iamneo

Levels of Testing

Levels of Testing

Testing levels are the procedure for finding the missing areas and avoiding overlapping and repetition between the development life cycle stages.

Levels of Testing

The key of successful test strategies is picking the right level of testing at each stage of the project .

“providing weeks of tedious repetitive testing and mountains of problem reports “ in the project , bugs are still found, there is rework to carry out and inevitably yet more testing. But with careful planning, software testing can go like a dream”

Key Factor :

This sort of decision needs to be made at an early stage in a project and will be documented in an overall software test plan.

Typically this general plan will be drafted at the same stage that the requirements are catalogued and would be finalized once functional design work has been completed**.**

The V-Model

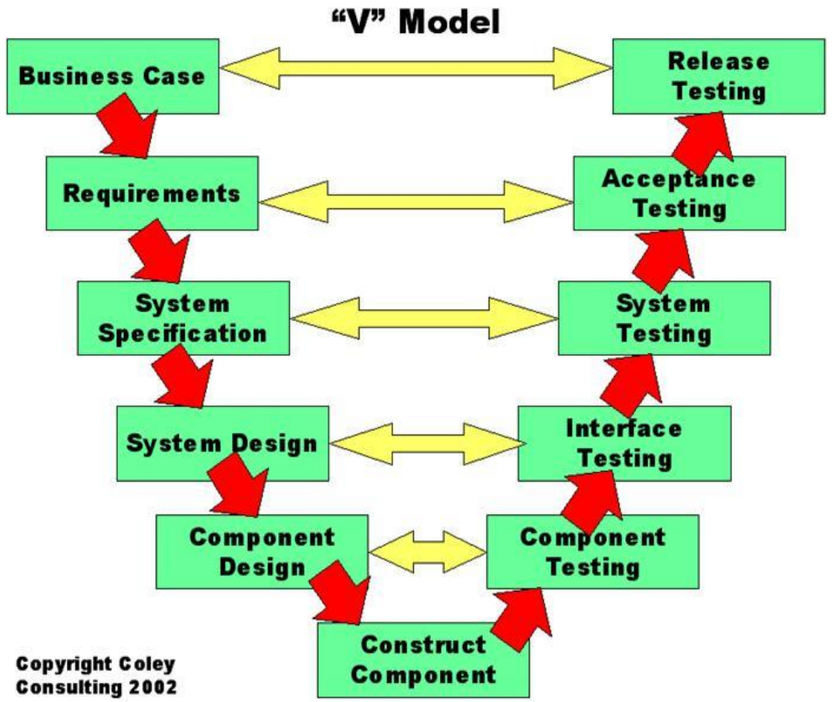
Followings are the Levels used in the project across the organization,

↔ Component testing (or) Module testing (or) Unit testing

↔ Integration Testing

↔ System Testing

↔ User Acceptance Testing



Unit Testing

How Does Unit Testing level fit into the Project ?

- As soon as the programmer develops the units of code or Module the unit tested for various scenarios. It is much more economical to find and eliminate the bugs early on.
- A unit is the smallest testable part of an application. In procedural programming a unit may be an individual program, function, procedure, etc.
- A manual approach to unit testing may employ a step-by-step instructional document. Nevertheless, the objective in unit testing is to isolate a unit and validate its correctness. The purpose of unit testing is to verify individual hardware or software units, or small groups of related code unit. In Unit Testing, the emphasis is on removal of basic coding errors and also, compliance with programming style and standards.

Unit Testing Expectations

- Unit testing takes place in development environment and we may need to create test data to follow a particular code path or test specific test cases
- Ideally, every path/line of code that is new or modified should be executed at least once. If the paths in program element are difficult to test then it is recommended to have a detailed code inspection to verify functionality.
- All programs and data should be verified for compliance with coding standards and programming style in order to find violation in naming conventions, conditions on usage of go-to, indention in code, and nesting of loops.
- All programs shall have manageable complexity and shall be built on modularity principles. Using techniques like Cyclomatic Complexity, the complexity of code shall be measured, and the same shall be reported.

Unit Test Considerations

Unit Test focuses on:

- Process or Logic
- Interface
- Local data structures
- Boundary conditions
- Independent paths
- Error handling paths

Tasks in Unit Testing

- **Arrive at Unit Test Plan**

A detailed Unit Test Plan must be in place. This plan can be part of the over all Test Plan or a separate plan by itself. Following are the activities that are necessary for creating a detailed Unit Test Plan.

  - ✓ Determine the types of unit tests required for the software unit
  - ✓ Review the detailed Unit Test Plan
  - ✓ Approve the Unit Test Plan

- **Identify all Interfaces to other units**

Determine if other units are available or if unit stubs must be written. Consider using stubs to capture intermediate results if other automated tools are not available

Tasks in Unit Testing

- Identify the Test Procedure and identify Test Cases for performing the Unit Testing

    1. **Path Testing** - Execution of every logic branch and line of code to find logic errors in control structures, dead code, errors at loop boundaries, and errors in loop initializations. This includes every state and every mode.
    2. **Boundary Condition Testing** - To find errors in input and output parameter tolerances and verify that the program limits are correctly stated and implemented.

Tasks in Unit Testing

- **Perform Unit Testing**
  - ✓ Performing the Unit Test includes:

    Perform the test in accordance with the unit test plan
  - ✓ Compare test results with expected results
  - ✓ Document the results
  - ✓ If the unit did not successfully pass unit test, schedule the unit for rework
- **Prepare Unit Test Report**

Test Report captures the result of the unit test activity. After testing, a detailed test report shall be generated to analyze the test results and take corrective actions. The test report contains results of the testing, indicating the compliance or non-compliance with coding standard, programming style, program specification, or design specification used in writing a code.

Unit Testing

Using an automation framework, the developer codes criteria into the test to verify the correctness of the unit.

During execution of the test cases, the framework logs those that fail any criterion

Limitations :

This type of testing level cannot be expected to catch every error in the program or module .

Unit tests can only show the presence of errors; it cannot show the absence of errors

System Integration Testing

Integration level Testing :

“ Integration testing “ is the Next level of testing.

Integration Testing is a level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

How Does Unit Testing level fit into the Project ?

Once unit tested components are delivered we than integrate them together

Prerequisites for Integration testing

Before we begin Integration testing it is very important that all the components have been successfully unit tested

System Integration Testing

Integration testing involves:

- Integrating independent software units or components to form a sizeable build and then testing the assembly.
- To find any issues in interfaces among units or components
- To find any gaps in the data flow

System Integration Testing

Integration testing takes place either in development environment or in test environment using real data (data which was processed on the legacy system and is now being re-used for testing), if possible; else simulated data need to be created to model real data. Integration testing verifies inter-component interfaces that would be helpful in evolving the system. Integration testing includes functional testing.

Integration testing strategy :

- Big bang approach
- Top-down approach
- Bottom –up approach
- Mixed approach

System Integration Testing

Big Bang : All the modules are simply put together and tested . This technique is used only for very small systems

Problems : Debugging error found during big bang integration are very expensive to fix .

Bottom -Up approach :

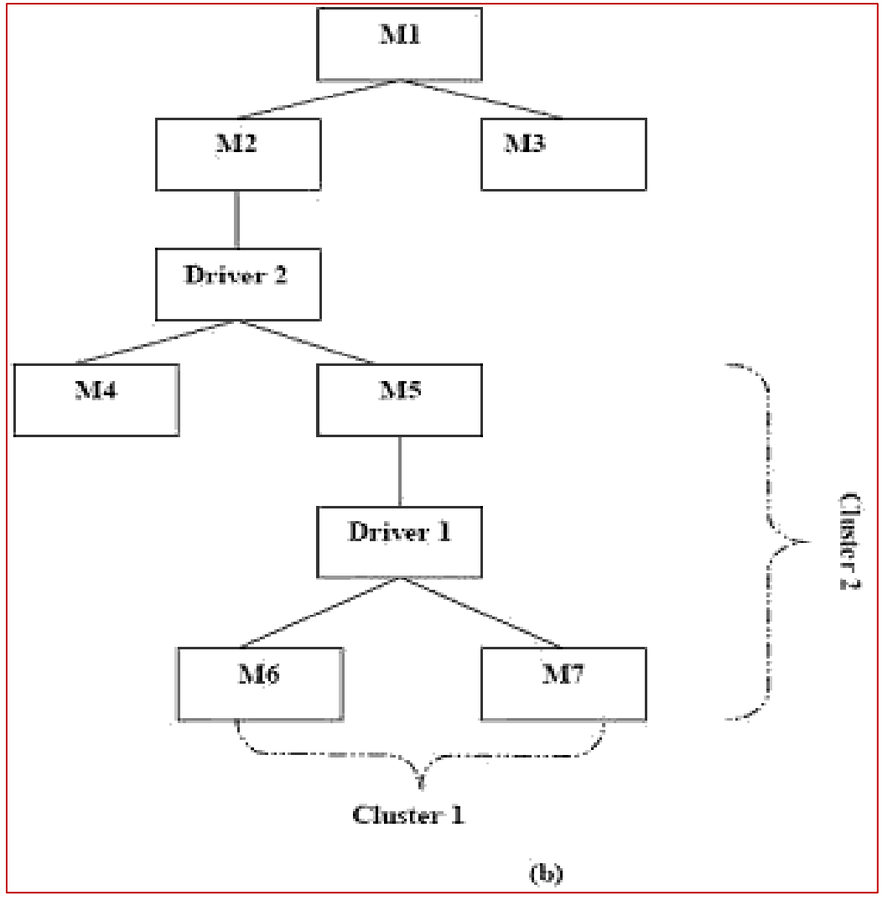
- It is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher level components.
- The process is repeated until the component at the top of the hierarchy is tested.
- This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage.

Demerits :

- Major control problems found last and need drivers and stubs

System Integration Testing

Flow of Bottom up approach

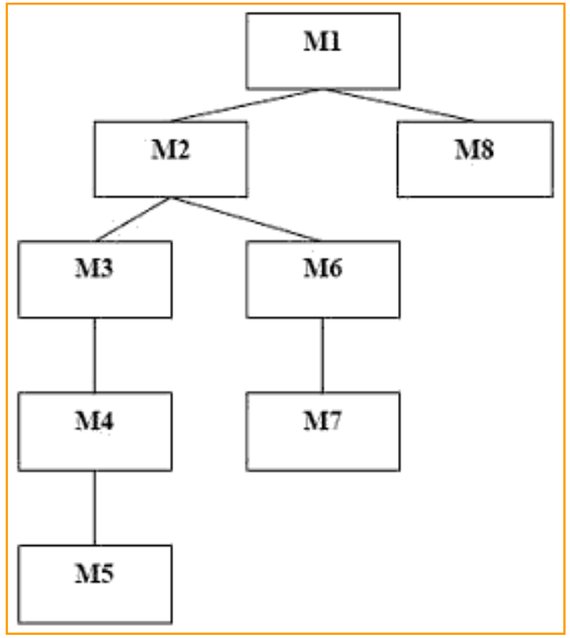


System Integration Testing

Top-down approach :

The highest-level modules be test and integrated first. This allows high-level logic and data flow to be tested early in the process and it tends to minimize the need for drivers.

The branch of the module is tested step by step until the end of the related module.



System Integration Testing

Sandwich or Mixed level testing :

It is an approach to combine top down testing with bottom up testing.

Summary

The main advantage of the Bottom-Up approach is that bugs are more easily found With Top-Down, it is easier to find a missing branch link

Notes:

Stub and Driver

It is dummy modules that simulates the low level modules. Drivers is the dummy modules (to drive) that simulate the high level modules.

System Testing

System Level Testing :

The first time entire system testing can be tested as a Whole system against the FRS and SRS

Focus :

Where the focus is to have almost a destructive attitude and test not only the design and believed expectations of the customer.

Techniques :

System Testing

- System Testing involves testing the system as a whole. System Testing commences after completion of unit, functional, and integration testing. The emphasis is on verifying end-to-end workflows and scenarios. All the software components, all the hardware components, should be tested.
- The whole system is checked not only for validity but also for met objectives. The system test is intended to verify program performance in accordance with the system specification and those requirement specifications referenced from the RS document.
- System Testing requires system test environment that comprises of deployment like environment from hardware and software requirements. In addition, it may require test automation tools and analysis tools. In addition, we may need test data. As days are progressing, part of system testing is happening in the live environment as well which may include performance testing and external interface testing including interoperability testing.

System Testing Techniques

Structural System Testing Techniques:

Stress : Determine system performance with expected volumes

Execution : System achieves desired level of proficiency

Operations : System can be executed in a normal operational status

Security : System is protected in accordance with importance to organization

System Testing Techniques

Functional System Testing Techniques

- Requirements : System performance as specified (E.G) System Req
- Intersystems : Data is correctly passed from system to system
- Manual support : The people – computer interaction works
- Regression : Verifies that anything unchanged still performs correctly

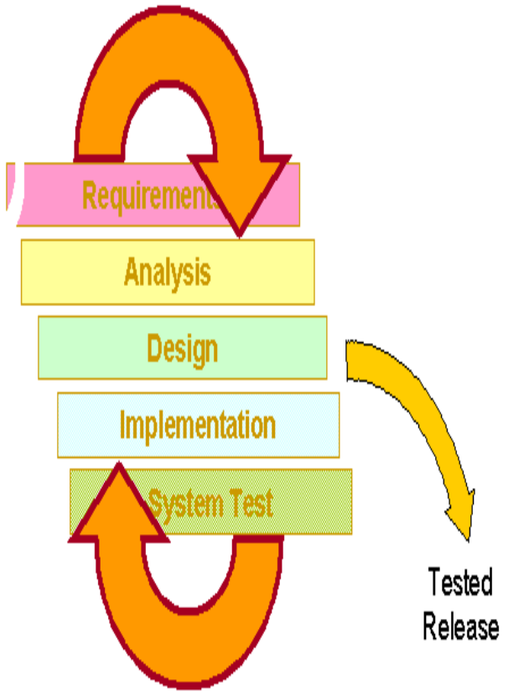
Systems Testing

When the System testing fit in to the Project ?

Once all the modules are integrated , several errors may arise or successfully completed then system testing done at this stage.

Types under System Testing : some examples

1. Security testing
2. Report /document testing
3. Sanity /Smoke testing
4. Exploratory testing
5. Ad-Hoc testing
6. Performance testing.



Etc.....

System Test Expectations

- The primary emphasis is verification of the system as a whole. Typically this means verifying all hardware and software requirements from a business workflow perspective. There should be a minimal number of errors and no critical errors in this test phase. If there are, it may indicate a lack of readiness to proceed.
- This serves as the final verification of requirements and design (for those items that can be tested at this time; some requirements are monitored throughout the life of the contract).
- Some performance tests may be conducted and used to model or extrapolate behaviour.
- All affected documentation should be updated to reflect fixes and changes, including in-line code comments and unit/component/function headers, design documents, user manuals, help desk procedures or bulletins, and help files. Training materials should be finalized during this phase.
- If there were any non-testable requirements, they should be verified at this time. This may include difficult code paths, or such things as the help desk, training materials, etc.

System Test Considerations

- Perform Functional Tests

Functional Tests include Functional Operability Testing (FOT) and Functional Stress Testing (FST). FOT tests the functional requirements of the SRS and FST tests the stress requirements. FOT and FST can be combined within a single set of procedures.

- Perform Regression Tests

Regression tests are run to verify that program changes implemented after the beginning of System Testing have not introduced program regression.

System Test Considerations

- Perform Load / Stress Tests

These tests are designed to satisfy the load and stress requirements for critical computer programs. Three periods of maximum stress are distributed throughout at least a 25-hour period.

- Perform Performance Test

Performance tests are conducted to evaluate the performance of a system or component with specified performance requirements. The performance requirements can be obtained from software requirements specifications. These tests are concerned with knowledge about validating the performance requirements of a system. These tests include knowledge about techniques to instrument performance measures like logging, event counts, event duration, and sampling. It also includes knowledge about methods for tuning a system for optimum saturation, load, and throughput threshold.

System Test Considerations

- Perform Security Test

In a networked environment the security test is performed to test the safety of the application and the system. This test includes checking the unauthorized activities on the application and the system such as hacking, illegal use of the application, etc.

- Arrive at Detailed System Test Plan

A detailed System Test Plan includes a detailed plan for the various tasks that are to be performed. This plan includes the time schedule, resource and the responsibilities for the activities to be performed. The detailed System test plan is made in accordance with the requirements specifications and requirements for the system testing.

Tasks in System Testing

- Perform System Tests

  System testing focuses on:

  - ✓ Building the system
  - ✓ Perform system testing by executing all the identified test cases or test scripts.
  - ✓ Log issues arising during the testing

- Prepare System Test Report

System test report is a result of the system test activity. After testing, a detailed test report is generated to analyze the test results and take corrective actions. The Test report contains results of the testing, indicating the compliance or non-compliance of the specific requirement

Acceptance Testing

The purpose of acceptance testing is for the users to verify the system or changes meet their original needs. The emphasis is on evaluating the system via normal business circumstances, but in a controlled testing environment.

Formal testing conducted to enable a user, customer, or other authorized entity to determine whether to accept a product or product component.

Alpha & Beta Testing

Alpha testing : In house virtual user environment can be created for this type of testing. Testing is done at the end of development. Still minor design changes may be made as a result of such testing.

Beta testing : Testing typically done by end-users or others. Final testing before releasing application for commercial purpose.

In a Product based company:

❖Alpha Testing defines as the testing which is done at the developer's site by the testers.

❖Beta testing is the testing done by the tester by generating the customer's environment.

In a Service based company:

❖Alpha testing is the testing conducted by customers at Developer's site.

❖Beta Testing is the process of giving the product to customers and let them do the testing at their environment

Summary - Levels of Testing

Levels of Testing	Description
Unit Testing	A unit is the smallest testable part of an application. it may be an individual program, function, procedure, etc.
Integration Testing	Black-box type testing that is based on overall requirements specifications; covers all combined parts of a system
System Integration Testing	Testing of combined parts of an application to determine if they function together correctly. The 'parts' can be code modules, individual applications.
E2E/System Testing	Black-box type testing is a testing process that exercises a software system's coexistence with others
UAT	Determining if software is satisfactory to an end-user or customer.