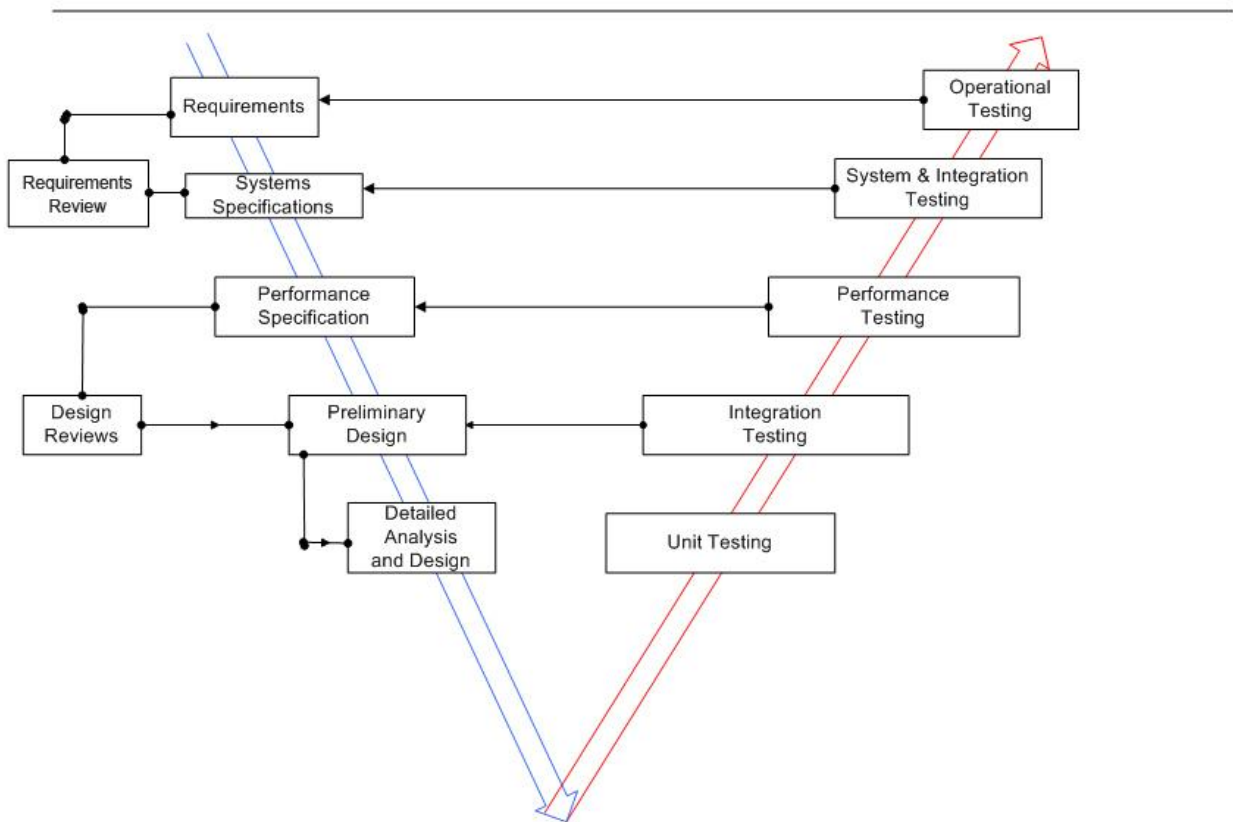




## TESTING AT DIFFERENT PHASE OF SOFTWARE DEVELOPMENT LIFE CYCLE:

Software Testing Flow Diagram





## Verification and Validation Testing:

Verification:

"Are we building the product right?"

The software should conform to its specification

Validation:

"Are we building the right product?"

The software should do what the user really requires.

You verify program by checking it against the most closely related design document (s) or specification(s), if there is an external specification, the functional testing verifies the program against it.

You validate a program by checking it against the published user or system requirements.

System testing and Integrity testing are validation tests.

One should follow standard IEEE for Software Verification and Validation Plans  
(ANSI/IEEE Standard 1012-1986)

### Verification activities:

Formal reviews, technical reviews, and inspections are various expressions used for the more structured types of verification.

They are also characterized by individual preparation by all participants prior to the meeting.

Participants are told by the inspection team leader what their reviewer role is.

When performing inspections well, most of the total defects to be discovered are found during preparation, though an effective inspection meeting will usually uncover some significant additional defects.

Walkthroughs are less formal than inspections mainly because of the lack of preparation. In walkthroughs the participants simply come to the meeting. The presenter prepares (usually the author of the product), and there's no additional effort by the participants prior to the meeting.



## Validation activities:

Validation activities can be divided into the following.

- 1) Low-level testing
  - i) Unit (module) testing
  - ii) Integration testing.
  
- 2) High-level testing
  - i) usability testing
  - ii) function testing
  - iii) system testing
  - iv) acceptance testing

## Unit Testing:

A System is developed piecemeal, as a collection of processes and modules. The distinction between incremental and big bang testing boils down to a choice between testing the product piecemeal or in one big lump.

Under an incremental testing strategy, each piece is first tested separately. The testing of individual pieces of a process or program is called module testing, unit testing, or element testing.

Module testing (or unit testing) is a process of testing the individual subprograms, subroutines, or procedures in a program. That is rather than initially testing the program as a whole, testing is first focused on the smaller building blocks of the program. The motivations for doing this are threefold.

First, module testing is a way of managing the combinatory of testing, since attention is focused initially on smaller units of the program.

Second, module testing eases the task of debugging (the process of pinpointing and correcting a discovered error), since, when an error is found, it is known to exist in a particular module.

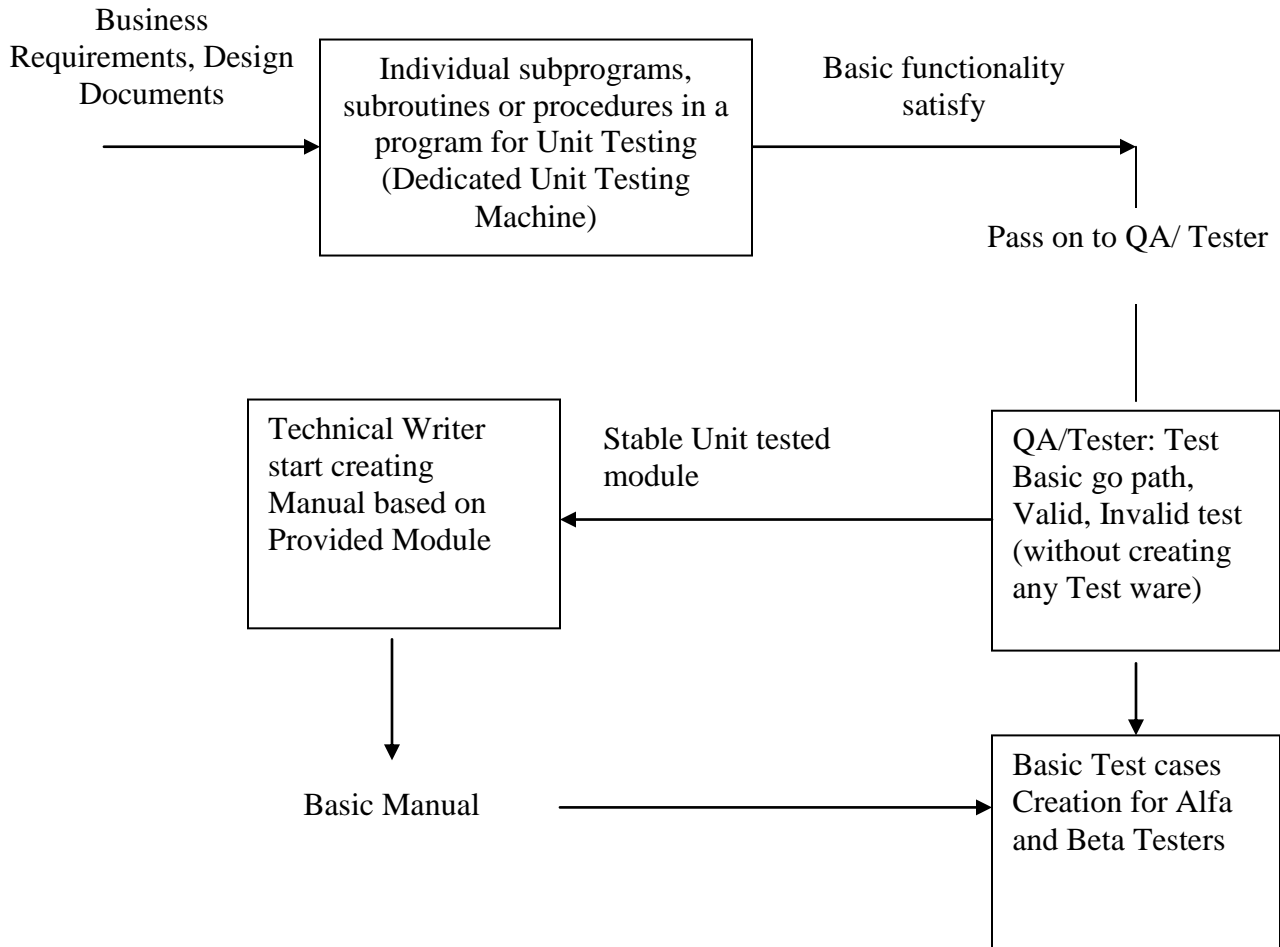
Finally, module testing introduces parallelism into the testing process by presenting us with the opportunity to test multiple modules simultaneously

## Benefits:

1. By Unit Testing We are checking initial Stability and functionality for the provided module
2. Before integrating particular module to the daily build, Testers understand its function and its importance as a part of visual plant
3. It's useful to cover more test cases by knowing its effect on other module as well as overall system



### Unit Testing Process





### **Integration Testing:**

Testing combinations of pieces of the product is called integration testing. The primary objective of integration testing is to discover errors in the interfaces between the components.

### **Usability Testing:**

Usability testing is the process of attempting to identify discrepancies between the user interfaces of a product and the human engineering requirements of its potential users.

It often involves evaluation of a product's presentation rather than its functionality.

Usability testing is considered a validation activity rather than a verification activity, because it requires a real user to interact with the end product.

Usability characteristics which can be tested include the following:

Accessibility, Responsiveness, Efficiency, Comprehensibility.

### **Functional Testing:**

Functional testing is the process of attempting to detect discrepancies between a program's functional specification and its actual behavior.

Functional testing is performed by a testing group before the product is made available to customers. It can begin whenever the product has sufficient functionality to execute some of the tests, or after unit and integration testing have been completed.

The steps of functional testing are:

1. Decompose and analyze the functional design specification
2. Partition the functionality into logical components and for each component; make a list of the detailed functions.
3. For each function, use the analytical black-box methods to determine inputs and outputs.
4. Develop the functional test cases
5. Develop function coverage matrix
6. Execute the test cases and measure logic coverage
7. Develop additional functional tests, as indicated by the combined logic coverage of functional and system testing.



## System Testing:

System testing is the process of attempting to demonstrate that a program or system does not meet its original requirements and objectives, as stated in the requirements specification. System testing is difficult. **There can be no design methodologies for test cases because requirements and objectives do not, and should not, describe the program's functions in precise terms.**

This is an ideal way to test user documentation, but it is often impractical because the manuals are usually not available when system test cases must be formulated.

Types/goals of system testing are as follows:

### Volume testing:

To determine whether the program can handle the required volumes of data, request etc.

### Load/stress testing:

To identify peak load conditions at which the program will fail to handle required processing loads within required time spans

### Security testing:

To show that the program's security requirements can be subverted.

### Usability (human factors) testing:

To identify those operations that will be difficult or inconvenient for users. Publications, facilities and manual procedures are tested.

### Performance testing:

Performance can be tested using glass box or black box techniques. Glass box testing allows a finer analysis because you can use profilers or hardware-based execution monitors to study the time the program spends in specific modules, along specific paths, or working with specific types of data.

One objective of performance testing is performance enhancement. The tests might determine which modules execute most often or use the most computer time. Those modules are re-examined and recoded to run more quickly.

Testing groups do black box performance tests. They use benchmark tests to compare the latest version's performance to previous versions. Poor performance can reflect bugs, especially when a part of the program that used to run quickly is now slow.



**Configuration testing:**

To determine whether the program operates properly when the software or hardware is configured in a required manner.

**Compatibility/conversion testing:**

To determine whether the compatibility objective of the program have been met and whether the conversion procedures work.

**Installability testing:**

To identify the ways in which the installation procedures lead to incorrect results.

**Recovery testing:**

To determine whether the system or program meets its requirements for recovery after a failure.

**Serviceability testing:**

To identify conditions whose serviceability needs will not meet requirements.

**Reliability/availability Testing:**

To determine whether the system meets its reliability and availability requirements.

**Acceptance Testing:**

Acceptance testing is the process of comparing the end product to the current needs of its end users. It is usually performed by the customer or end user after the testing group has satisfactorily completed usability, function and system testing. It usually involves running and operating the software in production mode for a pre-specified period.

If the software is developed under contract, acceptance testing is performed by the contracting customer. Acceptance criteria are defined in the contract. If the product is not developed under contract, the developing organization can arrange for alternative forms of acceptance testing- ALPHA and BETA

ALPHA and BETA testing are each employed as a form of acceptance testing.

The ALPHA test is usually performed by end users inside the developing company but outside the development organization. The BETA test is usually performed by a select subset of actual customers outside the company, before the software is made available to all customers.

The first step in implementing ALPHA and BETA is to define the primary objective of the test: progressive testing, and/or regressive testing. Progressive testing is the process of testing new code to determine whether it contains errors. Regressive testing is the process of testing a program to determine whether a change has introduced errors (regressions) in the unchanged code.

Both ALPHA and BETA are often more effective as regressive, rather than progressive, tests.