Mahindra Satyam

Testing Throughout SDLC

ISTQB

Learning Objectives

- Software Development Models
  - V-model (Sequential Development Model)
  - Iterative-incremental Development Models
- Test Levels
  - Component Testing
  - Integration Testing
  - System Testing
  - Acceptance Testing
- Test Types
  - Functional Testing
  - Non-Functional Testing
  - Structural Testing
  - Re-Testing and Regression Testing
- Maintenance Testing

Software Development Life Cycle

- The Software Development Life Cycle is a step-by-step process involved in the development of a software product.
- The whole process is generally classified into a set of steps and a specific operation will be carried out in each of the steps.
- The basic classification of the whole process is as follows:
  - **Initial**
  - **Analysis**
  - **Design**
  - **Coding**
  - **Testing**
  - **Maintenance**

Initial:

- The main aim of the initial phase is requirements gathering.
- The responsible person is “BA – **Business Analyst**”
- The outcome document name is “**BRS – Business Requirements Specification**”.

Contd..

Analysis:

- In this phase requirements are analyzed for the feasibility of implementation.
- Planning the project, Resource estimation, Schedule preparation, project prototype will takes place.
- The outcome document name is "**SRS – System Requirements Specification**".
- At the end of this phase, both project plan and test plan documents are delivered.

Design:

- The aim is to create the architecture of the total system(including database design).
- Dividing the system into no. of module and sub-modules.
- Preparing Data-flow diagrams, Control-flow diagrams, Use cases.
- Designing Database
- The outcome document name is "**TDD – Technical Design Document**".

Coding:

- In this phase, programmers prepares/write the code for given specification in the chosen programming language.
- The outcome is "**Source code**".

Contd..

Testing:

- In this phase, testing team will execute the prepared test cases on the given build/system.
- Main aim is, to verify that the given system working as per client requirements.
- If test fails, they will raise an incident.
- The outcome documents are: **Test Cases, Incidents.**

Maintenance:

- In this phase, the system deployed into the client's environment.
- Any issue raised by client, environmental problems are handled.
- Requirements change requests are handled
- Maintenance is made up of:
  - **Corrective maintenance**
    - Ex: fixing customer-reported problems
  - **Adaptive maintenance**
    - Ex: making the software run on a new version of an OS or Database.
  - **Preventive maintenance**
    - Ex: changing the application program code to avoid a potential security hole in an OS code

Mahindra Satyam

Software Development Models

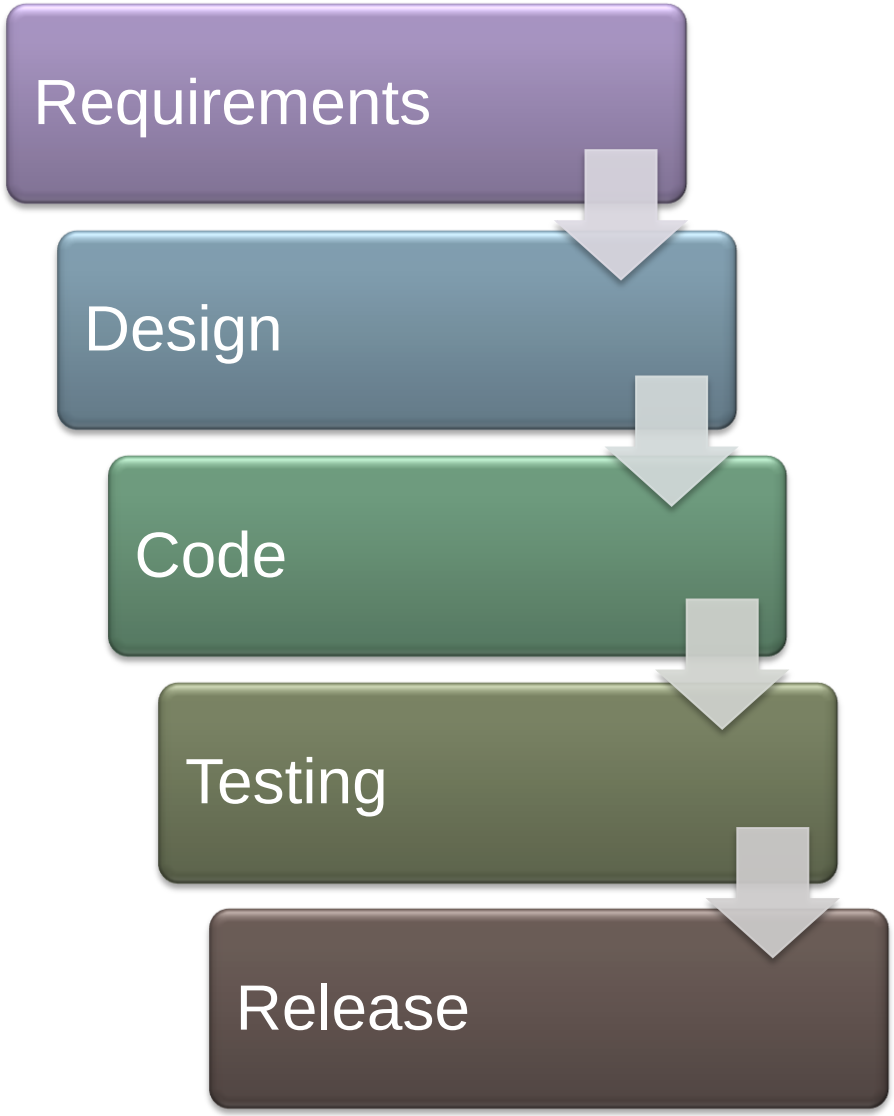
Mahindra Satyam

Waterfall Model

Software Development Models – Waterfall model

Waterfall Model

- In "The Waterfall" approach, the whole process of software development is divided into separate process phases.
- All these phases are cascaded to each other so that second phase is started as and when defined set of goals are achieved for first phase and it is signed off, so the name "**Waterfall Model**".
- All the methods and processes undertaken in Waterfall Model are more visible.



Contd..

Advantages

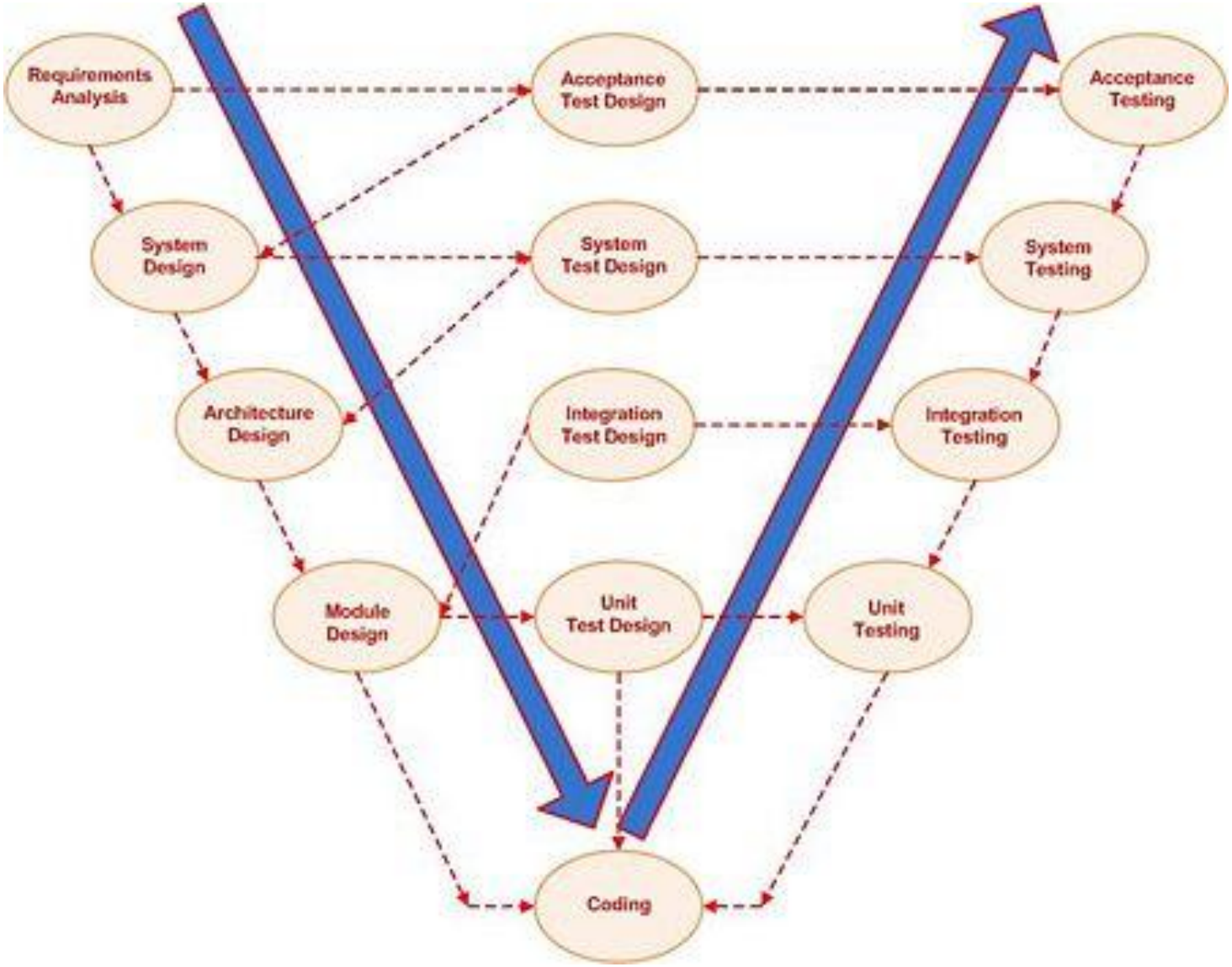
- It is the simplest software process model in terms of complexity and ease of implementation. As you know, it is nothing but common sense.
- This model is extremely easy to understand and therefore, is implemented at various project management levels and in a number of fields (not just software development).
- It employs a systematic, orthodox method of project development and delivery.

Disadvantages

- All requirements must be known upfront
- Jumping back and forth between two or more phases is not possible.
- The next phase can be reached only after the previous one has been completed.
- Bugs and errors in the code cannot be discovered until and unless the testing phase is reached. This can lead to a lot of wastage of time and other precious resources.
- Not suitable for projects wherein the project requirements are dynamic or constantly changing.

V -Model

V-Model



Contd..

- A framework to describe the software development lifecycle activities from requirements specification to maintenance. The V-model illustrates how testing activities can be integrated into each phase of the software development lifecycle.
- The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.
- Testing helps to ensure that the work-products are being developed in the right way (verification) and that the product will meet the user needs (validation).

Verification

- Checks that the work-product meets the requirements set out for it.
- Verification helps to ensure that we are building the product in the right way.

Validation

- Checks that the behavior of the work-product matches the customer needs as defined for the project.
- Validation helps to ensure that we are building the right product as far as the users are concerned.

Contd..

V-Model Objectives:

- Minimization of Project Risks
- Improvement and Guarantee of Quality
- Reduction of Total Cost over the Entire Project and System Life Cycle
- Improvement of Communication between all Stakeholders

Contd..

Advantages

- The users of the V-Model participate in the development and maintenance of the V-Model.
- A change control board publicly maintains the V-Model.
- The change control board meets once a year and processes all received change requests on The V-Model.
- At each project start, the V-Model can be tailored into a specific project V-Model, because the V-Model is project independent.
- It provides concrete assistance on how to implement an activity and its work steps, defining explicitly the events needed to complete a work step: each activity schema contains instructions, recommendations and detailed explanations of the activity.

Limitations

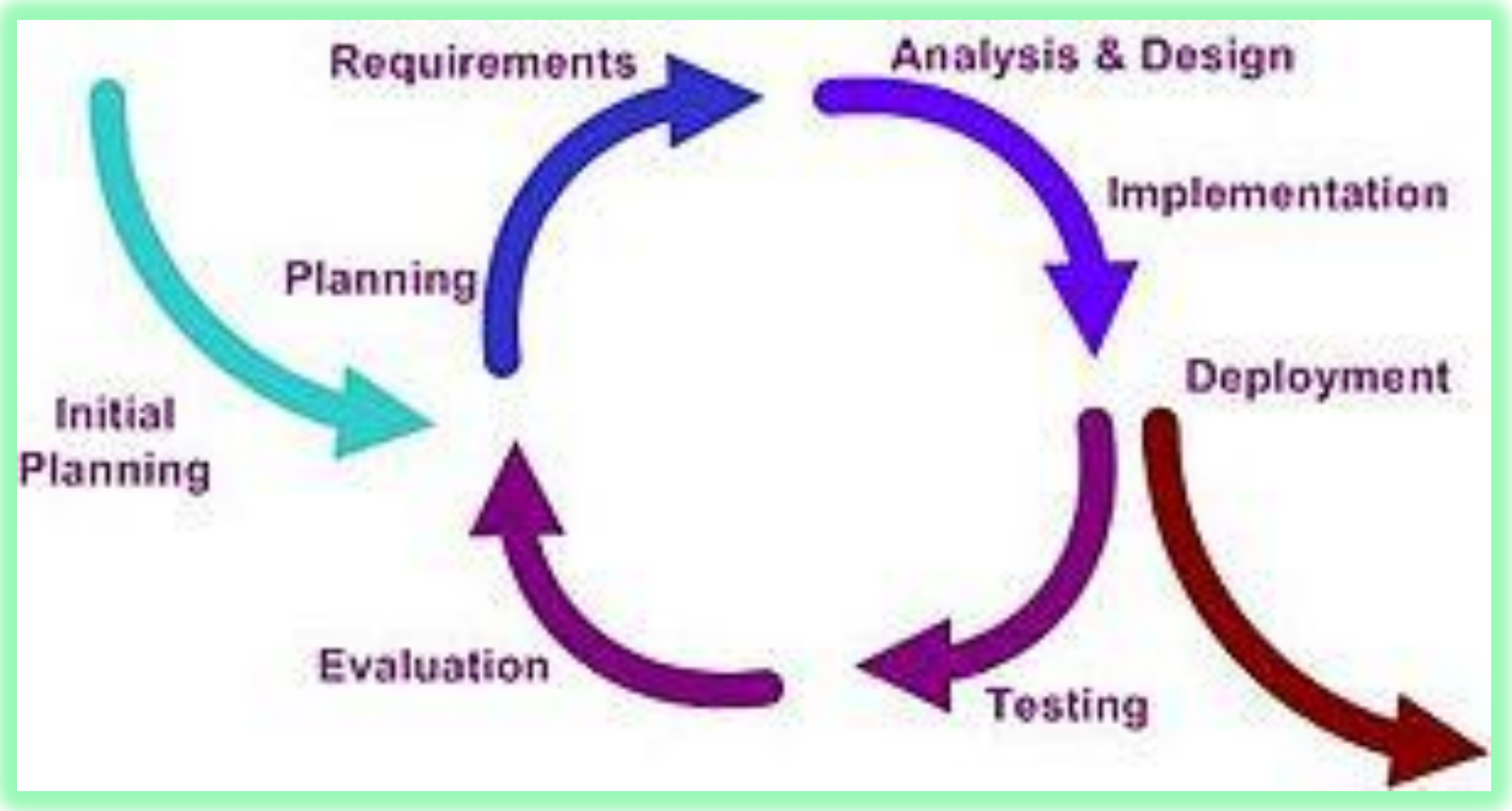
- The organization and execution of operation, maintenance, repair and disposal of the system are not covered by the V-Model. However, planning and preparation of a concept for these tasks are regulated in the V-Model.
- The V-Model addresses software development within a project rather than a whole organization.

Mahindra Satyam

Iterative-Incremental Model

Iterative-Incremental Model

- This is one where the requirements do not need to be fully defined before coding can start. Instead, a working version of the product is built, in a series of stages, or iterations - hence the name iterative or incremental development.



Contd..

- An Iterative model for development has fewer steps those are:
  - 1) Define Iteration Requirements
  - 2) Build Iteration
  - 3) Test Iteration
- This type of development is often referred to as **cyclical** - we go '**round the development cycle a number of times**', within the project.
- Example:
  - Assuming a development of a social networking website with parts like member registration, sign in, forgot password, member profile, search members, friends list, blog, blog search, photos, photo search and messaging.
- Development:
  - First iteration comprising of member registration, sign in, member profile and search members.
  - The second iteration can comprise of friends list, blog, blog search.
  - The third iteration can comprise of photos, photo search, messaging and forgot password.

Contd..

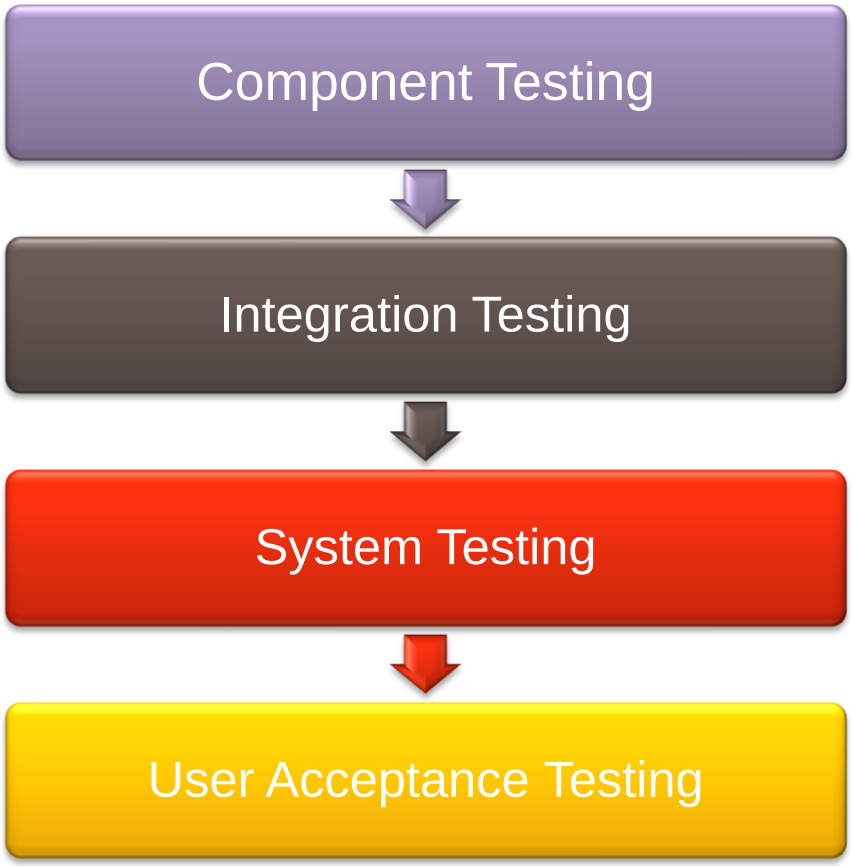
- **Drawbacks:**
  - The lack of formal documentation makes it difficult to test.
  - The working environment may be such that developers make any changes required, without formally recording them.
  - The amount of testing required to ensure that implementation of the changes does not cause unintended changes to other parts of the software (this is called regression testing).

Mahindra Satyam

Test Levels

Test Levels

- Test levels are group of test activities that are organized and managed together. A test level is linked to the responsibilities in a project.
- The typical levels of testing are:



Component Testing

The testing of individual software components is known as Component/Unit Testing.

Test basis:

- Component Requirements
- Detailed design
- Code

Typical test objects:

- Components
- Programs
- Data conversion/ migration programs
- Database modules

Component testing may include testing of functionality, and specific non-functional characteristics, such as:

- Resource-behavior (ex: searching for memory leaks)
- Robustness testing (testing with invalid inputs or stressful environmental conditions)
- Structural testing(ex: decision coverage)

Integration Testing

Mahindra Satyam

Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems is known as integration testing.

Test basis:

- Software and System design
- Architecture
- Workflows
- Use cases

Typical test objects:

- Subsystems
- Database implementation
- Infrastructure
- Interfaces, System configuration and configuration data

There may be more than one level of integration testing and it may be carried out on test objects of varying size as follows:

- Component integration testing tests the interactions between software components and is done after component testing.
- System integration testing tests the interactions between different systems or between hardware and software and may be done after system testing.

Contd..

- At each stage of integration, testers concentrate solely on the integration itself.
  - **Example:** If they are integrating module **A** with module **B** they are interested in testing the communication between the modules, not the functionality of the individual module as that was done during component testing.
- The greater the scope of integration, the more difficult it becomes to isolate defects to a specific component or system, which may lead to increased risk and additional time.
- There are three commonly quoted integration strategies, namely:
  - ✓ Big-Bang Integration
  - ✓ Top-Down Integration
  - ✓ Bottom-Up Integration

Contd..

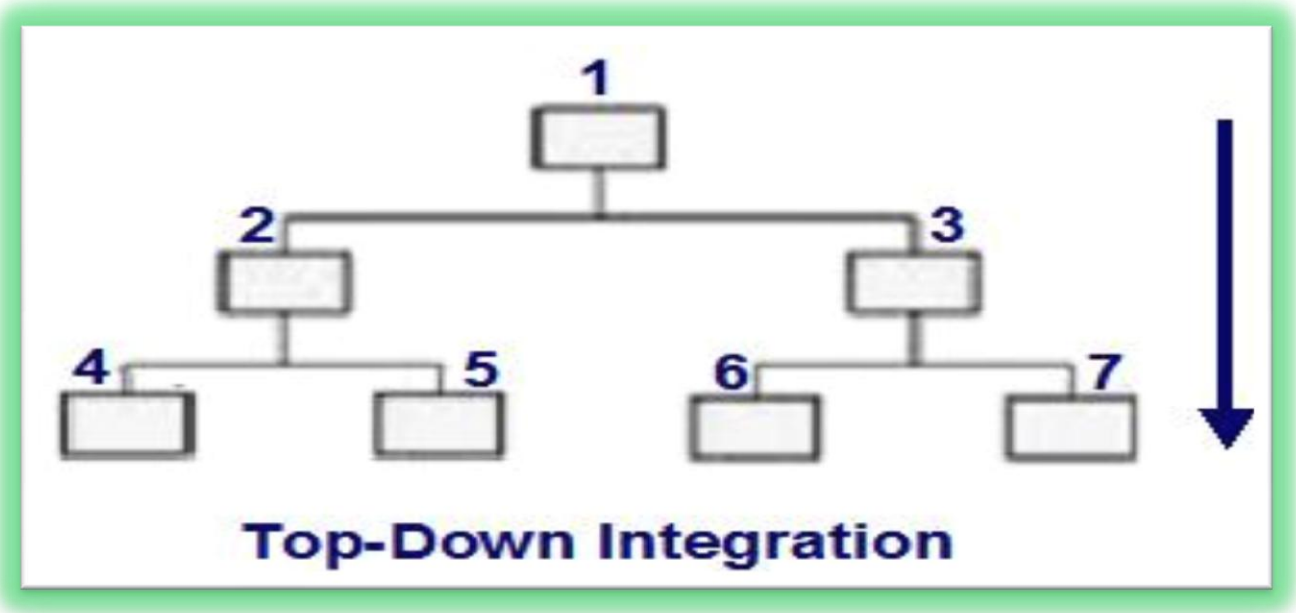
Big-Bang Integration

- A type of integration testing in which software elements, hardware elements, or both are combined all at once into a component or an overall system, rather than in stages.
- When testing this system, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.
- This type of integration is generally regarded as a poor choice of integration strategy. It introduces the risk that problems may be discovered late in the project, where they are more expensive to fix.

Contd..

Top-Down Integration

- An incremental approach to integration testing where the component at the top of the component hierarchy is tested first, with lower level components being simulated by stubs. Tested components are then used to test lower level components. The process is repeated until the lowest level components have been tested.



Contd..

- In the above figure,
  - Component 1 can call components 2 and 3. Thus in the structure, component 1 is placed above components 2 and 3.
  - Component 2 can call components 4 and 5.
  - Component 3 can call components 6 and 7.
  - In the structure, components 2 and 3 are placed above components 4 and 5 and components 6 and 7, respectively.

  Now the order of integration might be: 1,2 or 1,3 or 2,4 or 2,5 or 3,6 or 3,7

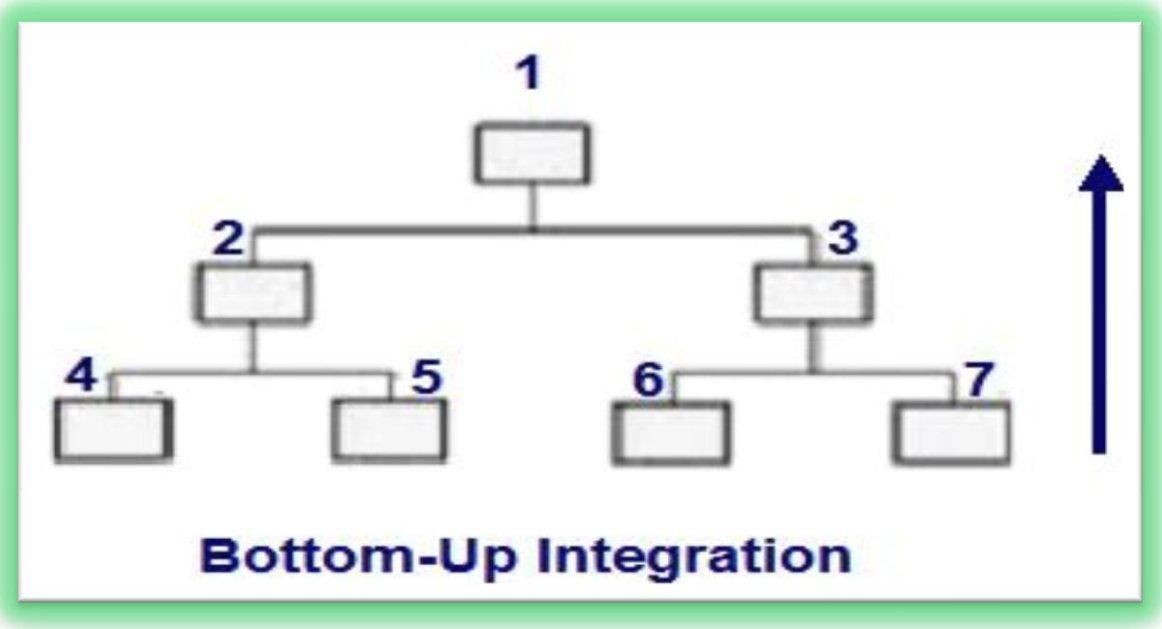
- In the Top-down integration testing the interactions of each component must be tested when it is built. If not…
  - This is done by creating a skeletal implementation of the component, called a **stub**.
  - **Stub:** A skeletal or special-purpose implementation of a software component, used to develop or test a component that calls or is otherwise dependent on it. It replaces a called component.

- In this example, stubs may be used to replace components 4 and 5, when testing component 2.

Contd..

Bottom-Up Integration

- An incremental approach to integration testing where the lowest level components are tested first, and then used to facilitate the testing of higher level components. This process is repeated until the component at the top of the hierarchy is tested.



Contd..

- As shown in following figure:

  The integration order might be: 4,2 or 5,2 or 6,3 or 7,3 or 2,1 or 3,1

  Components 4 - 7 would be integrated before components 2 and 3.

- In this case, the components that may not be in place are those that actively calls other components. When these special components call other components, they are called **drivers**.

- **Driver:** A software component or test tool that replaces a component that takes care of the control and/or the calling of a component or system.

- Components 2 and 3 could be replaced by drivers when testing components 4 – 7. They are generally more complex than stubs.

System Testing

The process of testing an integrated system to verify that it meets specified Requirements. [Hetzel]

Test basis:

- System and software requirement specification
- Use cases
- Functional specification
- Risk analysis reports

Typical test objects:

- System, user and operation manual
- System configuration and configuration data

System testing is concerned with the behavior of a whole system/product.

The system testing scope shall be clearly addressed in the Master Test Plan.

In System testing, the test environment should correspond to the final target or production environment.

An independent test team often carries out system testing.

User Acceptance Testing

- Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.

- **Test basis:**
  - User requirements
  - System requirements
  - Use cases
  - Business processes
  - Risk analysis reports
- **Typical test objects:**
  - Business processes on fully integrated system
  - Operational and maintenance processes
  - User procedures
  - Forms
  - Reports
  - Configuration data

Contd..

- The goal in acceptance testing is to establish confidence in the system, parts of the system or specific non-functional characteristics of the system.
- Finding defects is not the main focus in acceptance testing.
- Acceptance testing may assess the system's readiness for deployment and use.
- Acceptance testing is often the responsibility of the customers or users of a system.
- Acceptance testing may occur at various times in the life cycle, for example:
  - Before a COTS(commercial of the shelf) software installed/integrated.
  - Usability of a component during component testing.
  - New functional enhancements before system testing(BAT/BVT).

Contd..(Typical forms of Acceptance Testing)

Alpha Testing

- Simulated or actual operational testing by potential users/customers or an independent test team at the developers' site, but outside the development organization.
- Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing.

Beta/ Field Testing

- Operational testing by potential and/or existing users/customers at an external site not otherwise involved with the developers, to determine whether or not a component or system satisfies the user/customer needs and fits within the business processes.
- Beta testing is often employed as a form of external acceptance testing for off-the-shelf software in order to acquire feedback from the market.

Mahindra Satyam

Types of Testing

Black-box Testing

- Testing, either functional or non-functional, without reference to the internal structure of the component or system. ***-ISTQB***
- Black-box testing is also known as **specification-based testing.**
- There are two types of Black-box testing, those are:
  - Functional Testing
  - Non-Functional Testing

Functional Testing

- Testing based on an analysis of the specification of the functionality of a component or system. *-ISTQB*
  - Ex1: Searching for flights on a website
  - Ex2: Calculating employee pay correctly using a payroll system.

Non-Functional Testing

- Testing the attributes of a component or system that do not relate to functionality.
- e.g. reliability, efficiency, usability, maintainability and portability.

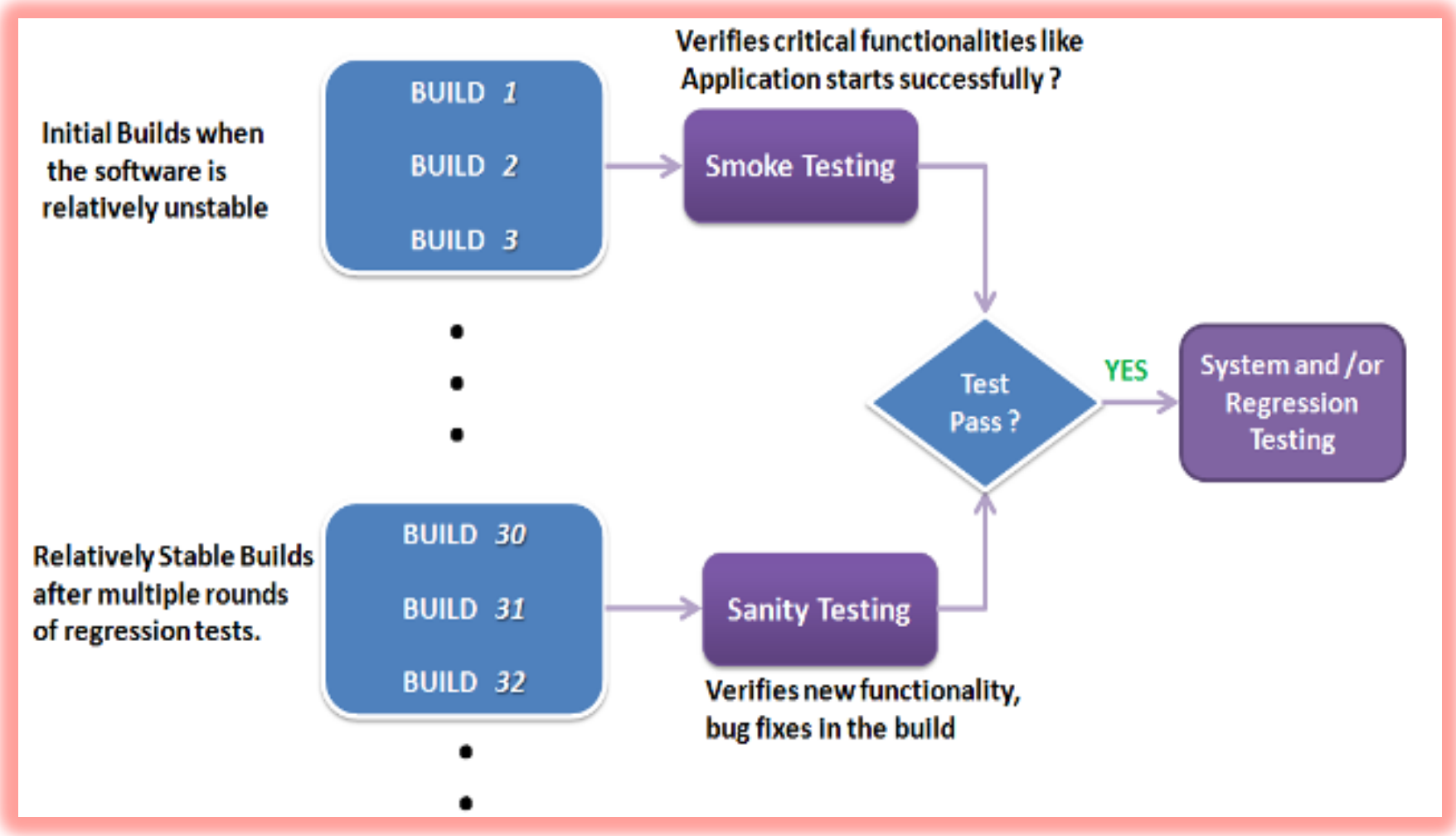
White-box Testing

- Testing based on an analysis of the internal structure of the component or system.
- White-box Testing is also known as **Structural Testing** or **Structure-based Testing** or **Glass-box Testing** or **Clear-box Testing**.

Gray-box Testing

- A combination of Black Box and White Box testing methodologies.
- Testing a piece of software against its specification but using some knowledge of its internal workings.
- It can be performed by either development or testing teams.

Smoke and Sanity Testing



Contd..

Smoke Testing

- Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine.
- The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing.
- This testing is performed by the developers or testers.
- Smoke testing is usually documented or scripted.
- Smoke testing exercises the entire system from end to end.
- Smoke testing is like General Health Check Up.

Sanity Testing

- Sanity Testing is done to check the new functionality/ bugs have been fixed.
- The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing.
- Sanity testing is usually performed by testers.
- Sanity testing is usually not documented and is unscripted.
- Sanity testing exercises only the particular component of the entire system.
- Sanity Testing is like specialized health check up.

Testing related to changes: Re-Testing and Regression Testing

- After a defect is detected and fixed the changed software should be retested to confirm that the problem has been successfully removed. This is called **retesting or confirmation testing.**
  - When the developer removes the defect, this activity is called debugging, which is not a testing activity. Testing finds a defect, debugging fixes it.
- The unchanged software should also be retested to ensure that no additional defects have been introduced as a result of changes to the software. This is called **regression testing**.
- Regression testing should also be carried out if the environment has changed.
- Regression tests need to be run many times over a testing project, when changes are made. This repetition of tests makes regression testing suitable for automation in many cases.

Maintenance Testing

- Testing that takes place on a system which is in operation in the live environment is called maintenance testing.
- After the deployment, it may become necessary to change the system. Changes may be due to the following reasons:
  - 1) Additional features being required.
  - 2) The system being migrated to a new operating platform.
  - 3) The system being retired - data may need to be migrated or archived.
  - 4) Planned upgrade to COTS(Commercial Off-The-Shelf) based systems(ready-made products).
  - 5) New faults being found requiring fixing (these can be ‘hot fixes’).

Thank you

mahindrasatyam.com

Safe Harbor

This document contains forward-looking statements within the meaning of Section 27A of the Securities Act of 1933, as amended, and Section 21E of the Securities Exchange Act of 1934, as amended. The forward-looking statements contained herein are subject to certain risks and uncertainties that could cause actual results to differ materially from those reflected in the forward-looking statements. Mahindra Satyam undertakes no duty to update any forward-looking statements.