is handed over to the software QA staff, for additional testing in an environment that is similar to the intended use.
- Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers‘ site
- Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

Alpha Testing and Beta Testing

- Beta Testing is also known as field testing. It takes place at customer's site. It sends the system/software to users who install it and use it under real- world working conditions.
- A beta test is the second phase of software testing in which a sampling of the intended audience tries the product out. (Beta is the second letter of the Greek alphabet.) Originally, the term alpha testing meant the first phase of testing in a software development process. The first phase includes unit testing, component testing, and system testing. Beta testing can be considered —pre-release testing.
- The goal of beta testing is to place your application in the hands of real users outside of your own engineering team to discover any flaws or issues from the user‘s perspective that you would not want to have in your final, released version of the application.
- Example: Microsoft and many other organizations release beta versions of their products to be tested by users.

Open and Closed Beta

- Developers release either a closed beta or an open beta;
- Closed beta versions are released to a select group of individuals for a user test and are invitation only, while
- Open betas are from a larger group to the general public and anyone interested.
- The testers report any bugs that they find, and sometimes suggest additional features they think should be available in the final version.
- We have the opportunity to get your application into the hands of users prior to releasing it to the general public.

Open and Closed Beta

- Users can install, test your application, and send feedback to you during this beta testing period.
- Your beta testers can discover issues with your application that you may have not noticed, such as confusing application flow, and even crashes.
- Using the feedback you get from these users, you can fix problems before it is released to the general public.
- The more issues you fix that solve real user problems, the higher the quality of your application when you release it to the general public.
- Having a higher-quality application when you release to the general public will increase customer satisfaction.
- These users, who are early adopters of your application, will generate excitement about your application.

Static Testing vs Dynamic Testing

- Static testing is the testing of the software work products manually, or with a set of tools, but they are not executed.
- It starts early in the Life cycle and so it is done during the verification process.
- It does not need computer as the testing of program is done without executing the program. For example: reviewing, walk through, inspection, etc.
- Most static testing techniques can be used to ‗test‘ any form of document including source code, design documents and models, functional specifications and requirement specifications.

Static Testing vs Dynamic Testing

- Since static testing can start early in the life cycle so early feedback on quality issues can be established.
- As the defects are getting detected at an early stage so the rework cost most often relatively low.
- Development productivity is likely to increase because of the less rework effort.
- Types of the defects that are easier to find during the static testing are: deviation from standards, missing requirements, design defects, non- maintainable code and inconsistent interface specifications.
- Static tests contribute to the increased awareness of quality issues.

Dynamic Testing

- This testing technique needs computer for testing.
- It is done during Validation process.
- The software is tested by executing it on computer.
- Example of this Dynamic Testing Technique: Unit testing, integration testing, and system testing.

Manual vs Automatic Testing

- Whether you want the manual testing or the automation for your application, it depends on many factors.
- To make the right decision, right information is needed. They both have their own importance and place in the world of testing.
- We cannot underestimate the power, time saving capability, and accuracy of the automation tools and we cannot deny the creativity and out of box thinking of a manual tester.
- Yes, both are important and have their own role in testing software.
- The question here is that, how do you make a decision of what to choose which works best for your project and takes care of all the testing needs and how to proceed with it? Is only one of them is required or balances of both will work the best?

Manual vs Automatic Testing

- **Key points** for deciding Manual vs Automatic Testing
- Type of project
- Number of Regression tests
- Skillset of the software testing team
- Budget
- Need of Random / Exploratory Testing
- Team size
- Testing as long term equity

Testers workbench

- To understand the technology of software testing it is necessary to understand the workbench concept.
- A Workbench is a method of documenting how a particular activity must be fulfilled.
- A workbench is referred to a stages, steps, and assignments.
- In the sequence of comprehension testing techniques, we must know the concept of workbench.
- Normally a workbench concept is a method of scheduling that how a particular action has to be executed.
- Workbench concept normally contains the following actions mentioned as below:-

  [Input → Execute → Check → Production Output → Rework]

11- steps of testing process

- Step 1: Assess Development Plan and Status
- Step 2: Develop the Test Plan
- Step 3: Test Software Requirements
- Step 4: Test Software Design
- Step 5: Program (Build) Phase Testing
- Step 6: Execute and Record Results
- Step 7: Acceptance Test
- Step 8: Report Test Results
- Step 9: The Software Installation
- Step 10: Test Software Changes
- Step 11: Evaluate Test Effectiveness

Different Types of Testing

- Installation Testing
- Usability Testing
- Regression Testing
- Performance Testing
- Load Testing
- Stress Testing
- Security Testing

Installation Testing

- Installation testing is check that software application is successfully installed & it is working as expected after installation.
- This is testing phase prior to end users will firstly interact with the actual application.
- Installation testing is also called as —Implementation Testing.
- This is most important as well as most interesting step in the Software testing life cycle.
- Installation testing tips with some broad test cases:

1) Install Full Version Of Application

2) Automate testing efforts

3) use of disk image to be installing on dedicated machine

Installation Testing Guideline

4) Required Disk Space check in installation
5) Group the different file system format(if any) the disk space may vary like on NTFS, FAT32, and FAT16 etc
6) Use of Distributed Testing Environment
7) Automate the check of files installed after installation
8) Confirm for registry changes after installation
9) Negative testing in Installation Testing
10) Uninstallation testing

Components of Usability testing

- **Learnability**: How easy is it for users to accomplish basic tasks the first time they encounter the design?
- **Efficiency**: How fast can experienced users accomplish tasks?
- **Memorability**: When users return to the design after a period of not using it, does the user remember enough to use it effectively the next time, or does the user have to start over again learning everything?
- **Errors**: How many errors do users make, how severe are these errors and how easily can they recover from the errors?
- **Satisfaction**: How much does the user like using the system?

Advantages of Usability testing

- Usability test can be modified to cover many other types of testing such as functional testing, system integration testing, unit testing, smoke testing etc.
- Usability testing can be very economical if planned properly, yet highly effective and beneficial.
- If proper resources (experienced and creative testers) are used, usability test can help in fixing all the problems that user may face even before the system is finally released to the user. This may result in better performance and a standard system.
- Usability testing can help in discovering potential bugs and potholes in the
- system which generally are not visible to developers and even escape the other type of testing.

Regression Testing

- When any modification or changes are done to the application or even when any small change is done to the code then it can bring unexpected issues. Along with the new changes it becomes very important to test whether the existing functionality is intact or not. This can be achieved by doing the regression testing.
- The purpose of the regression testing is to find the bugs which may get introduced accidentally because of the new changes or modification.
- During confirmation testing the defect got fixed and that part of the application started working as intended. But there might be a possibility that the fix may have introduced or uncovered a different defect elsewhere in the software.
- The way to detect these unexpected side-effects‘ of fixes is to do regression testing.
- This also ensures that the bugs found earlier are NOT creatable.

Regression Testing

- Usually the regression testing is done by automation tools because in order to fix the defect the same test is carried out again and again and it will be very tedious and time consuming to do it manually.
- During regression testing the test cases are prioritized depending upon the changes done to the feature or module in the application. The feature or module where the changes or modification is done that entire feature is taken into priority for testing.
- This testing becomes very important when there are continuous modifications or enhancements done in the application or product. These changes or enhancements should NOT introduce new issues in the existing tested code.
- This helps in maintaining the quality of the product along with the new changes in the application.

An Example of Regression Testing

- Let‘s assume that there is an application which maintains the details of all the students in school. This application has four buttons Add, Save, Delete and Refresh. All the buttons functionalities are working as expected.
- Recently a new button Update‘ is added in the application. This Update‘ button functionality is tested and confirmed that it‘s working as expected. But at the same time it becomes very important to know that the introduction of this new button should not impact the other existing buttons functionality.
- Along with the Update‘ button all the other buttons functionality are tested in order to find any new issues in the existing code. This process is known as regression testing.

Regression testing techniques

- **Corrective Regression Testing**: Corrective regression testing can be used when there is no change in the specifications and test cases can be reused.
- **Progressive Regression Testing**: Progressive regression testing is used when the modifications are done in the specifications and new test cases are designed.
- **Retest-All Strategy**: The retest-all strategy is very tedious and time consuming because here we reuse all test which results in the execution of unnecessary test cases. When any small modification or change is done to the application then this strategy is not useful.
- **Selective Strategy:** In selective strategy we use a subset of the existing test cases to cut down the retesting effort and cost. If any changes are done to the program entities, e.g. functions, variables etc., and then a test unit must be rerun. Here the difficult part is to find out the dependencies between a test case and the program entities it covers.

Regression testing

When to use It??

[Any new feature is added, Any enhancement is done, Any bug is fixed, Any performance related issue is fixed]

Advantages of Regression testing:

- It helps us to make sure that any changes like bug fixes or any enhancements to the module or application have not impacted the existing tested code.
- It ensures that the bugs found earlier are NOT creatable.
- Regression testing can be done by using the automation tools
- It helps in improving the quality of the product.

Disadvantages of Regression testing

- If regression testing is done without using automated tools then it can be very tedious and time consuming because here we execute the same set of test cases again and again.
- Regression test is required even when a very small change is done in the code because this small modification can bring unexpected issues in the existing functionality.

Performance Testing

- Performance testing is the process of determining the speed or effectiveness of a computer, network, software program or device.
- Performance Testing is done to ensure that the software will perform as expected in the production environment.
- The outcome of the this testing helps the testers in the finding the gaps between the expected and actual result.
- To achieve maximum success in performance testing, some criteria should be defined which would measure and compare the actual output of performance testing.
- Different metrics are used to measure the output of performance testing. Information obtained from these metrics helps the companies to increase their productivity levels, decrease the error rate and provide excellent quality service level to achieve their business goals.

Performance Testing

- In an application there are many performance keys generated during performing performance testing.
- It is not possible to thoroughly study each of these testing counters and work on them.
- A performance tester should have a strong knowledge of the weak areas of the application under test and the right way to set these performance parameters.
- Now we will understand the performance metrics and the potential weak areas of web applications and the performance parameters of these areas which need to be kept under an eye during performance testing.
- Performance Measures
  [Throughput, Response Time, Tuning, Bench-marking]

Load Testing

- Load testing is a type of non-functional testing.
- A load test is type of software testing which is conducted to understand the behavior of the application under a specific expected load.
- Load testing is performed to determine a system‘s behavior under both normal and at peak conditions.
- It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation. E.g. If the number of users are increased then how much CPU, memory will be consumed, what is the network and bandwidth response time.
- Load testing can be done under controlled lab conditions to compare the capabilities of different systems or to accurately measure the capabilities of a single system.

Load Testing

- Load testing involves simulating real-life user load for the target application. It helps you determine how your application behaves when multiple users hits it simultaneously.
- Load testing differs from stress testing, which evaluates the extent to which a system keeps working when subjected to extreme workloads or when some of its hardware or software has been compromised.
- The primary goal of load testing is to define the maximum amount of work a system can handle without significant performance degradation.
- Examples of load testing include:
- Downloading a series of large files from the internet.
- Running multiple applications on a computer or server simultaneously.
- Assigning many jobs to a printer in a queue.
- Subjecting a server to a large amount of traffic.
- Writing and reading data to and from a hard disk continuously.

Stress Testing

- It is a type of non-functional testing.
- It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results.
- It is a form of software testing that is used to determine the stability of a given system.
- It put greater emphasis on robustness, availability, and error handling under a heavy load, rather than on what would be considered correct behavior under normal circumstances.
- The goals of such tests may be to ensure the software does not crash in conditions of insufficient computational resources (such as memory or disk space).

Examples of stress testing

- Double the baseline number for concurrent users/HTTP connections
- Randomlyshut down and restart ports on the network
- switches/routers that connect the servers (via SNMP commands for example)
- Restart the offline database

Security Testing

- It is a type of non-functional testing.
- Security testing is basically a type of software testing that‘s done to check whether the application or the product is secured or not. It checks to see if the application is vulnerable to attacks, if anyone hack the system or login to the application without any authorization.
- It is a process to determine that an information system protects data and maintains functionality as intended.
- The security testing is performed to check whether there is any information leakage in the sense by encrypting the application or using wide range of software‘s and hardware‘s and firewall etc.
- Software security is about making software behave in the presence of a malicious attack.
- The six basic security concepts that need to be covered by security testing are: confidentiality, integrity, authentication, availability, authorization and non-repudiation.

Why Security Testing?

- For Finding Loopholes
- For identifying Design Insecurities
- For identifying Implementation Insecurities
- For identifying Dependency Insecurities and Failures
- For Information Security
- For Process Security
- For Internet Technology Security
- For Communication Security
- For Improving the System
- For confirming Security Policies
- For Organization wide Software Security
- For Physical Security

Security Testing Techniques

- **OS Hardening**
  [Configure and Apply Patches, Updating the Operating System, Disable or Restrict unwanted Services and Ports, Lock Down the Ports, Manage the Log Files, Install Root Certificate, Protect from Internet Misuse and be Cyber Safe, Protect from Malware]
- **Vulnerability Scanning**
  [Identify Known Vulnerabilities, Scan Intrusively for Unknown Vulnerabilities]
- **Penetration Testing**
  [Simulating Attack from a Malicious Source, Includes Network Scanning and Vulnerability Scanning, Simulates Attack from someone Unfamiliar with the System, Simulates Attack by having access to Source Code, Network, Passwords]

Security Testing Techniques

- **Port Scanning and Service Mapping**
  [Identification and locating of Open Ports, Identification of Running Services]
- **Firewall Rule Testing**
  [Identify Inappropriate or Conflicting Rules , Appropriate Placement of Vulnerable Systems behind Firewall, Discovering Administrative Backdoors or Tunnels]
- **Network Scanning**
  [Identifying Active Hosts on a network, Collecting IP addresses that can be accessed over the Internet, Collecting OS Details, System Architecture and Running Services, Collecting Network User and Group names, Collecting Routing Tables and SNMP data]

Security Testing Techniques

- **Password Cracking**
  Collecting Passwords from the Stored or Transmitted Data, Using Brute Force and Dictionary Attacks, Identifying Weak Passwords
- **Ethical Hacking**
  Penetration Testing, Intrusion Testing and Red Teaming
- **File Integrity Testing**
  [Verifying File Integrity against corruption using Checksum, Session Hijacking, Exploitation of Valid Computer Session, Exploitation of the Web Session control mechanism, Gain unauthorized access to the Web Server
- **Phishing**
  Masquerading as a trustworthy entity in an electronic communication, Acquiring usernames, passwords and credit card details
- U**RL Manipulation**
  Make a web server Deliver inaccessible web pages, URL Rewriting]

Black Box and White Box Testing

- Functional Testing (**Black Box**)
- Functionality testing is performed to verify that a software application performs and functions correctly according to design specifications.
- During functionality testing we check the core application functions, text input, menu functions and installation and setup on localized machines, etc.
- Functionality testing verifies that an application is still fully functional after localization.
- Even applications which are professionally internationalized according to world-readiness guidelines require functionality testing

Black Box and White Box Testing

- The following is needed to be checked during the functionality testing:
- Installation and setup on localized machines running localized operating systems and local code pages.
- Text input, including the use of extended characters or non-Latin scripts.
- Core application functions.
- String handling, text, and data, especially when interfacing with non-Unicode applications or modules.
- Regional settings default.
- Text handling (such as copying, pasting, and editing) of extended characters, special fonts, and non-Latin scripts.
- Accurate hot-key shortcuts without any duplication.

Black Box Testing

- The technique of testing without having any knowledge of the interior workings of the application is Black Box testing.
- The tester is unaware to the system architecture and does not have access to the source code.
- Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.
- Well suited and efficient for large code segments.
- Code Access not required.
- Clearly separates user's perspective from the developer's perspective through visibly defined roles.
- Large numbers of moderately skilled testers can test the application with no
- knowledge of implementation, programming language or operating systems.

Disadvantages of Black Box Testing

- Limited Coverage since only a selected number of test scenarios are actually performed.
- Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
- Blind Coverage, since the tester cannot target specific code segments or error prone areas.
- The test cases are difficult to design

Black Box and White Box Testing

- Structural Testing (White Box)
- The structural testing is the testing of the structure of the system or component.
- Structural testing is often referred to as ‘white box‘ or ‘glass box‘ or ‘clear-box testing‘ because in structural testing we are interested in what is happening ‘inside the system/application‘.
- In structural testing the testers are required to have the knowledge of the internal implementations of the code. Here the testers require knowledge of how the software is implemented, how it works.
- During structural testing the tester is concentrating on how the software does it. For example, a structural technique wants to know how loops in the software are working.

Black Box and White Box Testing

- Different test cases may be derived to exercise the loop once, twice, and many times.
- This may be done regardless of the functionality of the software.
- Structural testing can be used at all levels of testing. Developers use structural testing in component testing and component integration testing, especially where there is good tool support for code coverage.
- Structural testing is also used in system and acceptance testing, but the structures are different.
- For example, the coverage of menu options or major business transactions could be the structural element in system or acceptance testing.

White Box Testing

- White box testing is the detailed investigation of internal logic and structure of the code.
- White box testing is also called glass testing or open box testing.
- In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code.
- The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.
- As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.
- It helps in optimizing the code.
- Extra lines of code can be removed which can bring in hidden defects.
- Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.

Disadvantages of White Box Testing

- Due to the fact that a skilled tester is needed to perform white box testing, the costs are increased.
- Sometimes it is impossible to look into every nook and corner to find out
- hidden errors that may create problems as many paths will go untested.
- It is difficult to maintain white box testing as the use of specialized tools like code analyzers and debugging tools are required.

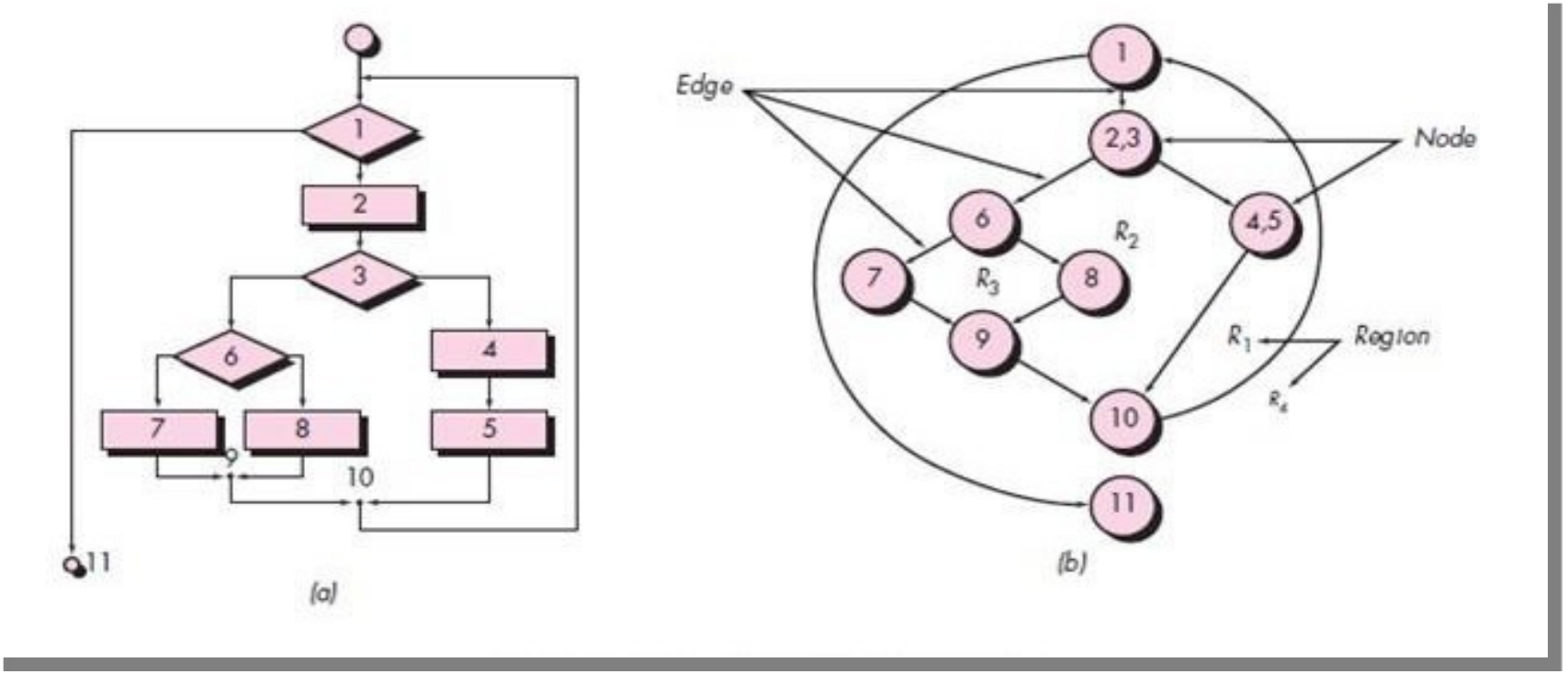
Basis Path Testing

- White-box technique usually based on the program flow graph. It derives logical complexity measures of a procedural design for defining a basis set of execution paths.
- Test cases produced to exercise each statement in the program at least one time during testing.
- The cyclomatic complexity of the program computed from its flow graph using the formula V(G) = E – N + 2 or by counting the conditional statements in the PDL representation and adding 1
- Determine the basis set of linearly independent paths (the cardinality of this set id the program cyclomatic complexity)
- Prepare test cases that will force the execution of each path in the basis set.

Flow graph notation

- The flow graph can be used to represent the logical flow control and therefore all the execution paths that need testing.
- To illustrate the use of flow graphs consider the procedural design depicted in the flow chart below.
- This is mapped into the flow graph below where the circles are nodes that represent one or more procedural statements and the arrow in the flow graph called edges represent the flow control.
- Each node that includes a condition is known as a predicate node, and has two or more edges coming from it.

Flow graph notation



Control Structure Testing

- White-box techniques focusing on control structures present in the software
- Condition testing (e.g. branch testing) focuses on testing each decision statement in a software module, it is important to ensure coverage of all logical combinations of data that
- may be processed by the module (a truth table may be helpful)
- Data flow testing selects test paths based according to the locations of variable definitions and uses in the program (e.g. definition use chains)
- Loop testing focuses on the validity of the program loop constructs (i.e. simple loops, concatenated loops, nested loops, unstructured loops), involves checking to ensure loops start and stop when they are supposed to (unstructured loops should be redesigned whenever possible)

Graph-based Testing Methods

- Black-box methods based on the nature of the relationships (links) among the program objects (nodes), test cases are designed to traverse the entire graph
- Transaction flow testing (nodes represent steps in some transaction and links represent logical connections between steps that need to be validated)
- Finite state modeling (nodes represent user observable states of the software and links represent transitions between states)
- Data flow modeling (nodes are data objects and links are transformations from one data object to another)
- Timing modeling (nodes are program objects and links are sequential connections between these objects, link weights are required execution times)

Graph-based Testing Methods

- Black-box methods based on the nature of the relationships (links) among the program objects (nodes), test cases are designed to traverse the entire graph
- Transaction flow testing (nodes represent steps in some transaction and links represent logical connections between steps that need to be validated)
- Finite state modeling (nodes represent user observable states of the software and links represent transitions between states)
- Data flow modeling (nodes are data objects and links are transformations from one data object to another)
- Timing modeling (nodes are program objects and links are sequential connections between these objects, link weights are required execution times)

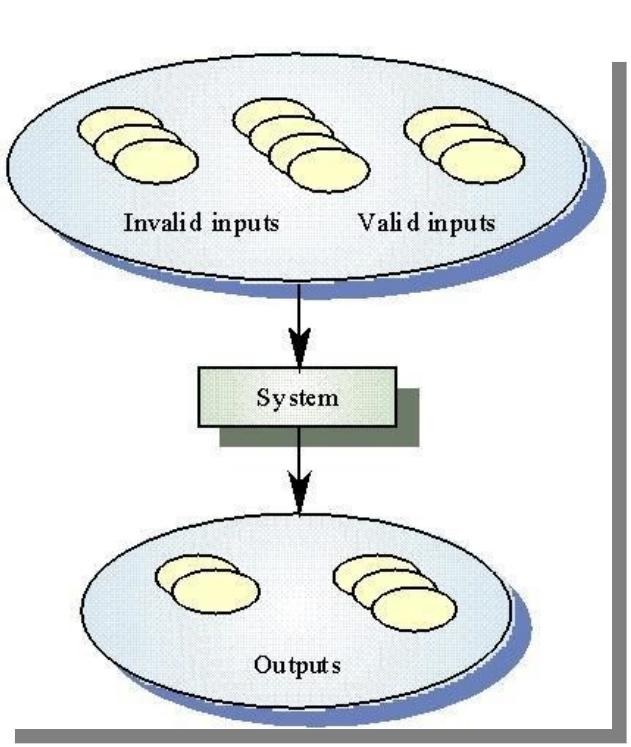
Equivalence Partitioning

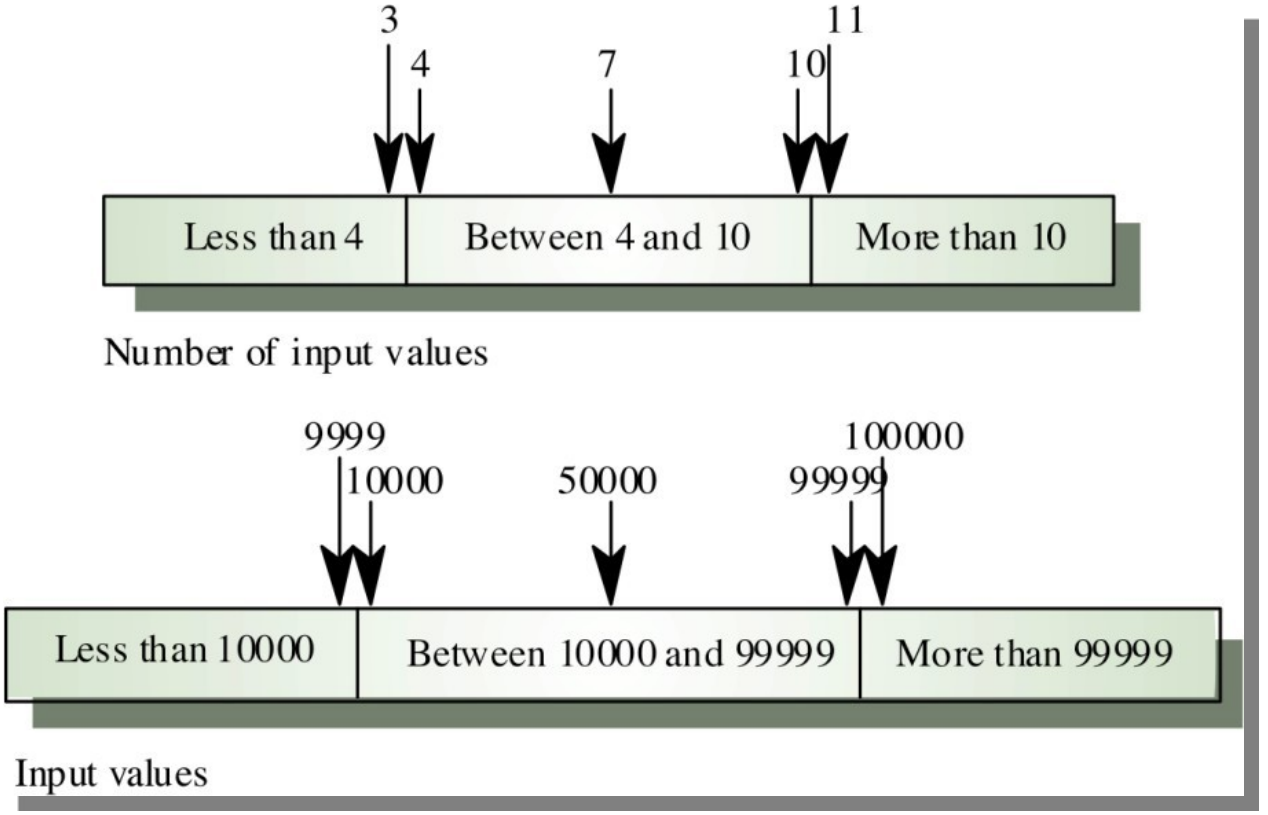
- Black-box technique that divides the input domain into classes of data from which test cases can be derived
- Input data and output results often fall into different classes where all members of a class are related
- An ideal test case uncovers a class of errors that might require many arbitrary test cases to be executed before a general error is observed
- Each of these classes is an equivalence partition where the program behaves in an equivalent way for each class member
- Test cases should be chosen from each partition

Equivalence Partitioning

- If input condition specifies a range, one valid and two invalid equivalence classes are defined
- If an input condition requires a specific value, one valid and two invalid equivalence classes are defined
- If an input condition specifies a member of a set, one valid and one invalid equivalence class is defined
- If an input condition is Boolean, one valid and one invalid equivalence class is defined
- If input is a 5-digit integer between 10,000 and 99,999, equivalence Partitions are <10,000, 10,000-99, 999 and > 10, 000
- Choose test cases at the boundary of these sets 00000, 09999, 10000, 99999, 10001

Equivalence Partitioning





Domain Testing

- Domain testing is a software testing technique in which selecting a small number of test cases from a nearly infinite group of test cases.
- For testing few applications, Domain specific knowledge plays a very crucial role.
- Domain testing is a type of functional testing and tests the application by feeding interesting inputs and evaluating its outputs
- Practitioners often study the simplest cases of domain testing less than two other names, "boundary testing" and "equivalence class analysis."

Domain Testing

- **Boundary testing** - Boundary value analysis (BVA) is based on testing at the boundaries between partitions. We will be testing both the valid and invalid input values in the partition/classes.
- **Equivalence Class testing** - The idea behind this technique is to divide (i.e. to partition) a set of test conditions into groups or sets that can be considered the same (i.e. the system should handle them equivalently), hence 'equivalence partitioning.'
- That simplified form applies for Domain testing –
- Only to tests of input variables
- Only when tested at the system level
- Only when tested one at a time
- Only when tested in a very superficial way

Domain Testing

Variable	Valid Class Equivalence Class	Invalid Class Equivalence Class	Boundaries & Special cases	Notes
X	0-100		0	
			100	
		<0	-1	
		>100	101	

- If a field accepts ranges from 0-100, the field should not accept -1 and 101 as they are invalid entries and beyond the boundaries.
- The field should accept values such as 0,100 and any number between them

Non Functional Testing

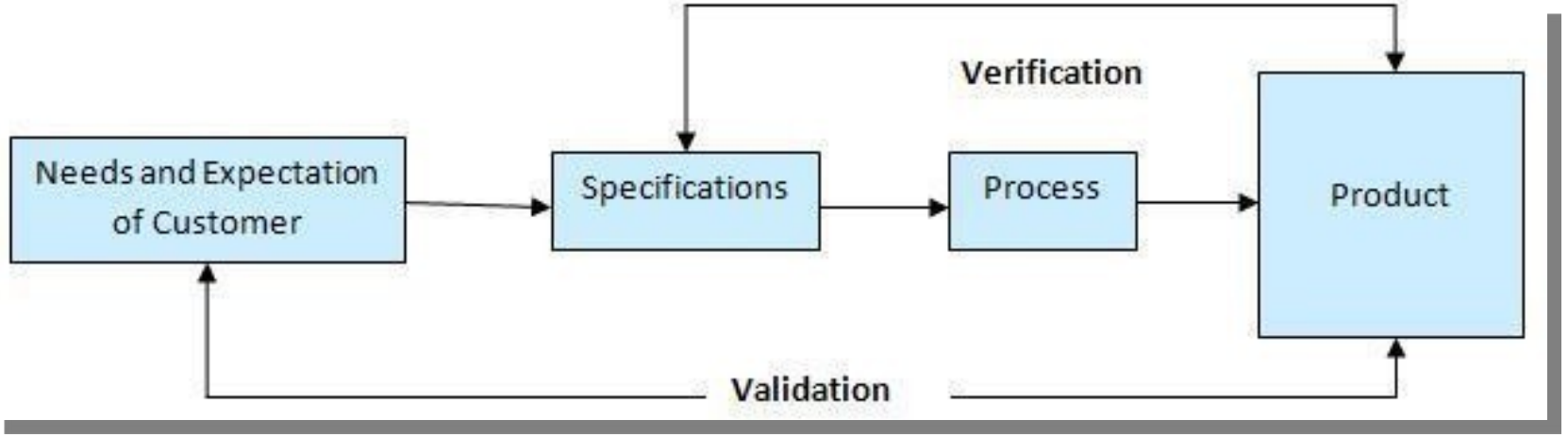
Reliability testing, **Usability** testing, **Efficiency** testing, **Maintainability** testing, **Portability** testing, **Baseline** testing, **Compliance** testing, **Documentation** testing, **Endurance** testing,**Load** testing, **Performance** testing, **Compatibility** testing, **Security** testing, **Scalability** testing, **Volume** testing, **Stress** testing, **Recovery** testing, **Internationalization** testing and **Localization** testing

Validation Testing

- Software validation is the process of testing software to check whether it satisfies the customer needs or not. This testing is done during and/or at the end of the process of software development.
- Verification and validation testing are two important tests which are carried out on software before it has been handed over to the customer.
- The aim of both verification and validation is to ensure that the product is made according to the requirements of the client, and does indeed fulfill the intended purpose.
- While verification is a quality control process, the quality assurance process carried out before the software is ready for release is known as validation testing.

Validation Testing

- Its goal is to validate and be confident about the product or system, and that it fulfills the requirements given by the customer.
- The acceptance of the software from the end customer is also its part. Often, testing activities are introduced early in the software development life cycle.



Validation Testing Activities

- **Validation Planning**- To plan all the activities that need to be included while testing.
- **Define Requirements**- To set goals and define the requirements for testing.
- **Select Appropriate Team**- To select a skilled and knowledgeable development team (third party included).
- **Develop Documents**- To develop a user specifications document describing the operating conditions.
- **Evaluation** - To evaluate the software as per the specifications and submit a validation report.
- **Incorporating Changes** - To change the software so as to remove any errors found during evaluation

High Level vs Low level Testing

- **High Level Test Cases**:-
  These test cases define the functionality of a software/product in a broader way without going into deep functionality. Like if we have to write high level test cases for login functionality we'll test that 'User should be able to login success full with valid credentials'.
- **Advantages**:-
  A tester is not bound to follow the test cases step by step and thus it gives a chance to explore more edge cases. This also increases the chance to find new bugs.
- **Disadvantages**: -
  It‘s not sure that all scenarios are covered and it‘s difficult for an inexperienced tester to work with these test cases.

High Level vs Low level Testing

- **Low Level Test cases**:-
  These test cases define the functionality of a software/product in deep way. These test cases generally include details like 'Excepted Result', 'Test Data',etc. Like if we have to write low level test case for login functionality then we'll describe UI of the login page (user name & password text boxes, Save, Forget password link), URL of the login page, and login with proper test data (valid credentials).
- **Advantages:-**
  Its advantages is that a tester is unlikely to miss bugs and also its easy for an inexperienced tester to work with these test cases as he can easily get the desired results by following steps.
- **Disadvantages:** - It‘s a tedious to execute these test cases again and again and tester may not find job challenges.

Enterprise Application Development