Software Testing Methodologies

Introduction to Software Testing

AGENDA

SESSION 1	What is Software Testing
SESSION 2	Error and bug terminology
SESSION 3	Why we are doing Testing
SESSION 4	Benefits of Testing. Today's IT world and quality assurance / testing
SESSION 5	General principles of Testing
SESSION 6	Psychology of Testing
SESSION 7	How to perform the Testing
SESSION 8	Test Environment
SESSION 9	Types Of Software

AGENDA

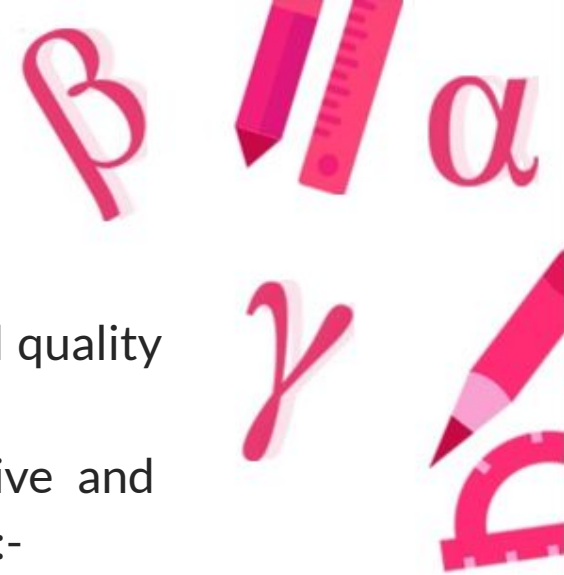
SESSION 10	Tester’s Tasks and Responsibilities
SESSION 11	SDLC
SESSION 12	Watefall Model, Iterative Model
SESSION 13	V Model

What is Software Testing?

Software – An application or set of instructions to perform some operation

Testing is a process used to help identify the quality of developed computer software. Software testing is performed to verify that the completed software package functions according to the expectations defined by the requirements/specifications. The overall objective to not to find every software bug that exists, but to uncover situations that could negatively impact the customer, usability and/or maintainability

What is Software Testing?



- It is the process used to identify the correctness, completeness and quality of developed computer software
- It is the process of executing a program/application under positive and negative conditions by manual or automated means. It checks for the :-
  - Specification
  - Functionality
  - Performance
- Check whether the software matches with the requirements, fit for use and the performance factors like speed, throughput etc. are satisfied

Definition

Correctness - defined as the adherence to the specifications that determine how users can interact with the software and how the software should behave when it is used correctly

Completeness – defined as how much of a program's source code has been tested i.e if the application has been tested with all the possible inputs and no part has been left untested

Error, Defect & Failure

A person makes an Error that creates a fault in software that further can cause a failure in operation

Error : An error is a human action that produces the incorrect result

Bug : The presence of error at the time of execution of the software

Fault : State of software caused by an error

Failure : Deviation of the software from its expected result. It is an event

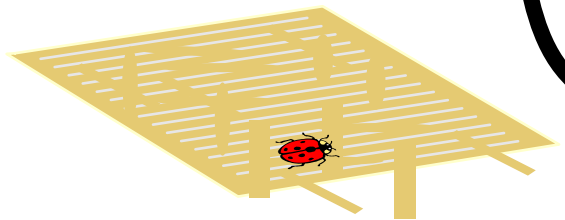
Error, Defect & Failure

- Human action producing an incorrect result: error/mistake
- Errors are inevitable
- Manifestation of error in software: defect
- Defects are also known as bugs
- Deviation of the software from its expected delivery or service: failure
- Failures are caused by defects in software

Error, Defect & Failure

A person makes an error ...

... that creates a fault in the software ...

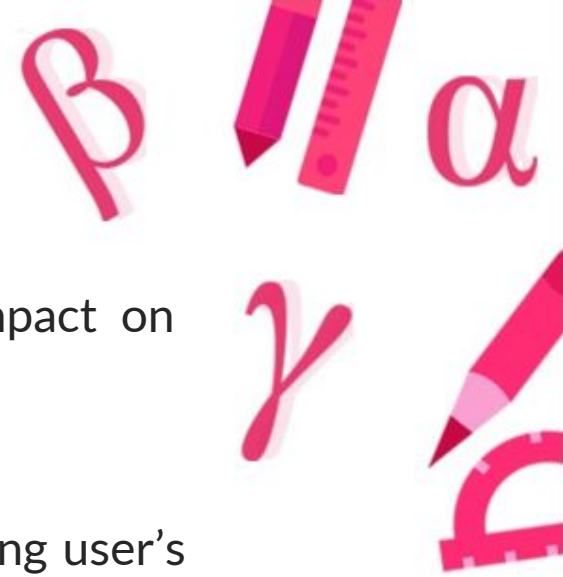


... that can cause a failure in operation

Why Software Testing ?

- **Software Testing** is important as it may cause mission failure, impact on operational performance and reliability if not done properly

- Effective software testing delivers quality software products satisfying user's requirements, needs and expectations



Objectives of testing

- The system is “ Fit for purpose”-

  Executing a program with the intent of finding an error

- To check if the system meets the requirements and be executed successfully in the Intended environment

- To check if the system does what it is expected to do.

Testing improves quality

- Finding defect and measuring quality in terms of defects
- Building confidence
- Preventing Defects
- Reducing risks

Who is a Software Tester???

Software Tester is the one who performs testing and find bugs, if they exist in the application under test

Qualities of a good tester…

Any tester should carry following attitude during testing:

- Curious to know about things happening
- Cautious to reach any conclusions
- Critical not to get easily satisfied by work done

Why do we need a tester?

- Developers are not good testers
- Software is not defect free
- Unreliable software can cause failures
- Failures have associated costs like loss of business
- Testing helps to find defects and learn about reliability of software
- Defects are less costly to correct the early they are found
- Helps reduce risk of problems occurring in an operational environment
- Can contribute to quality of software system

Causes of Software Defects

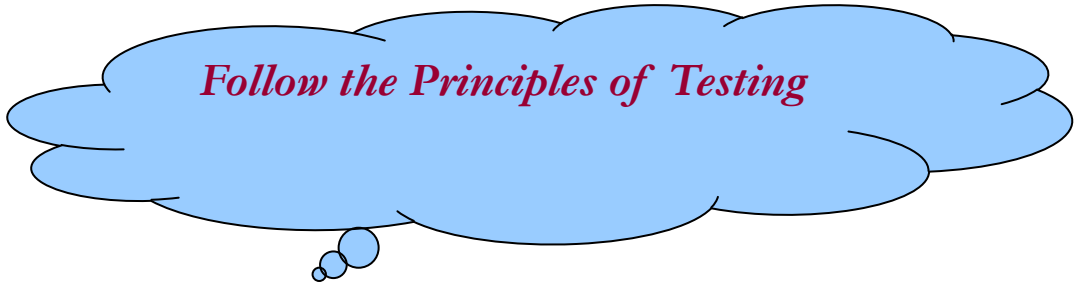
- Communication
- Software complexity
- Programming errors
- Changing requirements
- Time pressures
- Egos
- Poorly documented code
- Improper use of tools
- Environmental conditions

Is it possible to test everything?

Exhaustive testing is not possible

- Possible inputs and outputs are very large
- Software specification is very subjective
- Too much independent path

The solution of this problem is?



Principles of Testing

- Testing should be done at optimum level which is neither under-testing nor over-testing

"Too little testing is crime – Too much testing is a sin"

General testing principles

- Testing shows presence of defect
- Exhaustive testing is impossible
- Early testing
- Defect clustering
- Pesticide paradox
- Testing is context dependent
- Absence-of-error fallcay

Principle 1

Testing shows presence of defects

- Testing can show that defects are present but can not prove there are no defects
- Testing reduce probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness

Principle 2

Exhaustive testing is impossible

- Testing everything (all combinations of inputs and preconditions)is not feasible except for small cases
- Instead of exhaustive testing, we use risk and priorities to focus testing efforts

Principle 3

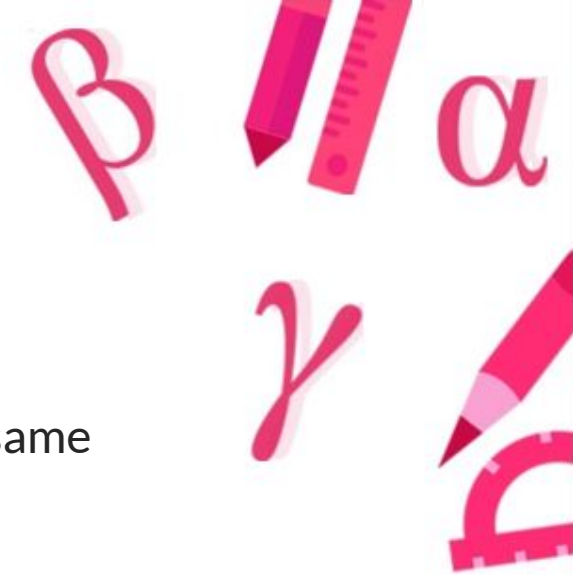
Early Testing

- Testing activities should start as early as possible in the software system development life cycle, and should be focused on defined objectives

Principle 4

Defect Clustering

- A small number of modules contain most of the defects discovered during pre-release testing



Principle 5

Pesticide Paradox

- If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new bugs
- To overcome this “pesticide paradox”, the test cases needs to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of software or system to potentially find more defects

Principle 6

Testing is context dependent

- Testing is done differently in different contexts
- For example, safety critical software is tested differently from ecommerce sites

Principle 7

Absence-of-errors fallacy

- Finding and fixing defects does not helps if the system built is unusable and does not fulfills users' needs and expectations

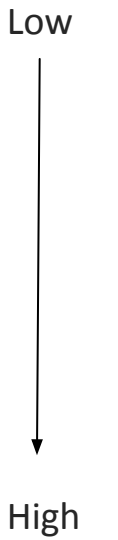
Psychology of testing

- The purpose of testing is to build confidence that the system is working
- But the purpose of testing is also to find defects
- Finding defect destroys confidence
- So the purpose of testing is to destroy confidence
- Paradox of testing-The best ways to build confidence is to try and destroy it

Level of independence

Independent testing is more effective

- Tests designed by person who wrote the software under test
- Test designed by another person
- Test designed by a person from a different organizational group
- Test designed by person from another organization



Benefits and drawbacks of independence

Benefits

- Unbiased and different view
- Verify assumptions during specification and implementation of the system

Drawbacks

- Isolation from development team
- Bottleneck as the last checkpoint
- Developers loose a sense of responsibility for quality

Who Should Test?

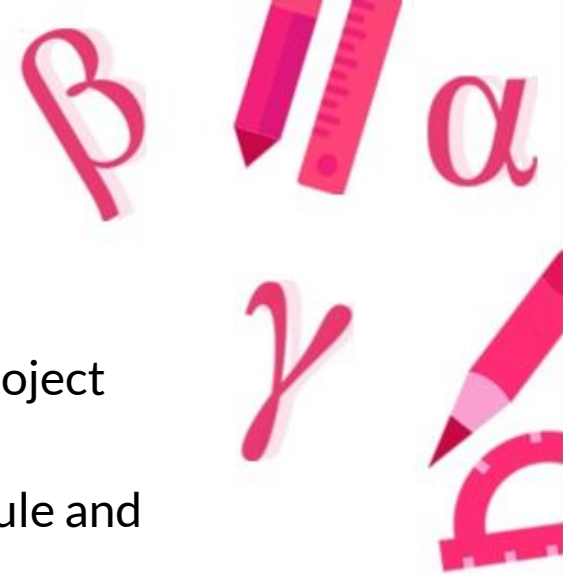
- Developer
  - Understands the system
  - But, will test gently
  - And, is driven by deadlines

- Independent tester
  - Must learn system
  - But, will attempt to break it
  - And, is driven by "quality"

Communication is important

- Testing generally bring “bad/unwanted news”
- Good interpersonal skills are needed to communicate factual information about defect, progress and risks in a constructive way
- Testers should maintain a good relationship with developers

The Testing Team



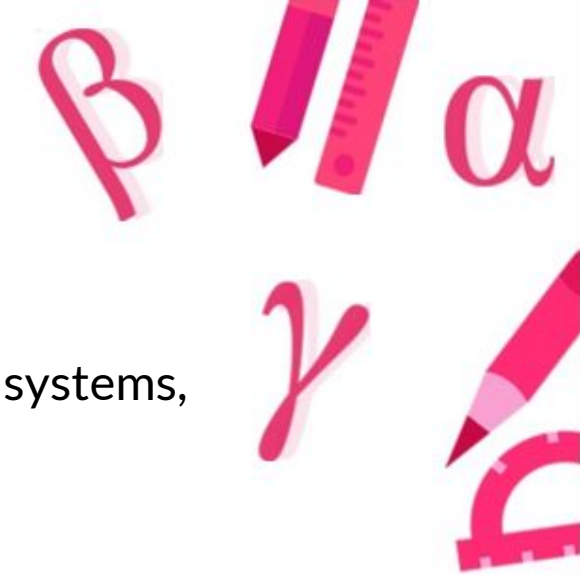
Program Manager

- The planning and execution of the project to ensure the success of a project minimizing risk throughout the lifetime of the project
- Responsible for writing the product specification, managing the schedule and making the critical decisions and trade-offs

QA Lead

- Coach and mentor other team members to help improve QA effectiveness
- Work with other department representatives to collaborate on joint projects and initiatives
- Implement industry best practices related to testing automation and to streamline the QA Department

The Testing Team (Continued)



Test Analyst\Lead

- Responsible for planning, developing and executing automated test systems, manual test plans and regressions test plans
- Identifying the Target Test Items to be evaluated by the test effort
- Defining the appropriate tests required and any associated test data
- Gathering and managing the Test Data
- Evaluating the outcome of each test cycle

Test Engineer

- Writing and executing test cases and Reporting defects
- Test engineers are also responsible for determining the best way a
- Test can be performed in order to achieve 100% test coverage of all components

Testing Environment

A testing environment is a setup of software and hardware on which the testing team is going to perform the testing of the newly built software product. This setup consists of the physical setup which includes hardware, and logical setup that includes Server Operating system, client operating system, database server, front end running environment, browser (if web application), IIS (version on server side) or any other software components required to run this software product. This testing setup is to be built on both the ends – i.e. the server and client

How much testing is enough?

Factors deciding how much to test

Level of Risk

- Technical Risk
- Business product Risk
- Project Risk

Project Constraints

- Time
- Budget

How much testing is enough?

Use Risk to decide where to place emphasis while testing

- Decide what to test first
- Decide what to test most
- Decide what not to test

“Prioritize tests” to do the best testing in available time.

SDLC(Software Development Life Cycle)

- Standard model used worldwide to develop a software
- A framework that describes the activities performed at each stage of a software development project
- Necessary to ensure the quality of the software
- Logical steps taken to develop a software product

Software Development Lifecycle

Overall process to develop systems through multistep processes:

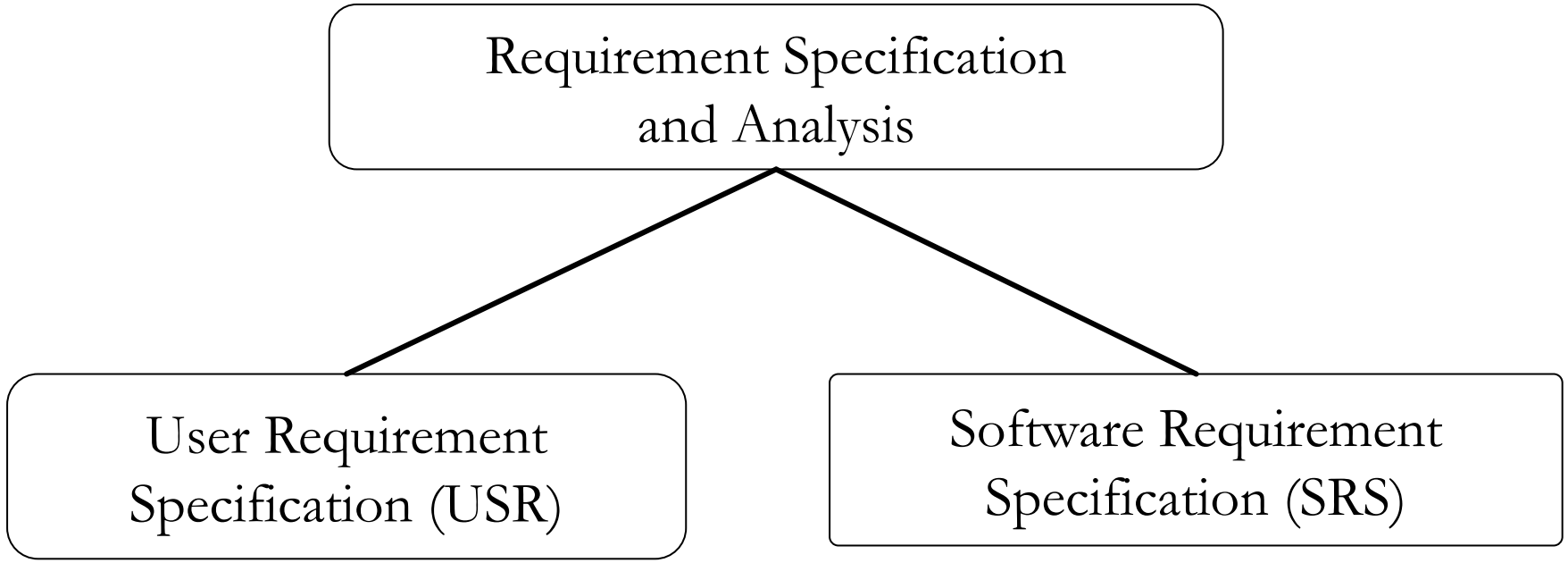
1. Business Requirement specification
2. Analysis (Functional Requirements)
3. Design
4. Coding
5. Testing
6. Deployment
7. Maintenance

Requirement Specification

- The process of writing in detail what the required from the software
- The business person writes the Business Requirements
- Requirement specification forms the basis of other documents in SDLC

Analysis

- Business requirement are further analyzed to form the functional requirements
- Functional requirement explain in detail what are the various functionalities of the application



Design

- Break down the bigger system into smaller systems
- Front end, back end and middle ware is designed
- Low Level Design and High Level Design is done
- Inputs and Outputs are identified

The output of SRS is the input of design phase

Two types of design -

- High Level Design (HLD)
- Low Level Design (LLD)

High Level Design (HLD)

- List of modules and a brief description of each module
- Brief functionality of each module
- Interface relationship among modules
- Dependencies between modules (if A exists, B exists etc.)
- Database tables identified along with key elements
- Overall architecture diagrams along with technology details

Low Level Design (LLD)

- Detailed functional logic of the module, in pseudo code
- Database tables, with all elements, including their type and size
- All interface details
- All dependency issues
- Error message listings
- Complete input and outputs for a module

Coding & Testing

Coding

- Developers build (code) the software

Testing

- Testers test the software

Deployment and Maintenance

Deployment

- Software is deployed

Maintenance

- Maintenance and enhancement of software

Maintenance

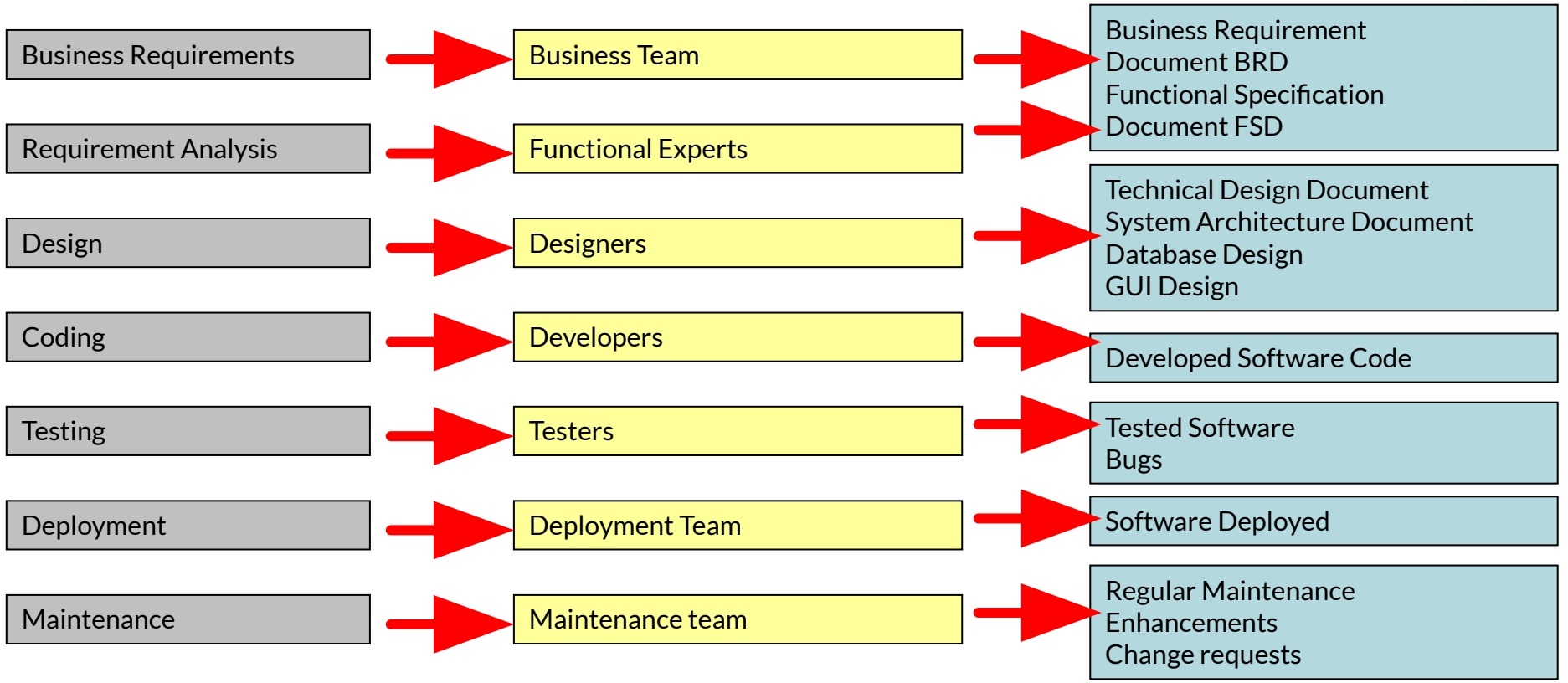
After the software is released and the client starts using the software, maintenance phase is started 3 things happen - Bug fixing, Upgrade, Enhancement

Bug fixing – bugs arrived due to some untested scenarios

Upgrade – Upgrading the application to the newer versions of the software

Enhancement - Adding some new features into the existing software

SDLC- Who does what?



When should S/W testing occur?

Throughout all phases-
Requirement phase

- Determine adequacy of requirements
- Generate functional test conditions

Design phase

- Determine consistency of design with requirements
- Determine adequacy of design
- Generate structural and functional test conditions

When should S/W testing occur?

Coding phase

- Determine consistency with design
- Determine adequacy of requirements
- Generate structural and functional test conditions

Test Phase

- Test application system

Deployment Phase

- Placed tested system into production

Maintenance Phase

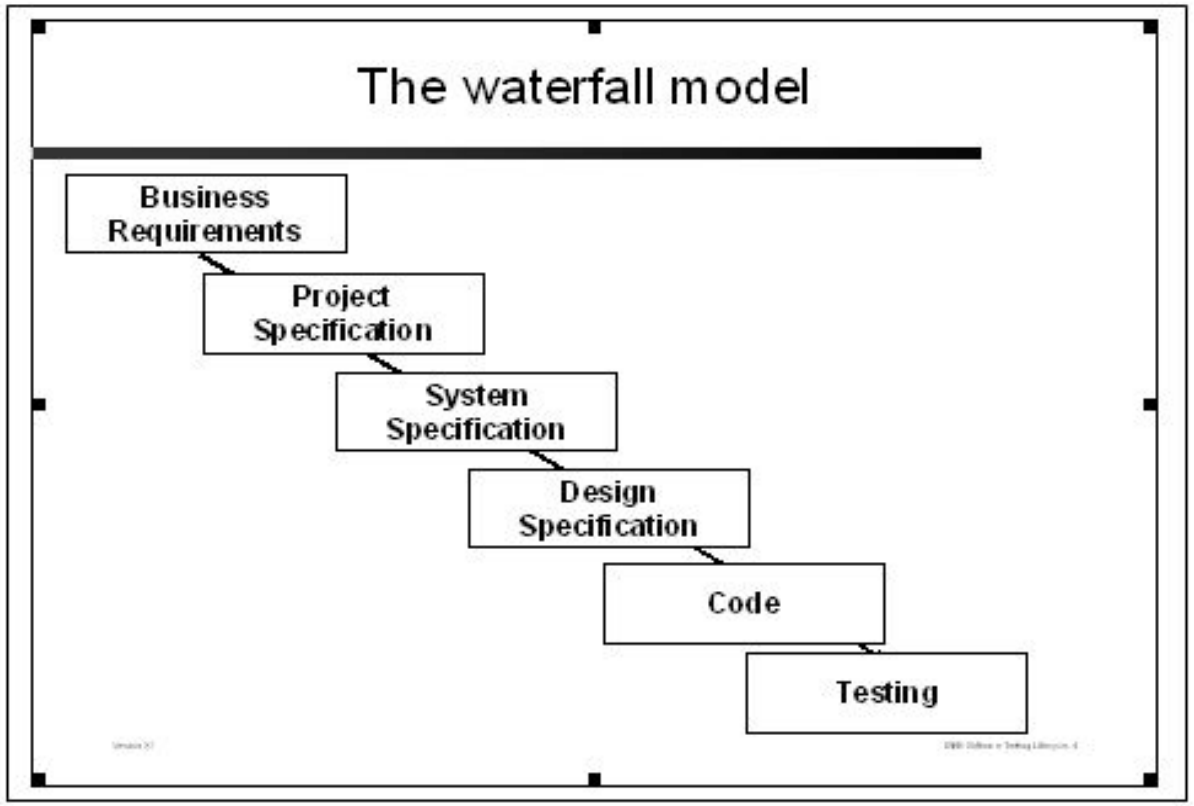
- Modify and test

Software Life Cycle Models

- WATERFALL MODEL
- V-PROCESS MODEL

Waterfall Model

Development broken into series of sequential stages
One of the earliest model developed for large projects
Also called linear sequential model



Waterfall Model

Features:

- Sequential steps (Phases)
- Feedback loops (Between two phases in development)
- Documentation-driven

When is it Suitable:

- Requirements are complete, stable and well understood
- Risk of delivery is minimal

Waterfall Model

Advantages

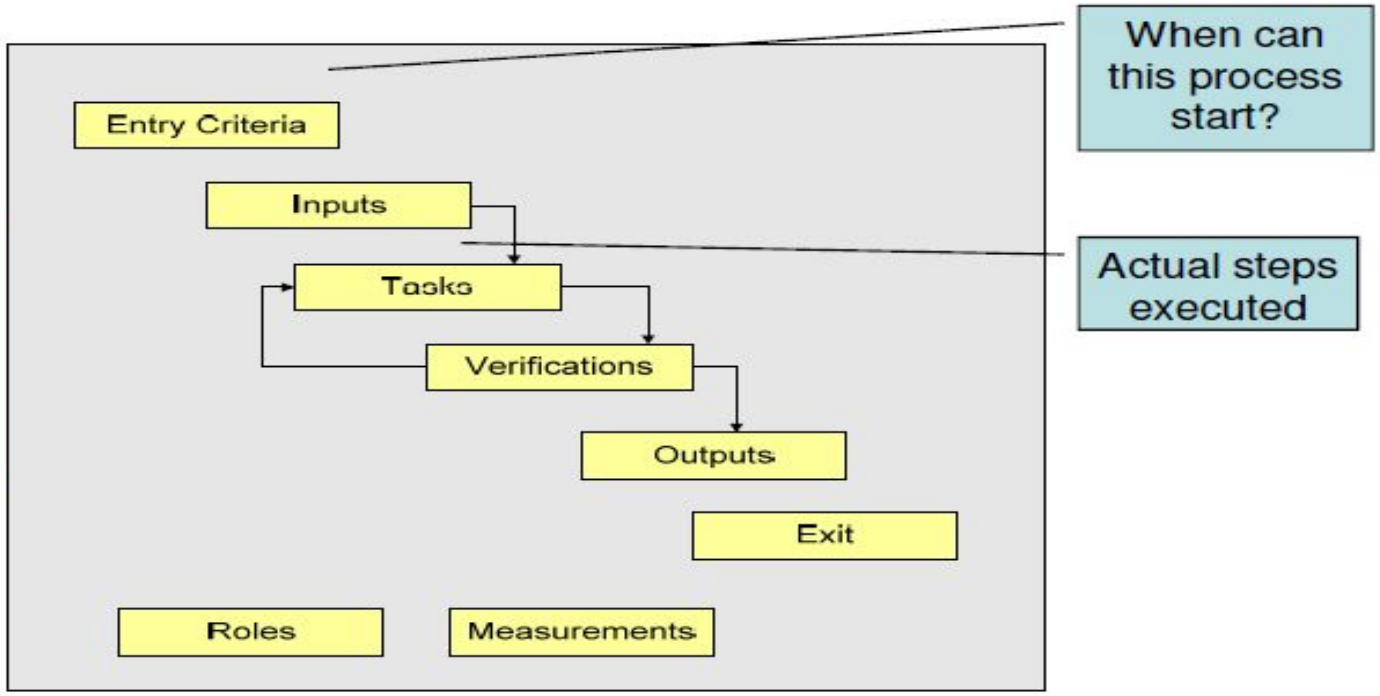
- Simplifies task scheduling as there are no Iterative or overlapping steps.

Drawbacks

- Testing is largely ignored until the end
- Mostly the finished product is delivered late to the testers. This causes time shortage for testers, may not cover testing fully
- Required Elaborated Documents of requirements.
- The customer must have patience

Entry and Exit criteria

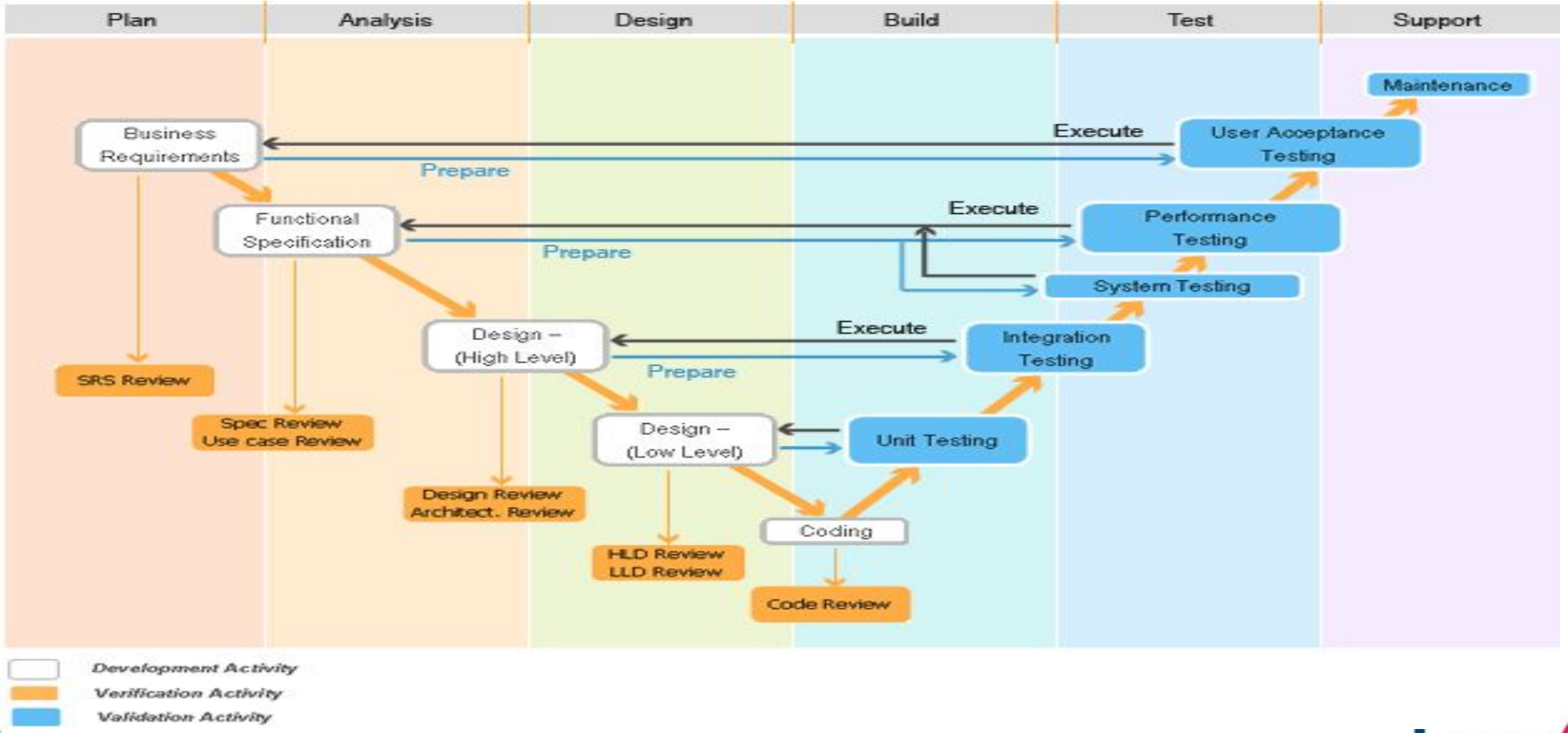
E=Entry criteria
T=Tasks
V=Verification
X=eXit criteria



Iterative development model

- Iterative development is process of establishing requirements, designing, building and testing a system, done as a series of smaller developments
- More realistic approach to development of large scale systems and software
- Each increment is fully tested before beginning the next
- Total planned testing effort will be more than that for waterfall model

V-Model


Plan
Analysis
Design
Build
Test
Support
Business Requirements
Functional Specification
Design – (High Level)
Design – (Low Level)
Coding
SRS Review
Spec Review
Use case Review
Design Review
Architect. Review
HLD Review
LLD Review
Code Review
Unit Testing
Integration Testing
System Testing
Performance Testing
User Acceptance Testing
Maintenance
Prepare
Execute
Development Activity
Verification Activity
Validation Activity

THANK YOU!