Testing Concepts

Introduction

- Testing is the process of executing a program with the intent of finding errors
- Testing is a process used to help identify the correctness, completeness and quality of a developed computer software
- Testing helps is Verifying and Validating if the Software is working as it is intended to be working

Need for Testing

- Contribute to the delivery of higher quality software product
- Undetected errors are costly to detect at a later stage
- Satisfied users and to lower maintenance cost

How Testing is conducted

- By examining the users' requirements
- By reviewing the artifacts like design documents
- By examining the design objectives
- By examining the functionality
- By examining the internal structures and design
- By executing code

Principles

- Economics of Testing
  - It is both the driving force and the limiting factor
- Driving - Earlier the errors are discovered and removed in the lifecycle, lower the cost of their removal.
- Limiting - Testing must end when the economic returns cease to make it worth while i.e. the costs of testing process significantly outweigh the returns

Exhaustive Testing

- Exhaustive Testing
  - Testing every possible input over every possible output
  - Can use every possible input condition as a test case
  - Is Exhaustive Testing feasible?
    - E.g. Online railway reservation system
    - Impossible to create test cases to represent all valid and invalid cases of source and destination cities

Exhaustive Testing (Contd.)

- Exhaustive testing is hence impossible
- Implications are one cannot test a program completely to guarantee that it is error free
- Objective is to therefore find maximum errors with a finite number of test cases

Limitations of Software Testing

- **Even if we could generate the input, run the tests, and evaluate the output, we would not detect all faults**
- **Correctness is not checked**
  - The programmer may have misinterpreted the specs, the specs may have misinterpreted the requirements
- **There is no way to find missing paths due to coding errors**

Psychology of Testing

- Test Engineers pursue defects not people
- Don’t assume that no error(s) will be found
- Test for Valid and Expected as well as Invalid and Unexpected
- The probability of the existence of more errors in a section of a program is proportional to the number of errors already found in that section
- Testing is extremely creative and intellectually challenging

Test Case

- “A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement”

  - Definition by IEEE

- In other words, a planned sequence of actions (with the objective of finding errors)

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

A good Test Case (Contd.)

- Assess conformance to specification
- Verify correctness of the product
- Minimize safety-related lawsuit risk
- Find safe scenarios for use of the product
- Assess quality

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

Test Case Proc - Format B

Test Case ID	Preconditions (If any)	Test Condition / Scenario	Test Steps	Input/Test Data	Expected Result
TC_1	Total of 3 marks & average of it	Validate avg field with invalid lower boundary	Enter average	avg = 0	Error1
TC_2	Total of 3 marks & average of it	Validate day field with invalid higher boundary	Enter average	avg = 101	Error2
TC_3	Total of 3 marks & average of it	Validate day field with valid boundary	Enter average	avg = 50	Message1
TC_4	Total of 3 marks & average of it	Validate day field with valid boundary	Enter average	avg = 69	Message2
TC_5	Total of 3 marks & average of it	Validate day field with valid boundary	Enter average	avg = 85	Message3

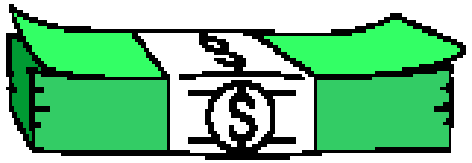
Test Case Proc - Format B

- **Popup Table with error messages**

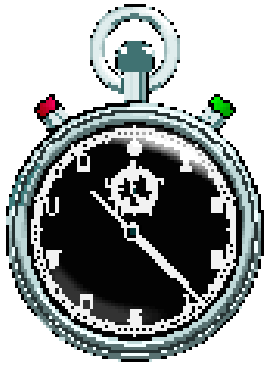
Popup Table	
Error1	Please enter valid value
Error2	Please enter valid value
Message1	Grade is second class
Message2	Grade is first class
Message3	Grade is distinction

Objectives of Testing

- To find greatest possible number of errors with manageable amount of efforts applied over a realistic time span with a finite ses

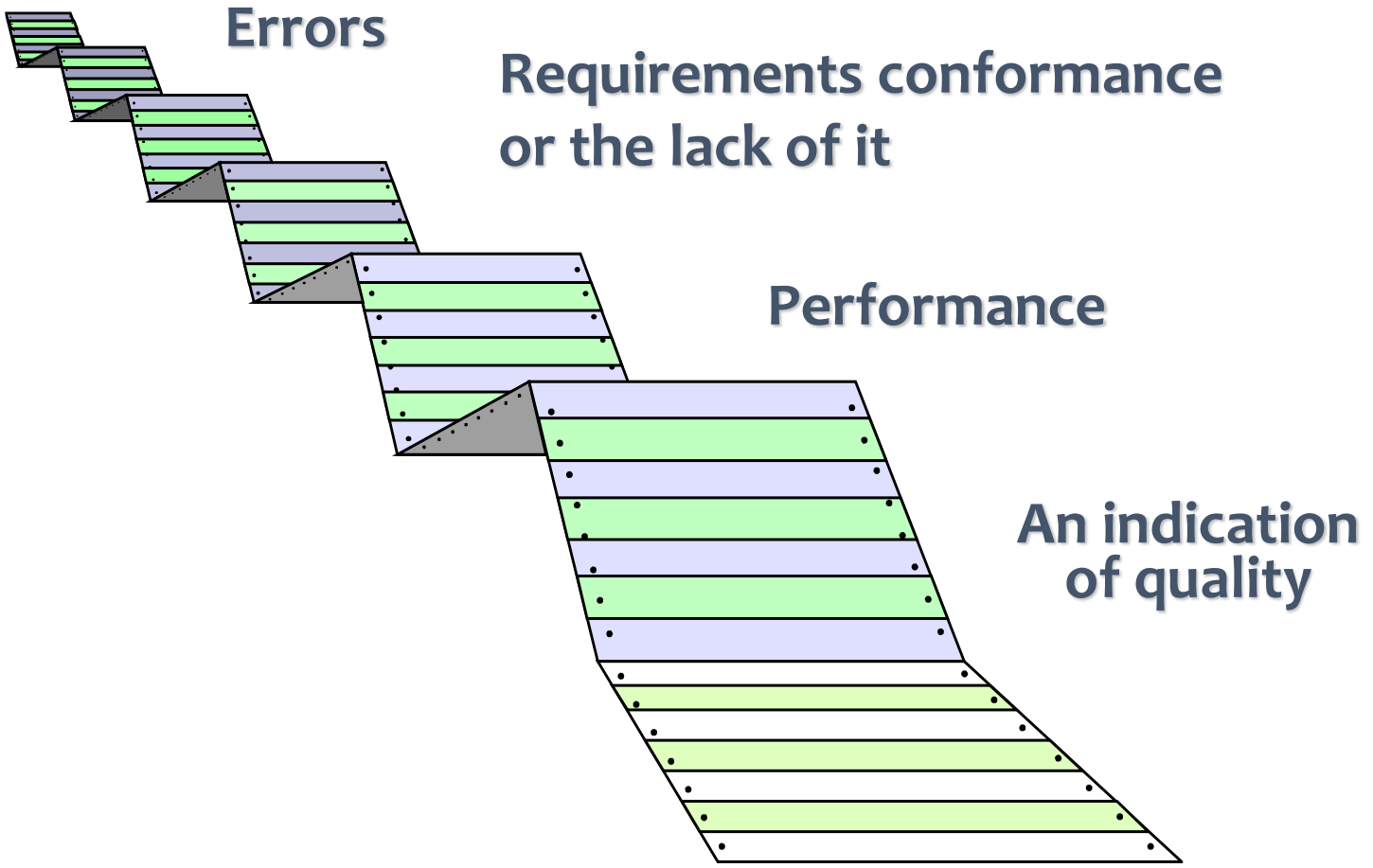


Cost effective



Time limited

What does Software Testing reveal



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

- **Validation**
  - 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.

    Example : a series of black box tests that demonstrate conformity with requirements.
  - It is ensures the functionality.
  - It is conducted by development team with the help from QC team.
  - Validation is Truth.
  - Validation is the following process of verification.

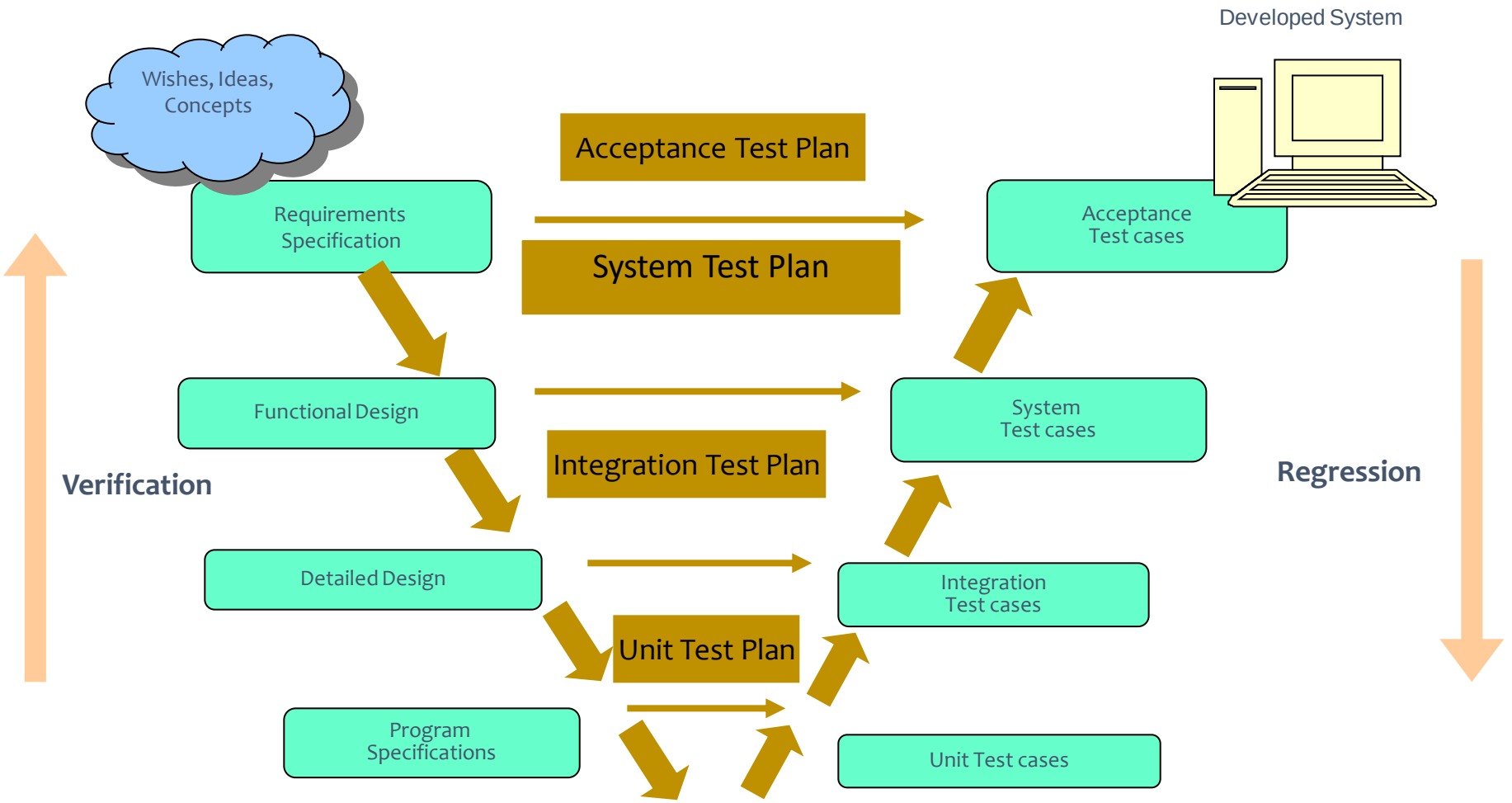
Why Write Test Cases Before Coding?

- **When adding a new feature or enhancing an existing solution, writing test cases forces you to think about what the code is supposed to accomplish.**
- **You end up with a clean and simple design that does exactly what you expect it to do.**

Introduction

- **There are some distinct test phases that take place in each of the software life cycle activity**
- **It is easier to visualize through the famous Waterfall model of development and V- model of testing**
- **The V proceeds from left to right, depicting the basic sequence of development and testing activities**

SDLC and V Model



V Model

- **The model is valuable because it highlights the existence of several levels or phases of testing and depicts the way each relates to a different development phase.**

Testing Phases

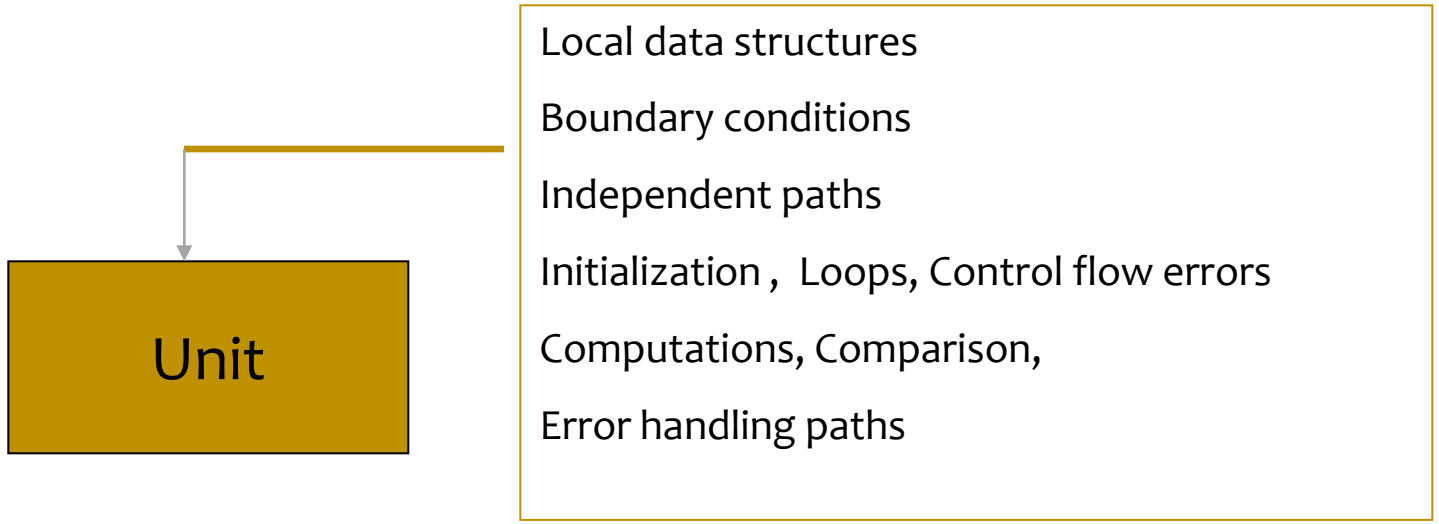
- **Unit testing**
  - Unit testing is code-based and performed primarily by developers to demonstrate that their smallest pieces of executable code function suitably.
- **Integration testing**
  - Integration testing demonstrates that two or more units or other integrations work together properly, and 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.

Introduction

- **The most 'micro' scale of testing to test particular functions, procedures or code modules. Also called as Module testing**
- **Typically done by the programmer and not by Test Engineers, as it requires detailed knowledge of the internal program design and code**
- **Purpose is to discover discrepancies between the unit's specification and its actual behavior**
- **Testing a form, a class or a stored procedure can be an example of unit testing**

Unit/Module Testing

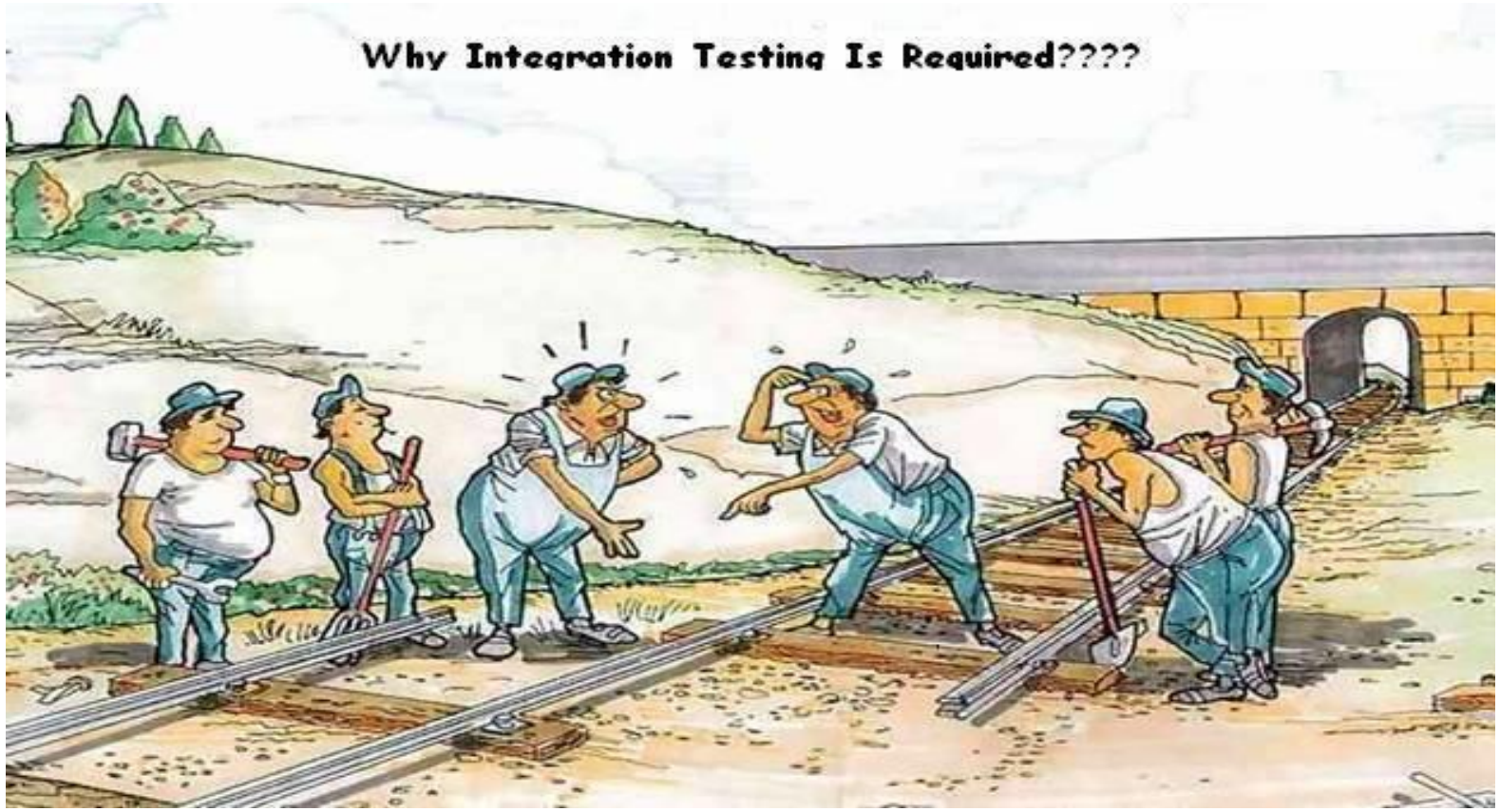
- **Unit testing uncovers errors in logic and function within the boundaries of a component.**



Introduction

- **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**

Why Integration Testing is Required?

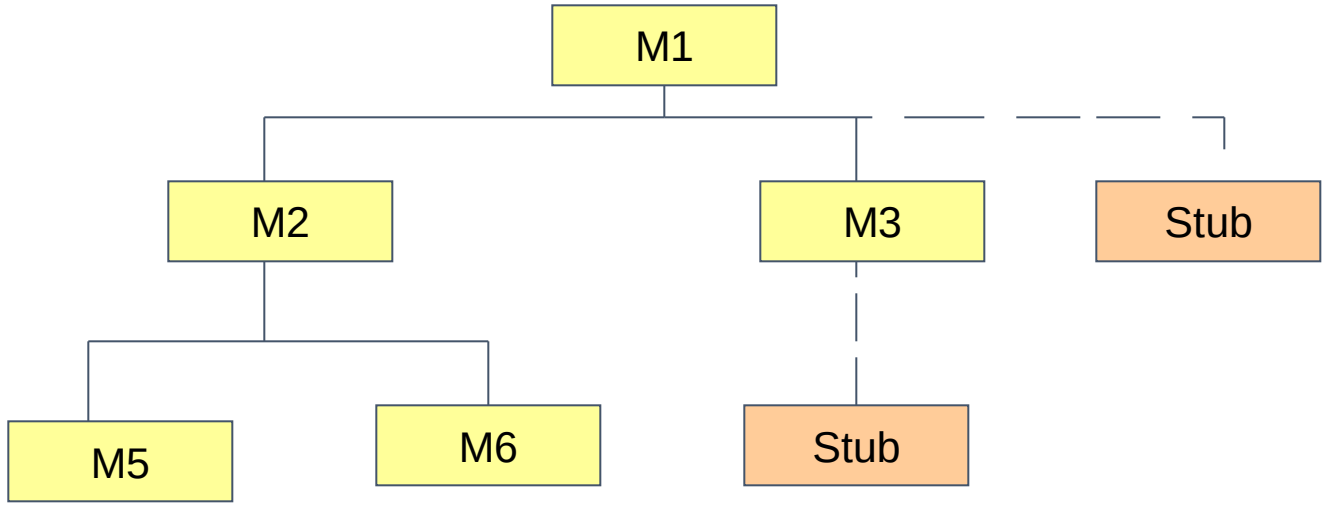


Types

- **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.
    - There are two types by which incremental module testing is achieved.
      - 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.

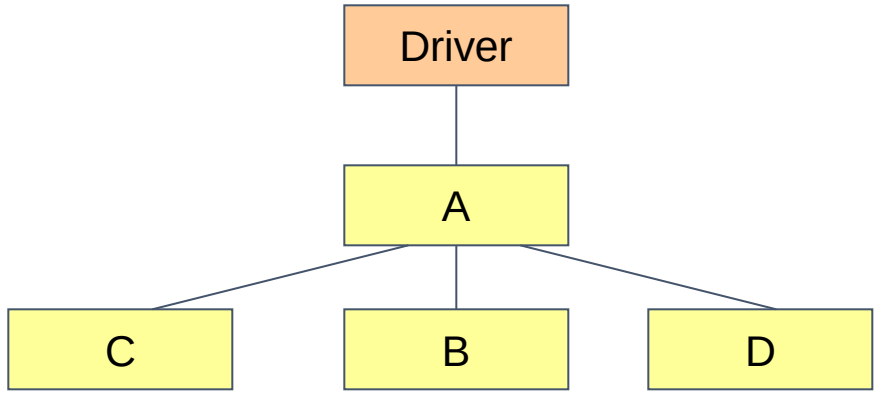


Top Down Integration Testing

- **Integration approach can be done Depth first or Breadth-first.**
- **Top down testing**
  - 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



Bottom Up Integration Testing

- **Bottom-Up testing**
  - 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	

Introduction

- **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.**

Types of System Testing

- **Types of System Testing**
  - Functional testing
  - Regression testing
  - Performance
  - Volume
  - Stress
  - Security
  - Usability
  - Recovery
  - Documentation
  - Configuration
  - Installation

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).

Regression Testing

- **Regression Testing is the testing of software after a modification has been made to ensure the reliability of each software release.**
- **Testing after changes have been made to ensure that changes did not introduce any new errors into the system.**
- **It applies to systems in production undergoing change as well as to systems under development**
- **Re-execution of some subset of test that have already been conducted**
- **Test suite contains**
  - Sample of tests that will exercise all software functions
  - Tests that focus on software functions that are likely to be affected by the change
  - Tests for software components that have been changed

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

Performance Testing

- **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.

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

- **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.**

Security Testing Vs Functional Testing

- **Functional testing checks for:**
  - Invalid links (outgoing/internal/broken)
    - Is forward/backward link representing to the correct page?
  - Validations in each fields in the forms or web pages
    - Is Username filed accepting any special characters?
    - Is date field accepting the correct format specified?
  - HTML/CSS syntactical errors Etc.
  - **Security testing is all about:**
  - With the help of a hyper links is it possible-
    - To access restricted resources like documents?
    - To insert any piece of code that could do the damage?
    - To upload any executable program?
  - Client side validations in each fields-
    - Properly validated fields - can they be bypassed- YES
  - HTML/CSS syntactical errors
    - Error pages can reveal sensitive information
    - Even a single Quote- ‘ - can do the damage

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

- **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

Introduction

- **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 :**
  - ALPHA TESTING :Generally in the presence of the developer at the developers site
  - BETA TESTING : Done at the customers site with no developer in site

Introduction

- **Also known as “Random” testing or “Ad-hoc” testing**
- **Exploratory testing is simultaneous learning, test design, and test execution. (… James Bach)**
- **A methodical approach-style is desirable**

What Is a Test Strategy?

- **It provides a road map that describes the steps to be conducted as part of testing**
- **When these steps are planned then how much effort, time and resources will be required are undertaken**
- **It must incorporate test planning, test case design, test execution and resultant data collection and evaluation**

Pre-execution activities

- Setting up the Environment
  - Similar to production environment
  - Hardware (e.g. Hard Disk, RAM, Processor)
  - Software (e.g. IE, MS office)
  - Access to Applications
- Setting up data for Execution
  - Any format (e.g. xml test data, system test data, SQL test data)
  - Create fresh set of your own test data
  - Use existing sample test data
  - Verify, if the test data is not corrupted
  - Ideal test data - all the application errors get identified with minimum size of data set

Pre-execution activities contd…

- Test data to ensure complete test coverage

➢ **Design test data considering following categories:**

  - No data
    - Relevant error messages are generated
  - Valid data set
    - Functioning as per requirements
  - Invalid data set
    - Behavior for negative values
  - Boundary Condition data set
    - Identify application boundary cases
  - Data set for Performance, Load and Stress Testing
    - This data set should be large in volume

Types of Test Environment

- Unit Test Environment
- Assembly/Integration Test Environment
- System/Functional/QA Test Environment
- User Acceptance Test Environment
- Production Environment

Before starting Execution

Validate the Test Bed

- Environment
  - Hardware (e.g. Hard Disk, RAM, Processor)
  - Software (e.g. IE, MS office)
- Access
  - Access to the Application
  - Availability of Interfaces (e.g. Printer)
  - Availability of created Test Data
- Application
  - High level testing on the application to verify if the basic functionality is working
  - There are no show-stoppers
  - Referred to as Smoke/Sanity/QA Build Acceptance testing

Entry Criteria for Functional Testing

Functional/System Testing Entry Criteria
➢ Integration Testing is complete and sign-off is received by Project team
➢ Integration test results are provided to the QA team within the Integration Execution & Signoff artefact.
➢ Development team provides a demonstration of application changes prior to promotion to QA Environment
➢ Code is delivered and successfully promoted to the Functional/System Test Environment as described in Master Test Plan
➢ Functional/System Test planning is detailed, reviewed and approved within the Master Test Plan

Entry Criteria for Functional Testing Contd…

Functional/System Testing Entry Criteria
➢ Smoke /Shake down test has been completed to ensure test environment is stable for testing.
➢ Functional/System Test planning is detailed, reviewed and approved within the Master Test Plan
➢ Test cases created are traceable back to SRS, and any approved change requests, using HPQC
➢ Functional/System Test Cases are created, reviewed and approved within the RBC Enterprise approved tool (HP QC)
➢ Test data is ready for Functional/System Testing

Execution

- **Run Tests**
  - Run test on the identified Test Bed
  - Precondition
  - Use the relevant test data
- **Note the Result**
  - Objective of test case
  - Action performed
  - Expected outcome
  - Actual outcome
  - Pass/Fail (according to pass/fail criteria)
- **Compare the Input and Output**
  - Validate the data (e.g. complex scenarios, data from multiple interfaces)
- **Record the Execution**
  - Test data information (e.g. type of client, account type)
  - Screenshots of the actions performed and results
  - Video recording (HP QC Add-in)

Execution contd…

- Report deviation
  - Log Defect for Failed Test Cases
  - Defect logging
    - Project
    - Summary
    - Description
    - Status
    - Detected By
    - Assigned To
    - Environment (OS, Release, Build, Server)
    - Severity
    - Priority
    - Steps to recreate and Screenshots

Exit Criteria for Functional Testing

Functional/System Testing Exit Criteria
➢ All high and medium risk tests identified in the detailed test plan are executed, including interface testing
➢ All planned testing is complete and documented
➢ Functional/System test execution results are captured
➢ All known defects have been entered into the defect tracking tool
➢ There are no known severity one or severity two defects
➢ Action plans have been created for outstanding severity three and four defects

Exit Criteria for Functional Testing Contd..

Functional/System Testing Exit Criteria
➢ Appropriate signoffs are obtained
➢ Location of test cases, automated test scripts, defects and Functional/System Execution & Signoff artefact are detailed within the SCM plan.
➢ Any known deviations from the BRD and SRS are documented and approved