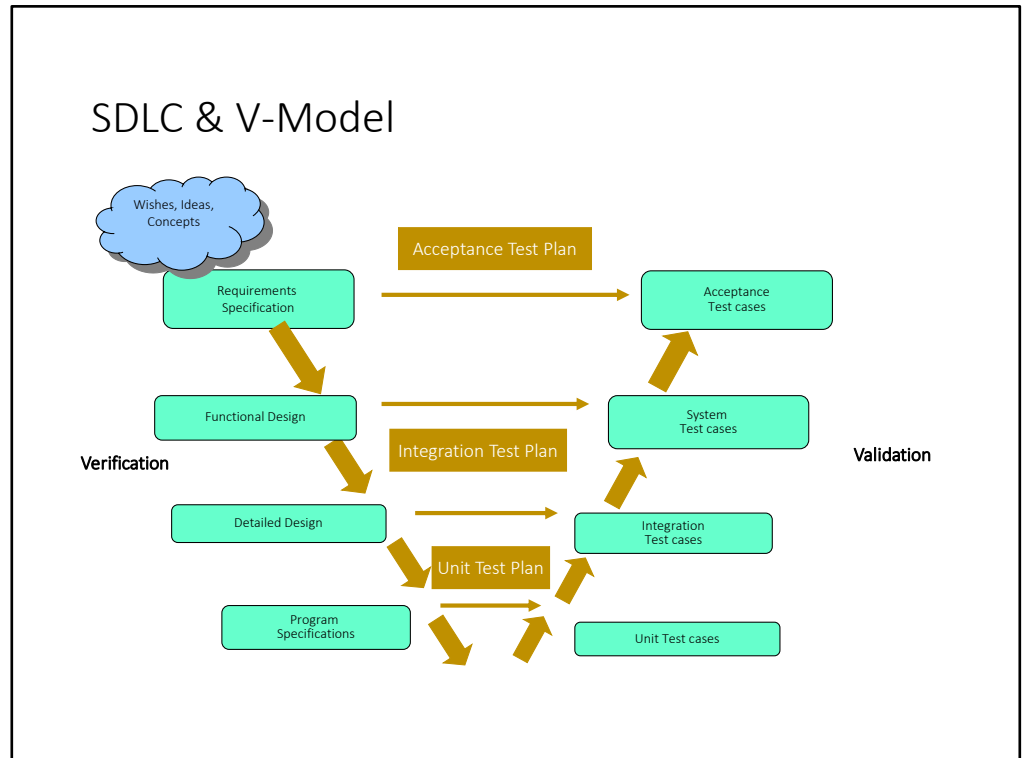


## Testing Concepts

### Lesson 3: Testing throughout the Software Life Cycle

## Lesson Objectives

- To understand the following topics:
  - SDLC and V-Model
  - Testing Phases
  - Introduction of Component Testing
  - Component /Unit Testing
  - Introduction of Integration testing
  - Why Integration Testing is Required?
  - Types of Integration testing
  - Top Down Integration Testing
  - Bottom Up Integration Testing
  - Top Down vs. Bottom Up Testing
  - Introduction to System Testing
  - Types of System Testing



#### V Model

The V proceeds from left to right, depicting the basic sequence of development and testing activities

Left side shows activities apart from testing

Right side shows Testing activities

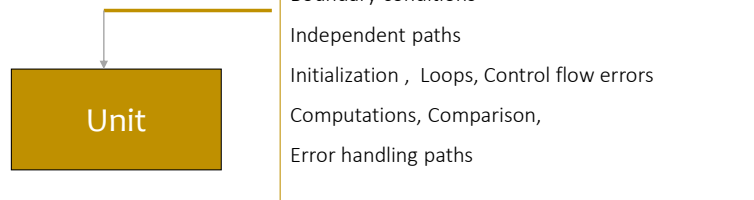
Specific Testing activities are carried in parallel to development activities

## Testing Phases

- **Unit (Component) testing**
  - Unit testing is code-based and performed primarily by developers to demonstrate that their smallest pieces of executable code function suitably. Testing a form, a class or a stored procedure can be an example of unit testing
- **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.

## Component /Unit Testing

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



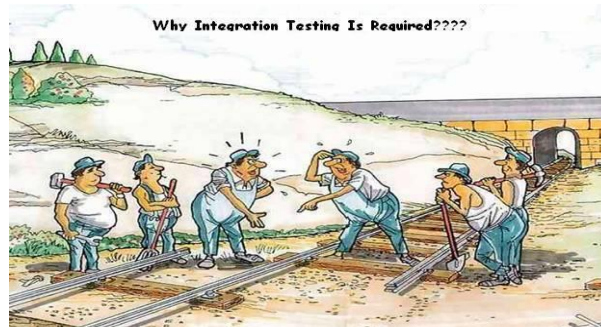
The module interface is tested to ensure that information properly flows into and out of the program under test. Local data structures are examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution.

Test cases should uncover errors such as

- (1) comparison of different data types
- (2) incorrect logical operators or precedence
- (3) expectation of equality when precision error makes equality unlikely
- (4) incorrect comparison of variables
- (5) improper or nonexistent loop termination
- (6) failure to exit when divergent iteration is encountered.

## Introduction of Integration testing

- Testing of combined parts of an application to determine if they function together correctly
- The main three elements are interfaces, module combinations and global data structures
- Attempts to find discrepancies between program & its external specification (program's description from the point of view of the outside world)
- Testing a module to check if the component of the modules are integrated properly is example of integration testing



Interfaces are the means by which data is passed to and from modules. "Interface integrity" – to ensure that when data is passed to another module, by way of a call, none of the data becomes lost or corrupted. This loss or corruption can happen by number of ways- calling and receiving parameters may be of the wrong type and so the data appears in the receiving programs in a garbled form; there may be different number of calling and receiving parameters and so the data is lost; arrays may be of different lengths.

Global data structures are those pieces of data, maybe in the form of files, databases or variables, that are existing through out the entire system. Every module has access to global data structures and may alter the contents thereof. The effect of this again may create some unusual combination of data which reveals an error in another module.

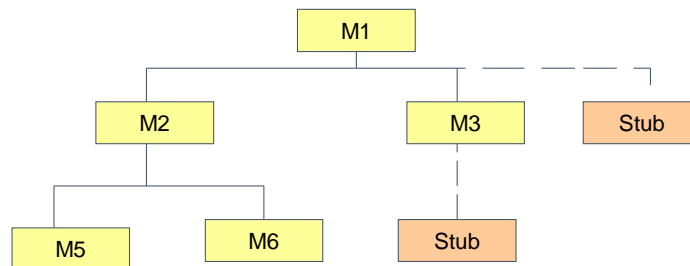
## Types of Integration testing

- Modules are integrated by two ways.
  - Non-incremental Testing (Big Bang Testing)
    - Each Module is tested independently and at the end, all modules are combined to form a application
  - Incremental Module Testing.
    - Top down Approach
    - Bottom up Approach

## Top Down Integration Testing

- Top Down Incremental Module Integration:

- 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.



The main control module is used as a test driver

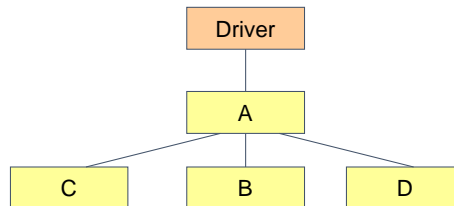
Stubs are substituted for all components directly subordinate to the main control module

Depending on the approach subordinate stubs are replaced by actual components



## Bottom Up Integration Testing

- Bottom Up Incremental Module Integration:
  - 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



Low-level components are combined into clusters (builds) that perform a specific sub function

A driver is written to coordinate test case input and output

Drivers are removed and clusters are combined moving upward in the program structure

## Top Down vs. Bottom Up Testing

Top Down Testing	
Advantages	Disadvantages
Advantageous if major flaws occur toward the top of the program	Stub modules must be produced
Once the I/O functions are added, representation of test cases are easier	Stub Modules are often more complicated than they first appear to be.
Early skeletal Program allows demonstrations and boosts morale	Before the I/O functions are added, representation of test cases in stubs can be difficult
	Test conditions may be impossible, or very difficult, to create
	Observation of test output is more difficult
	Allows one to think that design and testing can be overlapped
	Induces one to defer completion of the testing of certain modules.

## Top Down vs. Bottom Up Testing

Bottom Up testing	
Advantages	Disadvantages
Advantageous if major flaws occur toward the bottom of the program	Driver Modules must be produced
Test conditions are easier to create	The program as an entity does not exist until the last module is added
Observation of test results is easier	

## 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

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.

## 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 :
  - Business-process-based testing uses knowledge of the business processes
  - 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

The characteristics and their sub-characteristics are, respectively:

- **functionality**, which consists of five sub-characteristics: suitability, accuracy, security, interoperability and compliance; this characteristic deals with functional testing as described in Section 2.3.1;
- **reliability**, which is defined further into the sub-characteristics maturity (robustness), fault-tolerance, recoverability and compliance;
- **usability**, which is divided into the sub-characteristics understandability, learnability, operability, attractiveness and compliance;
- **efficiency**, which is divided into time behavior (performance), resource utilization and compliance;
- **maintainability**, which consists of five sub-characteristics: analyzability, changeability, stability, testability and compliance;
- **portability**, which also consists of five sub-characteristics: adaptability, installability, co-existence, replaceability and compliance

## Performance Testing (Cont.)

- The objective of performance testing is to devise test case that attempts to show that the program does not satisfy its performance objectives.
- To ensure that the system is responsive to user interaction and handles extreme loading without unacceptable operational degradation.
- To test response time and reliability by increased user traffic.
- To identify which components are responsible for performance degradation and what usage characteristics cause degradation to occur.

## 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 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

Any computer based system that manages sensitive information or causes actions that can improperly harm (or benefit) individuals is a target for improper or illegal penetration. Penetration spans a broad range of activities : hackers who attempt to penetrate systems for sport, disgruntled employees who attempt to penetrate for revenge, dishonest individuals who attempt to penetrate for illicit personal gain.

Firewalls – a filtering mechanism that is a combination of hardware and software that examines each incoming packet of information to ensure that it is coming from a legitimate source, blocking any data that are suspect.

Authentication – a verification mechanism that validates the identity of all clients and servers, allowing communication to occur only when both sides are verified.

Encryption – Protect sensitive data by modifying it in a way that makes it impossible to read by those with malicious intent.

Authorization – a filtering mechanism that allows access to the client or server environment only by those individuals with appropriate authorization codes. (e.g. Userid , Passwords)

## 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.

Top breaches/fraud in recent times :

<https://www.networkworld.com/article/2232951/top-5-security-breaches-of-2009.html>

1. Heartland Payment Systems - In January 2009 attackers were able to steal more than 130,000,000 credit card records
2. Virginia State Prescription Monitoring Program Records - Hackers stole 8.3 million records, erased the originals and created an encrypted backup of VPMP's database
3. Terrorists intercept US Drone unencrypted Video Feeds - Islamic terrorists have been able to hack into CIA state-of-the-art Predator drones with the help of just a 25.95 dollar off-the-shelf software, raising fears of remote control operated unmanned crafts being taken over and used against British and American targets.
4. Phishing attacks on banking sites – ICICI, SBI etc.,

## 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

The goal of globalization testing is to detect potential problems in application design that could inhibit globalization. It makes sure that the code can handle all international support without breaking the functionality that would cause either data loss or display problems.

Automated testing is not an effective solution for localization/globalization testing because the default mouse click position changes with the language and also the text in the recordings don't obviously match for all languages.

## 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 can be formal, informal or heuristic based on the needs and the availability.

There are many frameworks formulated for usability testing like NIST has formulated a CIF (Common Industry Format) for reporting the findings of the test.

Interactivity – Are interaction mechanisms( pull down menus ,pointers) easy to understand

Layout – Are navigation mechanisms, content and functions placed in a manner that allows the user to find them quickly?

Readability – Is text well written and understandable ? Are graphic representation easy to understand?

Aesthetics – Do layout, color, typeface and related characteristics lead to ease of use?  
Do users feel comfortable with the look and feel.

Display Characteristics – optimal use of screen size and resolution.

Time sensitivity – Can important features, functions and content be used or acquired in a timely manner

Personalization – Does the web application tailor itself to the specific needs of different user categories or individual users?

## 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)
  - 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

It Confirms that the program recovers from expected or unexpected events without loss of data or functionality. Events can include shortage of disk space, unexpected loss of communication, or power out conditions.

## 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

Generally, Technical Writers work to tight schedules, which often does not include documentation testing because there is no time. Besides, no one wants to take the risk of causing a rewrite or correcting product design and not shipping on schedule. Bad documentation has a ripple effect on the number of users it impacts such as Product Development, Training, and Customer Support.



## 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

### Configuration Test

Analyze system behaviour:

in various hardware and software configurations specified in the requirements

sometimes systems are built in various configurations for different users for instance, a minimal system may serve a single user, other configurations for additional users.

## 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

## User Acceptance Testing

- A test executed by the end user(s) in an environment simulating the operational environment to the greatest possible extent, that should demonstrate that the developed system meets the functional and quality requirements
- Not a responsibility of the Developing Organization
- To test whether or not the right system has been created
- Usually carried out by the end user
- Two types are :
  - User Acceptance Testing
  - Operational Acceptance Test / Production Acceptance Test
  - Contract Acceptance Testing
  - Compliance acceptance testing / Regulation Acceptance Testing
  - Alpha Testing
  - Beta Testing

User Acceptance Testing (UAT) - Focuses mainly on the functionality thereby validating the fitness-for-use and is performed by the users and application managers

Operational Acceptance Test / Production Acceptance Test - Validates whether the system meets the requirements for operation and may include testing of backup/restore, disaster recovery, maintenance tasks and periodic check of security vulnerabilities

Contract Acceptance Testing - Is performed against a contract's acceptance criteria for producing custom-developed software

Compliance acceptance testing / Regulation Acceptance Testing - Is performed against the regulations which must be adhered to, such as governmental, legal or safety regulations.

Alpha Testing – It is performed at the developer's site by a cross-section of potential users and members of the developer's organization

Beta Testing - It is performed by a cross-section of users who install it and use it under real-world working conditions

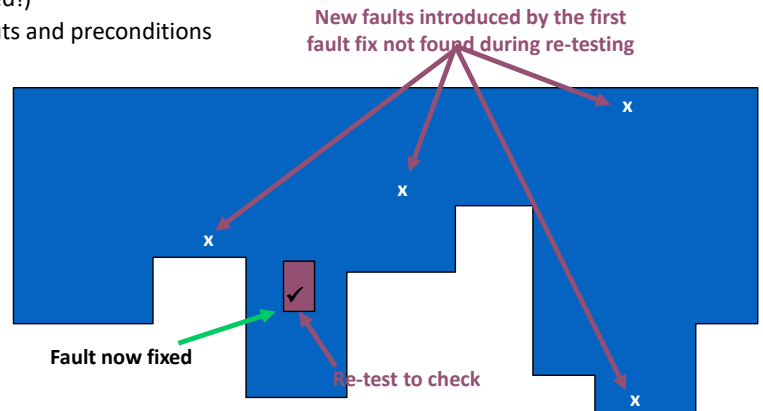
## Testing related to Changes : Re-Testing and Regression Testing

- The target of testing is the testing of changes.
- It includes below Test Types :
  - Confirmation testing (Re-testing) – Re-execution of the test after defects are fixed. The test is executed in exactly the same way as it was the first time
  - Regression testing – Ensures that the software is not adversely affected by the changes and critical functionality of the software is still intact. Generally there will be a regression test suite or regression test pack and executed whenever the software changes

## Re-testing (Confirmation Testing)

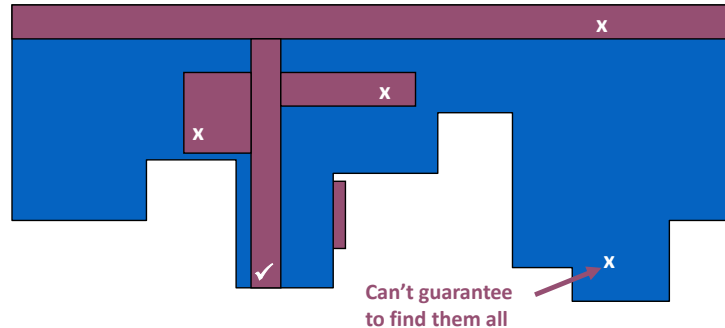
- Re-Running failed tests:

- Must be exactly repeatable
- Same environment, versions (except for the software which has been intentionally changed!)
- Same inputs and preconditions



## Regression Testing

- Regression tests are performed:
  - To look for any unexpected side-effects
  - After software changes, including faults fixed
  - When the environment changes, even if application stays same
  - For emergency fixes (possibly a subset)



All of the test cases in a regression test suite would be executed every time a new version of software is produced and this makes them ideal candidates for **automation**. If the regression test suite is very large it may be more appropriate to select a subset for execution.

Regression tests are executed whenever the software changes, either as a result of fixes or new or changed functionality. It is also a good idea to execute them when some aspect of the environment changes, for example when a new version of a database management system is introduced or a new version of a source code compiler is used.

## Maintenance Testing

- Testing done after the system is deployed or on existing system is called Maintenance testing
- It is different from maintainability testing, which defines how easy it is to maintain the system
- It consists of two parts:
  1. Testing the changes and defects
  2. Regression tests
- Impact and risk analysis is important activity performed to determine Test efforts
- It is triggered by planned modifications, Ad-hoc corrective modifications, migration, or retirement of the system

## 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
  - Exploratory testing is simultaneous learning, test design and test execution.



## 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
2. Response time
3. Aesthetics

A. Volume testing
B. Exploratory testing
C. Acceptance testing
D. Documentation testing
E. Performance testing
F. Usability testing