Testing Concepts

Lesson 2: Testing throughout the Software Development Life Cycle

Capgemini

Lesson Objectives

To understand the following topics :

- Software Development Lifecycle Models
  - Software Development and Software Testing
  - Software Development Lifecycle Models in Context
- Test Levels
  - Component Testing
  - Integration Testing
  - System Testing
  - Acceptance Testing

Lesson Objectives

- Test Types
  - Functional Testing
  - Non-functional Testing
  - White-box Testing
  - Change-related Testing
  - Test Types and Test Levels
- Maintenance Testing
  - Triggers for Maintenance
  - Impact Analysis for Maintenance
- Test Case Terminologies
- Test Data

2.1 Software Development Life Cycle (SDLC) Models

Testing is not a stand-alone activity

It has its place within a SDLC model

In any SDLC model, a part of testing is focused on Verification and a part is focused on Validation

- Verification: Is the deliverable built according to the specification?
- Validation: Is the deliverable fit for purpose?

2.1.1 Software Development & Software Testing

In any SDLC model, there are several characteristics of good testing:

- For every development activity, there is a corresponding test activity
- Each test level has test objectives specific to that level
- Test analysis and design for a given test level begin during the corresponding development activity
- Testers participate in discussions to define and refine requirements and design, and are involved in reviewing work products (e.g., requirements, design, user stories, etc.) as soon as drafts are available

No matter which SDLC model is chosen, test activities should start in the early stages of the lifecycle, adhering to the testing principle of early testing.

2.1.2 Software Development Lifecycle Models in Context

- SDLC models must be selected and adapted based on the context of project and product characteristics such as - project goal, the type of product being developed, business priorities (e.g., time-tomarket), and identified product and project risks.

Example : The development and testing of a minor internal administrative system should differ from the development and testing of a safety-critical system such as an automobile's brake control system.

Example : In some cases organizational and cultural issues may inhibit communication between team members, which can impede iterative development.

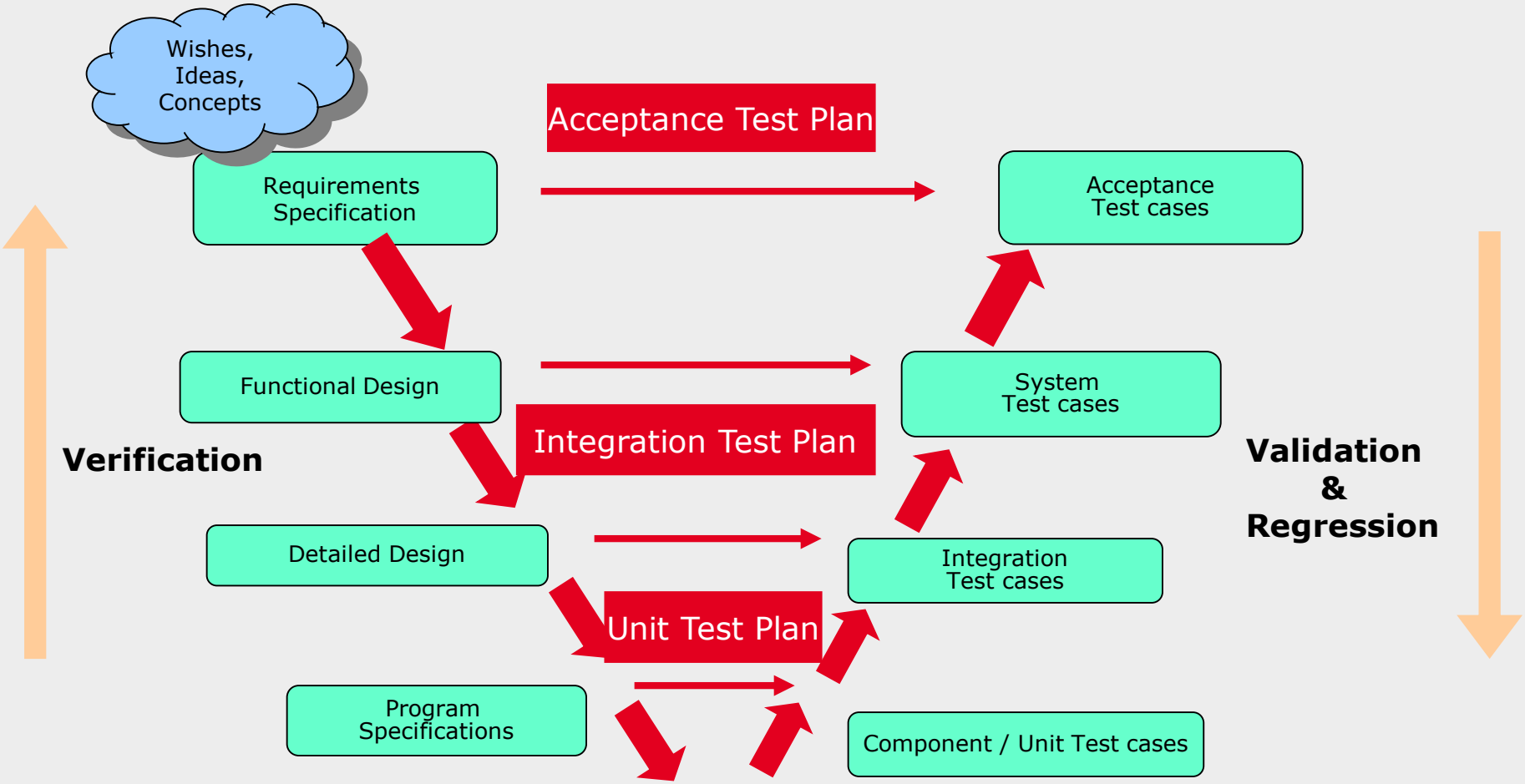
2.1.2 Software Development Lifecycle Models in Context (Cont.)

- Depending on the context of the project, it may be necessary to combine or reorganize test levels and/or test activities.

  Example : For the integration of a commercial off-the-shelf (COTS) software product into a larger system, the purchaser may perform interoperability testing at the system integration test level (e.g., integration to the infrastructure and other systems) and at the acceptance test level (functional and non-functional, along with user acceptance testing and operational acceptance testing).

2.2 Test Levels in V-Model

- Test levels are groups of test activities that are organized and managed together. Each test level is an instance of the test process related to other activities with in SDLC.



Verification and Validation

Verification

- Verification refers to a set of activities which ensures that software correctly implements a specific function.
- Purpose of verification is to check: Are we building the product right?
- Example: code and document reviews, inspections, walkthroughs.
- It is a Quality improvement process.
- It is involve with the reviewing and evaluating the process.
- It is conducted by QA team.
- Verification is Correctness.

Verification and Validation (Cont.)

Validation

- Validation is the following process of verification.
- Purpose of Validation is to check : Are we building the right product?
- Validation refers to a different set of activities which ensures that the software that has been built is traceable to customer requirements.
- After each validation test has been conducted, one of two possible conditions exist:
  1. The function or performance characteristics conform to specification and are accepted, or
  2. Deviation from specification and a deficiency list is created.
- It is conducted by development team with the help from QC team.

2.2 Test Levels (Cont.)

Unit (Component) testing

- Unit testing is code-based and performed primarily by developers to demonstrate that their smallest pieces of executable code function suitably.

Integration testing

- Integration testing demonstrates that two or more units or other integrations work together properly, and tends to focus on the interfaces specified in low-level design.

System testing

- System testing demonstrates that the system works end-to-end in a production-like environment to provide the business functions specified in the high-level design.

Acceptance testing

- Acceptance testing is conducted by business owners and users to confirm that the system does, in fact, meet their business requirements.

2.2 Test Levels (Cont.)

Test levels are characterized by the following attributes:

- Specific objectives
- Test basis, referenced to derive test cases
- Test object (i.e., what is being tested)
- Typical defects and failures
- Specific approaches and responsibilities.

For every test level, a suitable test environment is required.

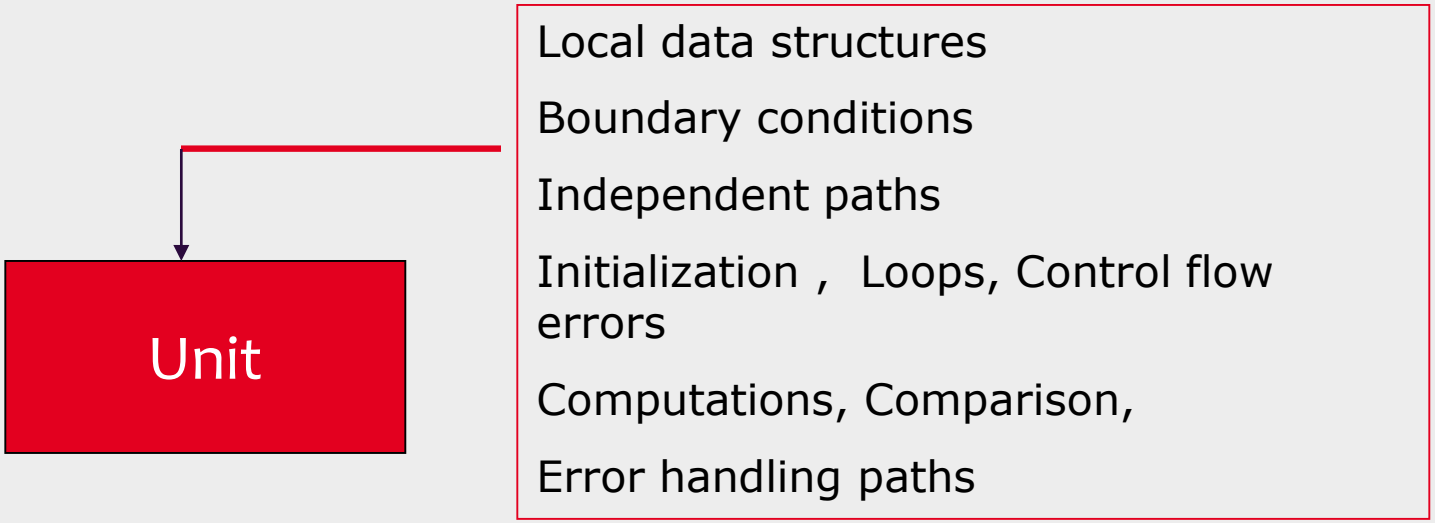
Example : In acceptance testing, for example, a production-like test environment is ideal, while in component testing the developers typically use their own development environment.

2.2.1 Component Testing

- The most 'micro' scale of testing to test particular functions, procedures or code modules or components is called Component testing ; Also called as Module or Unit testing
- Typically done by the programmer and not by Test Engineers, as it requires detailed knowledge of the internal program design and code.
- Objectives of component testing include:
  - Reducing risk
  - Verifying whether the functional and non-functional behaviors of the component are as designed and specified
  - Building confidence in the component’s quality
  - Finding defects in the component
  - Preventing defects from escaping to higher test levels

Component /Unit Testing

Unit testing uncovers errors in logic and function within the boundaries of a component.



Component Testing

Test Basis : Test basis for component testing include:

- Detailed design
- Code
- Data model
- Component specifications

Test objects : Typical test objects for component testing include:

- Components, units or modules
- Code and data structures
- Classes
- Database modules

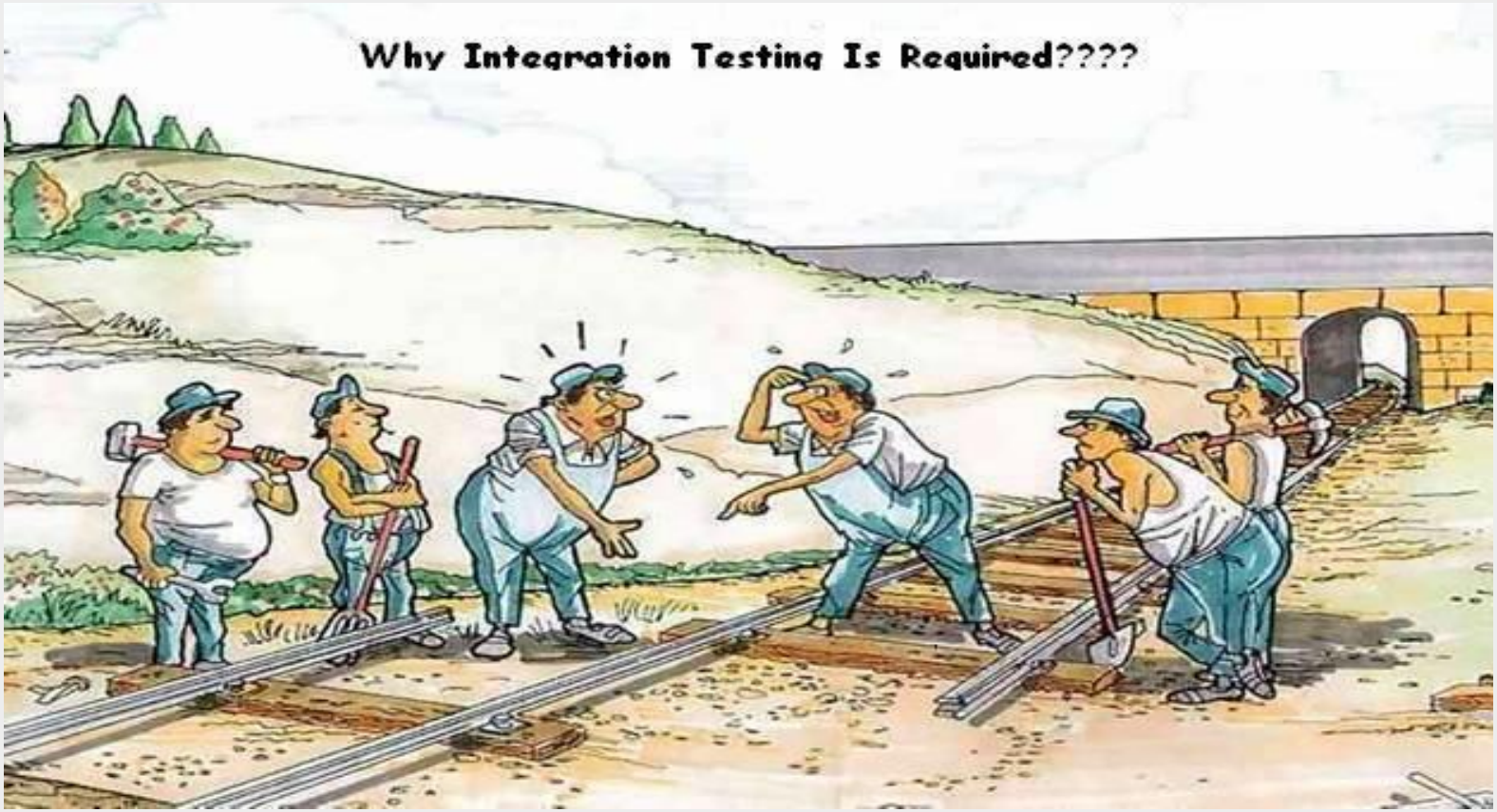
Typical defects and failures : Typical defects and failures for component testing include :

- Incorrect functionality (e.g., not as described in design specifications)
- Data flow problems
- Incorrect code and logic

2.2.2 Integration testing

- Integration testing focuses on interactions between components or systems.
- Objectives of Integration testing include:
  - Reducing risk
  - Verifying whether the functional and non-functional behaviors of the interfaces are as designed and specified
  - Building confidence in the quality of the interfaces.
  - Finding defects (which may be in the interfaces themselves or within the components or systems)
  - Preventing defects from escaping to higher test levels

Why Integration Testing is Required?



Two Levels of Integration testing

There are two different levels of integration testing which may be carried out on test objects of varying size as follows:

- **Component integration testing (CIT)** focuses on the interactions and interfaces between integrated components. Component integration testing is performed after component testing.

- **System integration testing (SIT)** focuses on the interactions and interfaces between systems, packages, and micro services. System integration testing may be done after system testing.

Integration Testing

Test Basis : Test basis for Integration testing include:

- Software and system design
- Sequence diagrams
- Interface and communication protocol specifications
- Use cases
- Architecture at component or system level
- Workflows
- External interface definitions

Test objects : Typical test objects for Integration testing include:

- Subsystems
- Databases
- Infrastructure
- Interfaces
- APIs
- Micro services

Integration Testing

Typical defects and failures for CIT include :

- Incorrect data, missing data, or incorrect data encoding
- Incorrect sequencing or timing of interface calls
- Interface mismatch
- Failures in communication between components
- Unhandled or improperly handled communication failures between components
- Incorrect assumptions about the meaning, units, or boundaries of the data being passed between components

Integration Testing

Typical defects and failures for SIT include:

- Inconsistent message structures between systems
- Incorrect data, missing data, or incorrect data encoding
- Interface mismatch
- Failures in communication between systems
- Unhandled or improperly handled communication failures between systems
- Incorrect assumptions about the meaning, units, or boundaries of the data being passed between systems
- Failure to comply with mandatory security regulations

Types of Integration testing

Modules are integrated by two ways.

1. **Non-incremental Testing (Big Bang Testing)**
   - Each Module is tested independently and at the end, all modules are combined to form a application
1. **Incremental Module Testing.**
   - There are two types by which incremental module testing is achieved.
     - **Top down Approach :** Firstly top module is tested first. Once testing of top module is done then any one of the next level modules is added and tested. This continues till last module at lowest level is tested and it is called as Stub.
     - **Bottom up Approach :** Firstly module at the lowest level is tested first. Once testing of that module is done then any one of the next level modules is added to it and tested. This continues till top most module is added to rest all and tested and it is called as Driver.

2.2.3 System Testing

- System testing focuses on the behavior and capabilities of a whole system or product, often considering the end-to-end tasks the system can perform and the non-functional behaviors it exhibits while performing those tasks.
- Test the software in the real environment in which it is to operate. (hardware, people, information, etc.)
- Observe how the system performs in its target environment, for example in terms of speed, with volumes of data, many users, all making multiple requests.
- Test how secure the system is and how can the system recover if some fault is encountered in the middle of procession.
- System Testing, by definition, is impossible if the project has not produced a written set of measurable objectives for its product.

System testing

- Objectives of System testing include:
  - Reducing risk
  - Verifying whether the functional and non-functional behaviors of the system are as designed and specified
  - Validating that the system is complete and will work as expected
  - Building confidence in the quality of the system as a whole
  - Finding defects
  - Preventing defects from escaping to higher test levels or production

System Testing

Test Basis : Test basis for System testing include:

- System and software requirement specifications (functional and non-functional)
- Risk analysis reports
- Use cases
- Epics and user stories
- Models of system behavior
- State diagrams
- System and user manuals

Test objects : Typical test objects for System testing include:

- Applications
- Hardware/software systems
- Operating systems
- System under test (SUT)
- System configuration and configuration data

System Testing

Typical defects and failures :

- Typical defects and failures for System Testing include :
  - Incorrect calculations
  - Incorrect or unexpected system functional or non-functional behavior
  - Incorrect control and/or data flows within the system
  - Failure to properly and completely carry out end-to-end functional tasks
  - Failure of the system to work properly in the production environment(s)
  - Failure of the system to work as described in system and user manuals

Types of System Testing

- Functional Testing
- Performance Testing
- Volume Testing
- Load Testing
- Stress Testing
- Security Testing
- Web Security Testing
- Localization Testing
- Usability Testing
- Recovery Testing
- Documentation Testing
- Configuration Testing
- Installation Testing
- User Acceptance Testing
- Testing related to Changes : Re-Testing and Regression Testing
- Re-testing (Confirmation Testing)
- Regression Testing
- Exploratory Testing
- Maintenance Testing

Functional Testing

The main objective of functional testing is to verify that each function of the software application / system operates in accordance with the written requirement specifications.

It is a black-box process :

- Is not concerned about the actual code
- Focus is on validating features
- Uses external interfaces, including Application programming interfaces (APIs), Graphical user interfaces (GUIs) and Command line interfaces (CLIs)

Testing functionality can be done from two perspectives :

1. Business-process-based testing uses knowledge of the business processes
2. Requirements-based testing uses a specification of the functional requirements for the system as the basis for designing tests

Performance Testing

Performance

- Performance is the behavior of the system w.r.t. goals for time, space, cost and reliability

Performance objectives:

- **Throughput :** The number of tasks completed per unit time. Indicates how much work has been done within an interval
- **Response time :** The time elapsed during input arrival and output delivery
- **Utilization :** The percentage of time a component (CPU, Channel, storage, file server) is busy

Volume Testing

This testing is subjecting the program to heavy volumes of data. For e.g.

- A compiler would be fed a large source program to compile
- An operating systems job queue would be filled to full capacity
- A file system would be fed with enough data to cause the program to switch from one volume to another.

Load Testing

Volume testing creates a real-life end user pressure for the target software. This tests how the software acts when numerous end users access it concurrently. For e.g.

- Downloading a sequence of huge files from the web
- Giving lots of work to a printer in a line

Stress Testing

- Stress testing involves subjecting the program to heavy loads or stresses.
- The idea is to try to “break” the system.
- That is, we want to see what happens when the system is pushed beyond design limits.
- It is not same as volume testing.
- A heavy stress is a peak volume of data encounters over a short time.
- In Stress testing a considerable load is generated as quickly as possible in order to stress the application and analyze the maximum limit of concurrent users the application can support.

Stress Testing(Cont.)

Stress tests executes a system in a manner that demands resources in abnormal quantity, frequency, or volume

Example :

- Generate 5 interrupts when the average rate is 2 or 3
- Increase input data rate
- Test cases that require max. memory

Stress Tests should answer the following questions

- Does the system degrade gently or does the server shut down
- Are appropriate messages displayed ? E.g. Server not available
- Are transactions lost as capacity is exceeded
- Are certain functions discontinued as capacity reaches the 80 or 90 percent level

Security Testing

Security Testing verifies that protection mechanisms built into the system will protect it from improper penetration.

Security testing is the process of executing test cases that subvert the program's security checks.

Example :

- One tries to break the operating systems memory protection mechanisms
- One tries to subvert the DBMS's data security mechanisms
- The role of the developer is to make penetration cost more than the value of the information that will be obtained

Web Security Testing

Web application security is a branch of Information Security that deals specifically with security of web applications.

It provides a strategic approach in identifying, analyzing and building a secure web applications.

It is performed by Web Application Security Assessment.

Localization Testing

Localization translates the product UI and occasionally changes some settings to make it suitable for another region.

The test effort during localization testing focuses on

- Areas affected during localization, UI and content
- Culture/locale-specific, language specific and region specific areas

Usability Testing

Usability is

- The effectiveness, efficiency and satisfaction with which specified users can achieve specified goals in a particular environment ISO 9241-11
- Effective–- Accomplishes user’s goal
- Efficient-- Accomplishes the goal quickly
- Satisfaction–- User enjoys the experience

Test Categories and objectives

- Interactivity (Pull down menus, buttons)
- Layout
- Readability
- Aesthetics
- Display characteristics
- Time sensitivity
- Personalization

Usability Testing (Cont.)

Using specialized Test Labs a rigorous testing process is conducted to get quantitative and qualitative data on the effectiveness of user interfaces

Representative or actual users are asked to perform several key tasks under close observation, both by live observers and through video recording

During and at the end of the session, users evaluate the product based on their experiences

Recovery Testing

A system test that forces the software to fail in variety of ways, checks performed

- recovery is automatic (performed by the system itself)
- reinitialization
- check pointing mechanisms
- data recovery
- restarts are evaluated for correctness

This test confirms that the program recovers from expected or unexpected events. Events can include shortage of disk space, unexpected loss of communication

Documentation Testing

This testing is done to ensure the validity and usability of the documentation

This includes user Manuals, Help Screens, Installation and Release Notes

Purpose is to find out whether documentation matches the product and vice versa

Well-tested manual helps to train users and support staff faster

Configuration Testing

Attempts to uncover errors that are specific to a particular client or server environment.

Create a cross reference matrix defining all probable operating systems, browsers, hardware platforms and communication protocols.

Test to uncover errors associated with each possible configuration

Installation Testing

Installer is the first contact a user has with a new software!!!

Installation testing is required to ensure:

- Application is getting installed properly
- New program that is installed is working as desired
- Old programs are not hampered
- System stability is maintained
- System integrity is not compromised

2.2.4 Acceptance testing

- Acceptance testing, like system testing, typically focuses on the behavior and capabilities of a whole system or product
- Objectives of Acceptance testing include:
  - Establishing confidence in the quality of the system as a whole
  - Validating that the system is complete and will work as expected
  - Verifying that functional and non-functional behaviors of the system are as specified

Forms of Acceptance testing

Common forms of acceptance testing include the following:

- User Acceptance testing (UAT)
- Operational Acceptance testing (OAT)
- Contractual and Regulatory Acceptance testing
- Alpha and Beta testing.

User Acceptance Testing (UAT)

- A test executed by the end user(s) is typically focused on validating the fitness for use of the system by intended users in a real or simulated operational environment. The main objective is building confidence that the users can use the system to meet their needs, fulfill requirements, and perform business processes with minimum difficulty, cost, and risk.

- Usually carried out by the end user to test whether or not the right system has been created

Operational Acceptance Testing (OAT)

- Acceptance testing of the system by operations or systems administration staff is usually performed in a (simulated) production environment.
- The tests focus on operational aspects, and may include:
  - Testing of backup and restore
  - Installing, uninstalling and upgrading
  - Disaster recovery
  - User management
  - Maintenance tasks
  - Data load and migration tasks
  - Checks for security vulnerabilities
  - Performance testing

Contractual and Regulatory Testing

- **Contractual** acceptance testing is performed against a contract's acceptance criteria for producing custom-developed software. Acceptance criteria should be defined when the parties agree to the contract. Contractual acceptance testing is often performed by users or by independent testers.

- **Regulatory** acceptance testing is performed against any regulations that must be adhered to, such as government, legal, or safety regulations. Regulatory acceptance testing is often performed by users or by independent testers, sometimes with the results being witnessed or audited by regulatory agencies.

Alpha and Beta Testing

- Alpha and beta testing are typically used by developers of commercial off-the-shelf (COTS) software who want to get feedback from potential or existing users, customers, and/or operators before the software product is put on the market.
- Alpha testing is performed at the developing organization’s site, not by the development team, but by potential or existing customers, and/or operators or an independent test team.
- Beta testing is performed by potential or existing customers, and/or operators at their own locations. Beta testing may come after alpha testing, or may occur without any preceding alpha testing having occurred.

Alpha and Beta Testing

Test Basis

Examples of work products that can be used as a test basis for any form of acceptance testing include:

- Business processes
- User or business requirements
- Regulations, legal contracts and standards
- Use cases
- System requirements
- System or user documentation
- Installation procedures
- Risk analysis reports

Alpha and Beta Testing

Test Objects

Typical test objects for any form of acceptance testing include:

- System under test
- System configuration and configuration data
- Business processes for a fully integrated system
- Recovery systems and hot sites (for business continuity and disaster recovery testing)
- Operational and maintenance processes
- Forms
- Reports
- Existing and converted production data

Alpha and Beta Testing

Typical Defects and Failures

Typical defects for any form of acceptance testing include:

- System workflows do not meet business or user requirements
- Business rules are not implemented correctly
- System does not satisfy contractual or regulatory requirements
- Non-functional failures such as security vulnerabilities, inadequate performance efficiency under high loads, or improper operation on a supported platform

2.3 Test Types

A test type is a group of test activities aimed at testing specific characteristics of a software system based on specific test objectives such as :

- Evaluating functional quality characteristics, such as completeness, correctness, and appropriateness
- Evaluating non-functional quality characteristics, such as reliability, performance efficiency, security, compatibility, and usability
- Evaluating whether the structure or architecture of the component or system is correct, complete, and as specified
- Evaluating the effects of changes, such as confirming that defects have been fixed (confirmation testing) and looking for unintended changes in behavior resulting from software or environment changes (regression testing)

2.3.1 Functional Testing

- Functional testing of a system involves tests that evaluate functional requirements such as business requirements specifications, epics, user stories, use cases, or functional specifications, or they may be undocumented.
- The functions are “what” the system should do
- Functional tests should be performed at all test levels though the focus is different at each level.
- Black-box techniques may be used to derive test conditions and test cases for the functionality of the component or system

2.3.2 Non-Functional Testing

- Non-functional testing of a system evaluates characteristics of systems and software such as usability, performance efficiency or security.
- Non-functional testing is the testing of “how well” the system behaves.
- non-functional testing can and often should be performed at all test levels, and done as early as possible.
- Black-box techniques may be used to derive test conditions and test cases even for nonfunctional testing too. For example, boundary value analysis can be used to define the stress conditions for performance tests.

2.3.3 White-box Testing

- White-box testing derives tests based on the system’s internal structure or implementation. Internal structure may include code, architecture, work flows, and/or data flows within the system.

- White-box testing derives tests based on the system’s internal structure or implementation. Internal structure may include code, architecture, work flows, and/or data flows within the system.

2.3.4 Change-related Testing

- When changes are made to a system, either to correct a defect or because of new or changing functionality, testing should be done to confirm that the changes have corrected the defect or implemented the functionality correctly, and have not caused any unforeseen adverse consequences.
  - Confirmation testing
  - Regression testing
- Confirmation testing and regression testing are performed at all test levels.
  - Especially in iterative and incremental development lifecycles (e.g., Agile), new features, changes to existing features, and code refactoring results in frequent changes to the code, which requires change-related testing.
  - Due to the evolving nature of the system (e.g. IoT system), confirmation and regression testing are very important.

Re-testing (Confirmation Testing)

- After a defect is fixed, the software is tested by re-running all test cases that failed due to the defect, within same environment, with same inputs and same preconditions on the new software version.
- The software may also be tested with new tests if, for instance, the defect was missing functionality.
- At the very least, the steps to reproduce the failure(s) caused by the defect must be re-executed on the new software version.
- The purpose of a confirmation test is to confirm whether the original defect has been successfully fixed

Regression Testing

- It is possible that a change made in one part of the code, whether a fix or another type of change, may accidentally affect the behavior of other parts of the code, whether within the same component, in other components of the same system, or even in other systems.
- Changes may include changes to the environment, such as a new version of an operating system or database management system.
- Such unintended side-effects are called regressions.
- Regression testing involves running tests to detect such unintended side-effects

2.3.5 Test Types and Test Levels
Example: Banking Application

It is possible to perform any of the test types at any test level.

Examples of Functional Tests :

- For CT, tests are designed based on how a component should calculate compound interest.
- For CIT, tests are designed based on how account information captured at the user interface is passed to the business logic.
- For ST, tests are designed based on how account holders can apply for a line of credit on their checking accounts.
- For SIT, tests are designed based on how the system uses an external micro service to check an account holder’s credit score.
- For UAT, tests are designed based on how the banker handles approving or declining a credit application.

2.3.5 Test Types and Test Levels (Cont..)

Example: Banking Application

Examples of Non-Functional Tests :

- For CT, performance tests are designed to evaluate the number of CPU cycles required to perform a complex total interest calculation
- For CIT, security tests are designed for buffer overflow vulnerabilities due to data passed from the user interface to the business logic.
- For ST, portability tests are designed to check whether the presentation layer works on all supported browsers and mobile devices.
- For SIT, reliability tests are designed to evaluate system robustness if the credit score micro service fails to respond.
- For UAT, usability tests are designed to evaluate the accessibility of the banker’s credit processing interface for people with disabilities.

2.3.5 Test Types and Test Levels (Cont..)

Example: Banking Application

Examples of White-box Tests :

- For CT, performance tests are designed to achieve complete statement and decision coverage for all components that perform financial calculations.
- For CIT, security tests are designed to exercise how each screen in the browser interface passes data to the next screen and to the business logic.
- For ST, tests are designed to cover sequences of web pages that can occur during a credit line application.
- For SIT, tests are designed to exercise all possible inquiry types sent to the credit score microservice.
- For UAT, tests are designed to cover all supported financial data file structures and value ranges for bank-to-bank transfers.

2.3.5 Test Types and Test Levels (Cont..)

Example: Banking Application

Examples of change-related Tests :

- For CT, automated regression tests are built for each component and included within the continuous integration framework.
- For CIT, tests are designed to confirm fixes to interface-related defects as the fixes are checked into the code repository.
- For ST, all tests for a given workflow are re-executed if any screen on that workflow changes.
- For SIT, tests of the application interacting with the credit scoring micro service are re-executed daily as part of continuous deployment of that micro service.
- For UAT, all previously-failed tests are re-executed after a defect found in acceptance testing is fixed.

2.4 Maintenance Testing

- Once deployed to production environments, software and systems need to be maintained.

- Changes of various sorts are almost inevitable in delivered software and systems, either to fix defects discovered in operational use, to add new functionality, or to delete or alter already-delivered functionality.

- Maintenance is also needed to preserve or improve non-functional quality characteristics of the component or system over its lifetime, especially performance efficiency, compatibility, reliability, security, compatibility, and portability.

2.4 Maintenance Testing (cont..)

- When any changes are made as part of maintenance, maintenance testing should be performed, both to evaluate the success with which the changes were made and to check for possible side-effects (e.g., regressions) in parts of the system that remain unchanged.
- Maintenance can involve planned releases and unplanned releases (hot fixes).
- Maintenance testing is required at multiple test levels, using various test types, based on its scope. The scope of maintenance testing depends on:
  - The degree of risk of the change, for example, the degree to which the changed area of software communicates with other components or systems
  - The size of the existing system
  - The size of the change

2.4.1 Triggers for Maintenance

There are several reasons (triggers) why maintenance testing takes place, both for planned and unplanned changes :

- Modification, such as planned enhancements (e.g., release-based), corrective and emergency changes, changes of the operational environment (such as planned operating system or database upgrades), upgrades of COTS software, and patches for defects and vulnerabilities
- Migration, such as from one platform to another, which can require operational tests of the new environment as well as of the changed software, or tests of data conversion when data from another application will be migrated into the system being maintained
- Retirement, such as when an application reaches the end of its life

2.4.2 Impact Analysis for Maintenance

Impact analysis may be done before a change is made, to help decide if the change should be made, based on the potential consequences in other areas of the system.

Impact analysis can be difficult if:

- Specifications (e.g., business requirements, user stories, architecture) are out of date or missing
- Test cases are not documented or are out of date
- Bi-directional traceability between tests and the test basis has not been maintained
- Tool support is weak or non-existent
- The people involved do not have domain and/or system knowledge
- Insufficient attention has been paid to the software's maintainability during development

2.5 Test Case Terminologies

Pre Condition

- Environmental and state which must be fulfilled before the component/unit can be executed with a particular input value.

Test Analysis

- is a process for deriving test information by viewing the Test Basis
- For testing, test basis is used to derive what could be tested

Test basis includes whatever the test are based on such as System Requirement

- A Technical specification
- The code itself (for structural testing)
- A business process

Test Condition

- It is a set of rules under which a tester will determine if a requirement is partially or fully satisfied
- One test condition will have multiple test cases

Test Case Terminologies (cont.)

Test Scenario

- It is an end-to-end flow of a combination of test conditions & test cases integrated in a logical sequence, covering a business processes
- This clearly states what needs to be tested
- One test condition will have multiple test cases

Test Procedure (Test Steps)

- A detailed description of steps to execute the test

Test Data/Input

- Inputs & its combinations/variables used

Expected Output

- This is the expected output for any test case or any scenario

Actual Output

- This is the actual result which occurs after executing the test case

Test Result/Status

- Pass / Fail – If the program works as given in the specification, it is said to Pass otherwise Fail.
- Failed test cases may lead to code rework

Other Terminologies

Test Suite – A set of individual test cases/scenarios that are executed as a package, in a particular sequence and to test a particular aspect

- E.g. Test Suite for a GUI or Test Suite for functionality

Test Cycle – A test cycle consists of a series of test suites which comprises a complete execution set from the initial setup to the test environment through reporting and clean up.

- E.g. Integration test cycle / regression test cycle

A good Test Case

Has a high probability of detecting error(s)
Test cases help us discover information
Maximize bug count
Help managers make ship / no-ship decisions
Minimize technical support costs
Assess conformance to specification
Verify correctness of the product
Minimize safety-related lawsuit risk
Find safe scenarios for use of the product
Assure quality

2.6 Test data

- An application is built for a business purpose. We input data and there is a corresponding output. While an application is being tested we need to use dummy data to simulate the business workflows. This is called test data.

- The test data may be any kind of input to application, any kind of file that is loaded by the application or entries read from the database tables. It may be in any format like xml test data, stand alone variables, SQL test data etc.

Properties of Good Test Data

Realistic – accurate in context of real life

- E.g. Age of a student giving graduation exam is at least 18

Practically valid – data related to business logic

- E.g. Age of a student giving graduation exam is at least 18 says that 60 years is also valid input but practically the age of a graduate student cannot be 60

Cover varied scenarios

- E.g. Don’t just consider the scenario of only regular students but also consider the irregular students, also the students who are giving a re-attempt, etc.

Exceptional data

- E.g. There may be few students who are physically handicapped must also be considered for attempting the exam

Summary

In this lesson, you have learnt:

- Verification refers to a set of activities which ensures that software correctly implements a specific function.
- Validation refers to a different set of activities which ensures that the software that has been built is traceable to customer requirements.
- Different testing phases are
  - Unit testing
  - Integration testing
  - System testing
  - Acceptance testing

Review Question

Question 1: _____ is a Quality improvement process

Question 2: To test a function, the programmer has to write a _____, which calls the function to be tested and passes it test data

Question 3: Volume tests executes a system in a manner that demands resources in abnormal quantity, frequency, or volume

- True/False

Question 4: Acceptance testing is not a responsibility of the Developing Organization

- True/False

Question 5: Difference between re-testing & regression testing is :

- re-testing is running a test again; regression testing looks for unexpected side effects
- re-testing looks for unexpected side effects; regression testing is repeating

Review Question: Match the Following

1. Beta testing	A. Volume testing
2. Response time	B. Exploratory testing
3. Aesthetics	C. Acceptance testing
	D. Documentation testing
	E. Performance testing
	F. Usability testing

Review Question: Match the Following

1. Economics of limiting	A. Driving
2. Testing	B. Exhaustive testing
3. A good test case	C. Limiting
4. Use every possible input condition as a test case	D. Test cycle
	E. Comparing outputs with specified or intended
	F. Maximize bug count