What is application?

Application is a combination of H/W and S/W to perform a specific task.

What is Software?

Software is a collection of programs or instructions that helps us to perform the task.

Types of software?

1. System software: that controls the working of the system (OS and server)
2. Programming software: Compiler, interpreters , debugger
3. Application Software: Games, applications in smart devices
4. Driver software - enables communication between OS and H/W or S/W.

Software Architecture:

Software Architecture is a system to represent the collection of components that accomplish a specific function or set of functions.

There may be one or multiple components in the foundation of an architecture on which software can be built. A software architecture helps to define and represent the component(s) and their relationship.

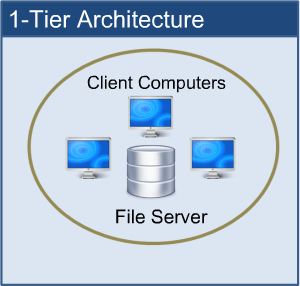
Example : Constructing a building with floor plan

Layers in software architecture:

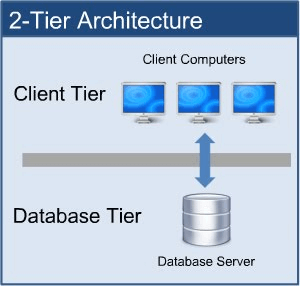
* **Presentation Layer:** This layer is responsible to display the user interface and manage user interaction.
* **Application Layer:**Application layer (also known as Business Layer) has all the business logic, rules and policies. The application layer is a bridge between the presentation layer and the data layer.
* **Data Layer:** This layer is responsible for storing the data.
* **Service Layer:** This layer is responsible to define and implement the service interface and the data contracts. Service layer communicates with the application layer

**Types of architecture:**

1. **One tier Architecture:** One-tier architecture has Presentation layer, Business layer and Data layers at the same tier i.e. at Client Tier. As the name suggested, all the layers and components are available on the same machine. MP3 player, MS Office etc.



1. **Two tier Architecture:** In such type of architecture, the client tier handles both Presentation and Application layers and the server handles the Database layer. The two-tier architecture is also known as a ‘Client-Server Application’. In two-tier architecture, communication takes place between the Client and the Server. Client system sends the request to the server system and the server system processes the request and sends the response back to the client system. Eg: Railway Ticket booking



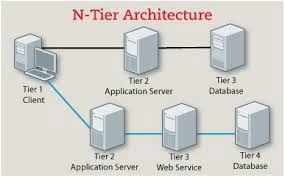
1. **Three tier Architecture:** All three major layers are separated from each other. Presentation layer resides at Client Tier, Application layer acts as middle-ware and lies at Business Tier and Data layer is available at Data Tier. This is a very common architecture.

Eg: GPay

### IMG_256

### ****4. N-tier Architecture:****

N-tier architecture is also called a Distributed Architecture or Multi-tier Architecture. It is similar to three-tier architecture but the number of the application server is increased and represented in individual tiers in order to distribute the business logic so that the logic can be distributed.



What is software testing?

**Software testing** is a process, to make sure the functionality and quality of a software application met with the specified requirements. It also ensures the software is defect-free.

Why software testing required?

1. Software testing is required to ensure the product is bug free.
2. meet the customer requirements.
3. Ensures the quality of the product or project.

**What is project?**

S/w application is developed for specific customer or client

requirement then it is Called project.

**What is product?**

S/w application is developed for multiple customers or client

requirement then it is called product.

**What is Quality of the software?**

**Software quality means:**

1. Bug free

2. Delivered on time

3. Within budget

4. Meet requirements

5. Meets expectation

6. Maintainable

**9) Software development Life cycle(SDLC)**

It is a process of developing a s/w through which sequence of phases.

SDLC produce high quality s/w.

**SDLC Phases:**

1) Requirement Gathering and Analysis

2) Designing

3) Developing

4) Testing

5) Deploy

6) Maintance

**10) SDLC Models:**

1) Waterfall model

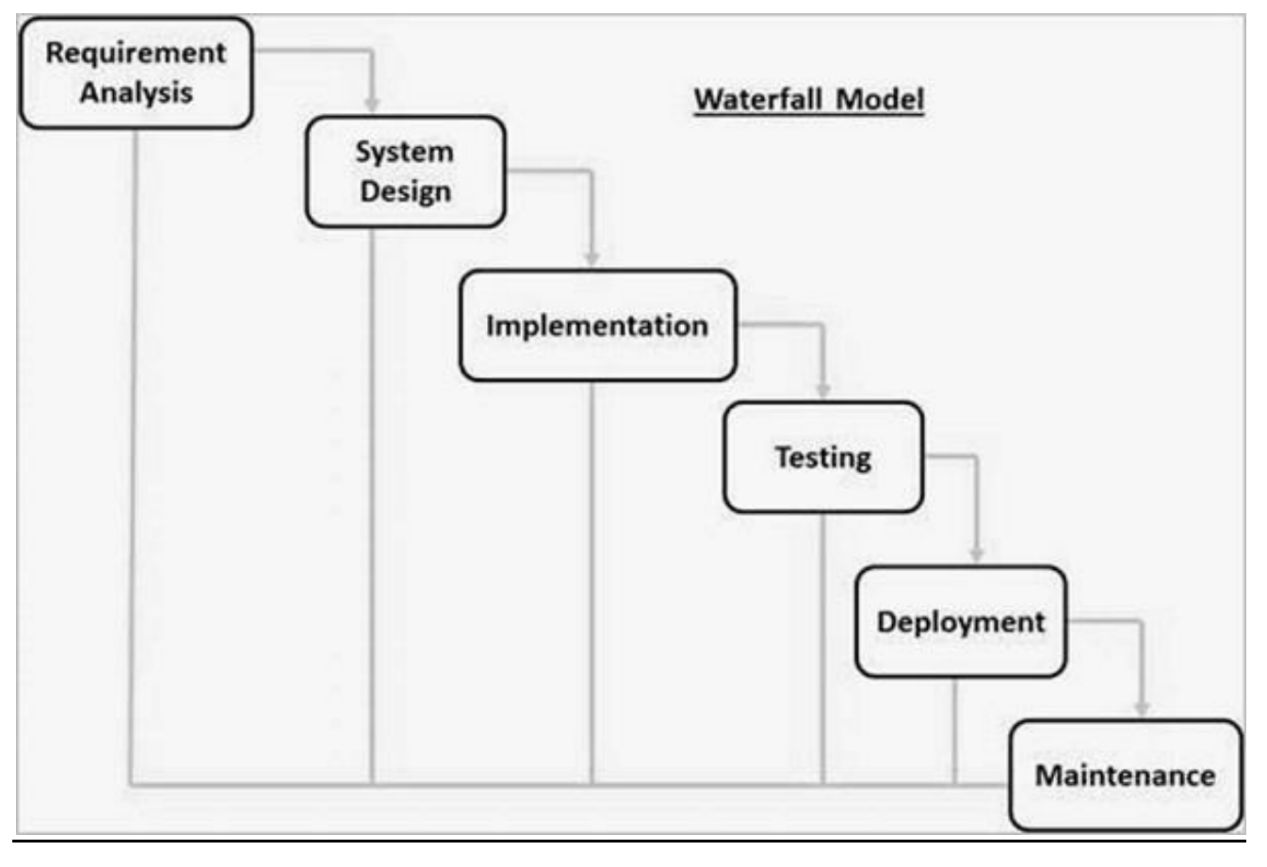
2) Incremental model

3) Spiral model

4) V –model

5) Agile model (95%)

**Waterfall Model: bike theft**



**Advantages of waterfall model**

• This model is simple and easy to understand and use.

• It is easy to manage due to the rigidity of the model – each phase

has specific deliverable and a review process.

• In this model phases are processed and completed one at a time.

Phases do not overlap.

• Waterfall model works well for smaller projects where

requirements are clearly defined and very well understood.

**Disadvantages of waterfall model**

• Once an application is in the **testing** stage, it is very difficult to go

back and change something that was not well-thought out in the

concept stage.

• No working software is produced until late during the life cycle.

• High amounts of risk and uncertainty.

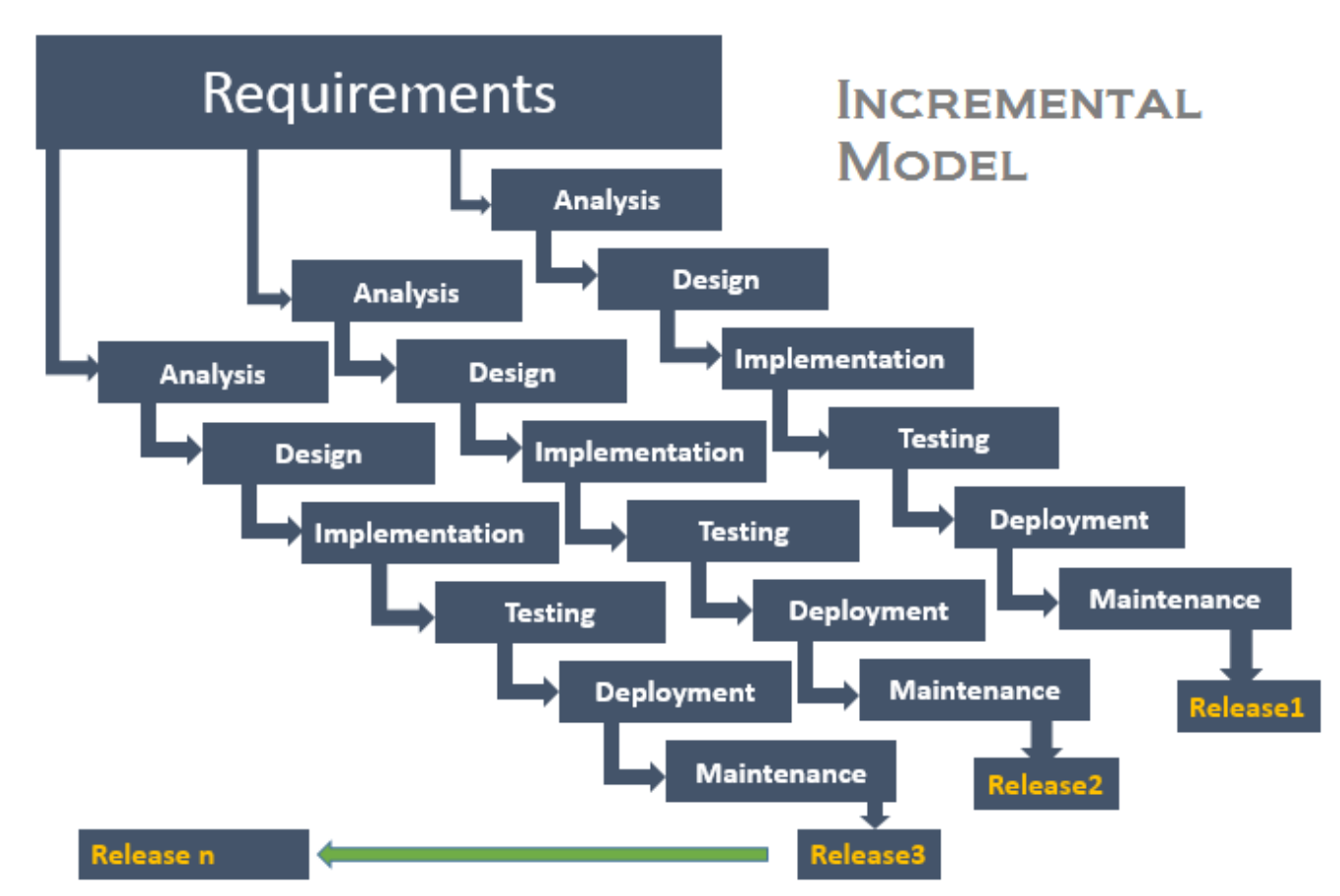
• Not a good model for complex and object-oriented projects.

• Poor model for long and ongoing projects.

• Not suitable for the projects where requirements are at a moderate

to high risk of changing.

1. **Incremental Model:**

 **Advantages of Incremental model:**

• Generates working software quickly and early during the software

life cycle.

• This model is more flexible – less costly to change scope and

requirements.

• It is easier to test and debug during a smaller iteration.

• In this model customer can respond to each built.

• Lowers initial delivery cost.

• Easier to manage risk because risky pieces are identified and

handled during it’d iteration.

**Disadvantages of Incremental model:**

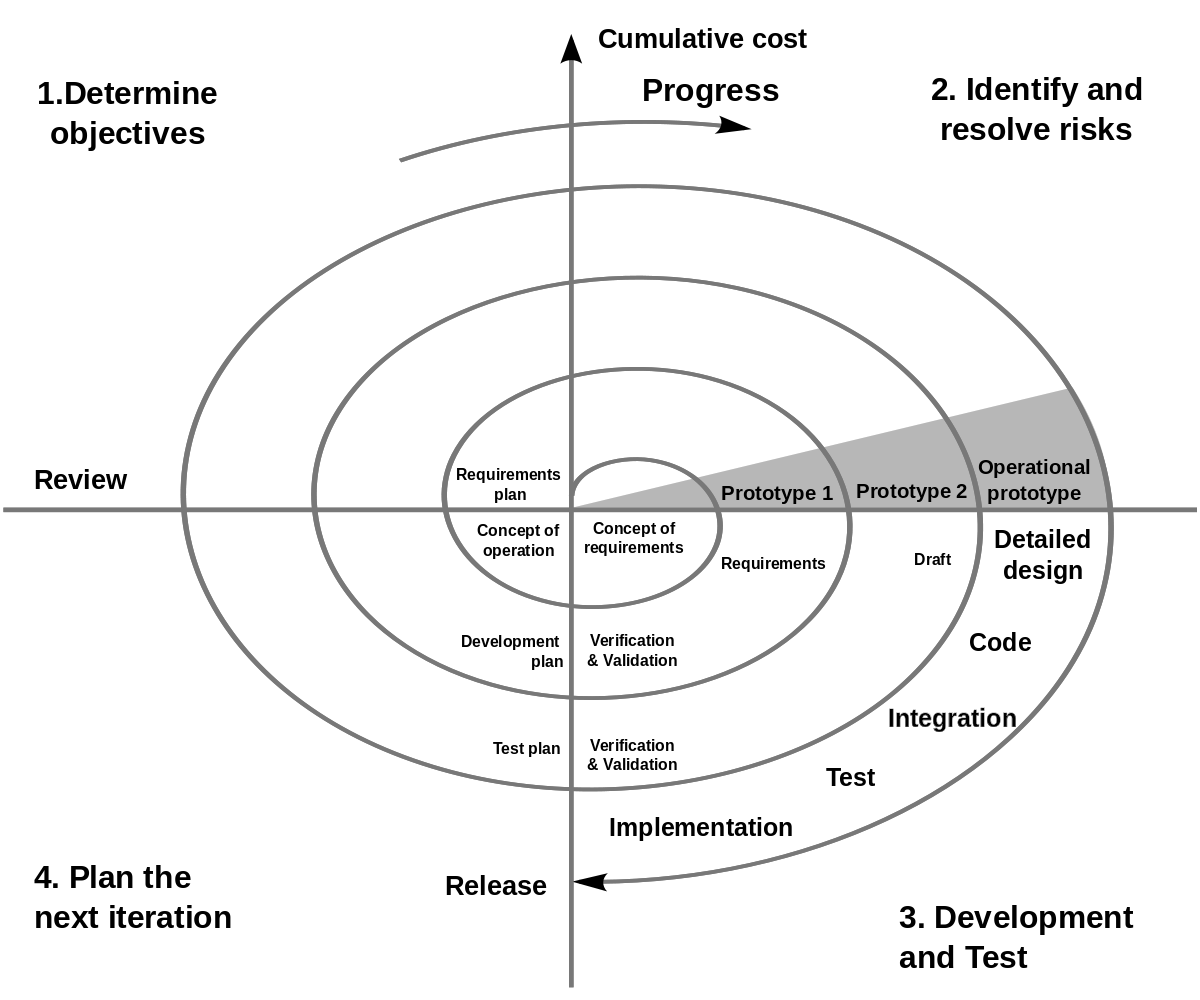
• Needs good planning and design.

• Needs a clear and complete definition of the whole system before

it can be broken down and built incrementally.

• Total cost is higher than **waterfall**.

1. **Sprial Model :**



**Advantages of Spiral model:**

• High amount of risk analysis hence, avoidance of Risk is enhanced.

• Good for large and mission-critical projects.

• Strong approval and documentation control.

• Additional Functionality can be added at a later date.

• Software is produced early in the **software life cycle**.

**Disadvantages of Spiral model:**

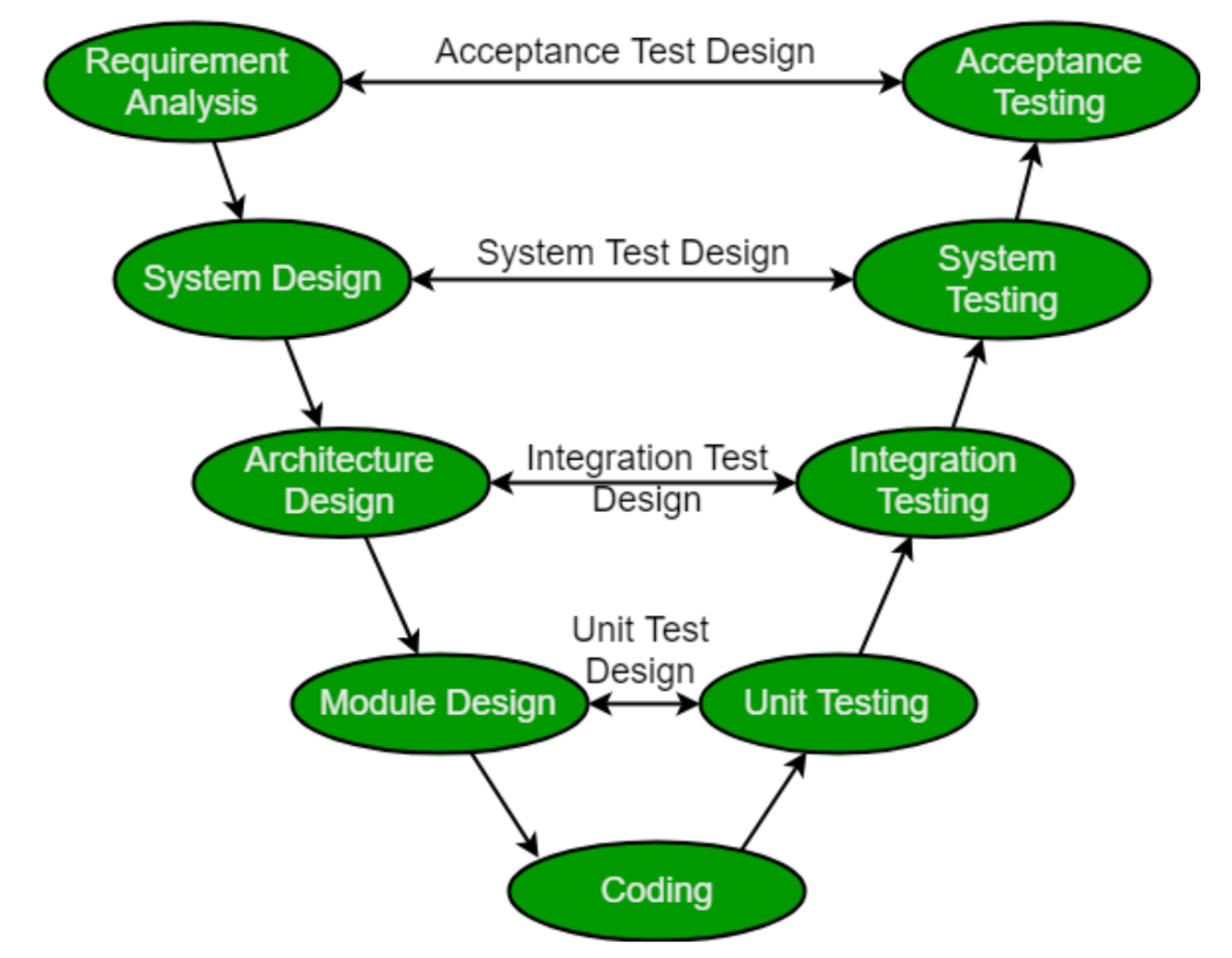
• Can be a costly model to use.

• Risk analysis requires highly specific expertise.

• Project’s success is highly dependent on the risk analysis phase.

• Doesn’t work well for smaller projects.

**4) V – Model:**



Verification(QA) Validation (QC)

(Management team, design, developer) (Developer, Tester)

**Advantages:**

• This is a highly disciplined model and Phases are completed one at

a time.

• V-Model is used for small projects where project requirements are

clear.

• Simple and easy to understand and use.• This model focuses on verification and validation activities early in

the life cycle thereby enhancing the probability of building an error

free and good quality product.

• It enables project management to track progress accurately.

**Disadvantages:**

• High risk and uncertainty.

• It is not a good for complex and object-oriented projects.

• It is not suitable for projects where requirements are not clear and

contains high risk of changing.

• This model does not support iteration of phases.

• It does not easily handle concurrent events.

**Design Phase for an Applications or S/W**

1) Web based application

2) Desktop app

3) Android application

4) SAP

**Design Phase:**

• **Requirement Analysis:** This phase contains detailed

communication with the customer to understand their

requirements and expectations. This stage is known as

Requirement Gathering.

• **System Design:** This phase contains the system design and the

complete hardware and communication setup for developing

product.

i) Front end design →UI or GUI (developer) high level design , low

level design

ii) Back end design → database (database developer)

• **Architectural Design:** System design is broken down further into

modules taking up different functionalities. The data transfer and communication between the internal modules and with the outside

world (other systems) is clearly understood.

• **Module Design:** In this phase the system breaks down into small

modules. The detailed design of modules is specified, also known as

Low-Level Design (LLD).

**Note:**Client -→Requirement Gathering -→ Module Design -→ Sub

Module -→ Functions

**software company teams and roles?**

Management Team -→ Business Analyst(BA), software Architect(SA), Project Manager(PM)

Designing Team → Team lead, designers

Developing Team→ Dev Lead, Developers

Testing Team →Team lead,testers

Operations Team or Supporting team

**Testing Environment**

Dev ------- Unit and Integration Testing

QA ------- System Testing

Staging--- UAT

Production---- Live

**Error, defect, bug, failure:**

**Error:** incorrect human action that produces a problem in the s/w is called error.

**Defect:** deviation from the expected result and actual result.

**Bug:** developer accepts your defect is valid that is bug.

**Failure:** bug identified by the client or end user in the s/w is called failure

**7) Why bugs in s/w?**

1. Miscommunication

2. s/w complexity

3. program errors

4. changing requirement

5. lack of skills

6. lack of time

**QA Vs QC (Quality Assurance Vs Quality Control)**

**QA:** Quality Assurance is defined as an activity that ensures the software application has no flaws.

1. The aim of quality assurance is to protect the defect.
2. It includes a complete software development cycle.
3. Quality assurance depicts benchmarks and methodologies to meet the client’s requirements.
4. It is a low level process.
5. This process is less time-consuming.
6. It is process related
7. Building the Quality preventing defects

iii) Verification

iv) Top Down approach

**QC:** Quality Control is an activity that ensures the quality of software applications.

1. This process aims to figure out the defects and then correct them.
2. It includes a complete software testing cycle.
3. Quality control ensures that the benchmarks are followed.
4. It is a high level process.
5. This process is more time-consuming.
6. QC is the actual testing of the s/w.
7. Testing the quality detecting defects

iii) Product oriented

iv) Validation

v) Bottom up approach

**Verification: (static testing)**

Checks whether we are building the right system.

1. Review meeting ( Requirement, Design, Code, Test Plan, Test Cases)
2. Walkthrough

iii) inspection

**i) Reviews:**

Conducts on documents to ensure correctness and completeness.

i) Requirement review (Management, design, code, testing)

ii) Design review (designers)

iii) Code review (developers)

iv) Test plan review(testers)

v) Test cases review(testers)

**ii) Walkthrough:**

It is a formal review we can discuss the issues. It happens any time.

**iii) Inspection**

It is formal approach 3-12 peoples will sit in the meeting Proper schedule which will be intimate via email.

**Validation :(Dynamic testing)**

Checks whether we are building the system right.

It is actual Testing

**Static Testing vs. Dynamic Testing:**

**Static Testing**: It is an approach to test project documents in the form

of reviews, inspection and walkthrough.

**Dynamic Testing**: it is an approach to test the actual software by giving

inputs and outputs

**Testing Techniques:**

1. White box Testing
2. Black box Testing
3. Grey box Testing(combination of white box and black box)

**1) What is white box testing?**

Testing the application with knowledge of programming skills.

Unit testing, integration testing done by developer.

**2) What is Black box Testing?**

Testing the application without knowledge of coding or programming skills.

To check functional behavior of the application It’s perform system testing and UAT testing done by tester.

**3) Grey box Testing:**

Combination of white box and black box testing techniques.

**Levels of Testing:**

1. Unit Testing ( CODE LEVEL CHECKING, DEVELOPER,WHITEBOX TESTING)
2. Integration Testing(CODE LEVEL CHECKING, DEVELOPER,WHITBOX TESTING)
3. System Testing (APPLICATION LEVEL CHECKING, TESTERS, BLACKBOX TESTING)

iv) User Acceptance Testing (APPLICATION LEVEL CHECKING, USER (OR) CLIENT (OR) TESTERS, BLACKBOX TESTING)

**1) Unit Testing:**

➔Small component to be tested

➔It is white box testing technique

➔It is conducted by the developer

To be tested in

1. Basic path
2. Control structures(conditional coverage, loop coverage)

iii) Mutation testing

**2) Integration Testing:**

➔Individual s/w modules are integrated logically and tested as a group

➔Checking communication flow and data flow one module to other modules

➔It is white box testing technique

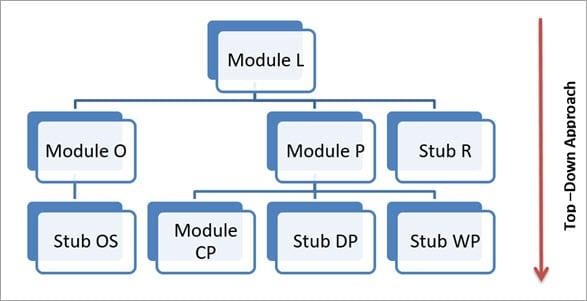
➔It is conducted by the developer

**Integration testing approaches:**

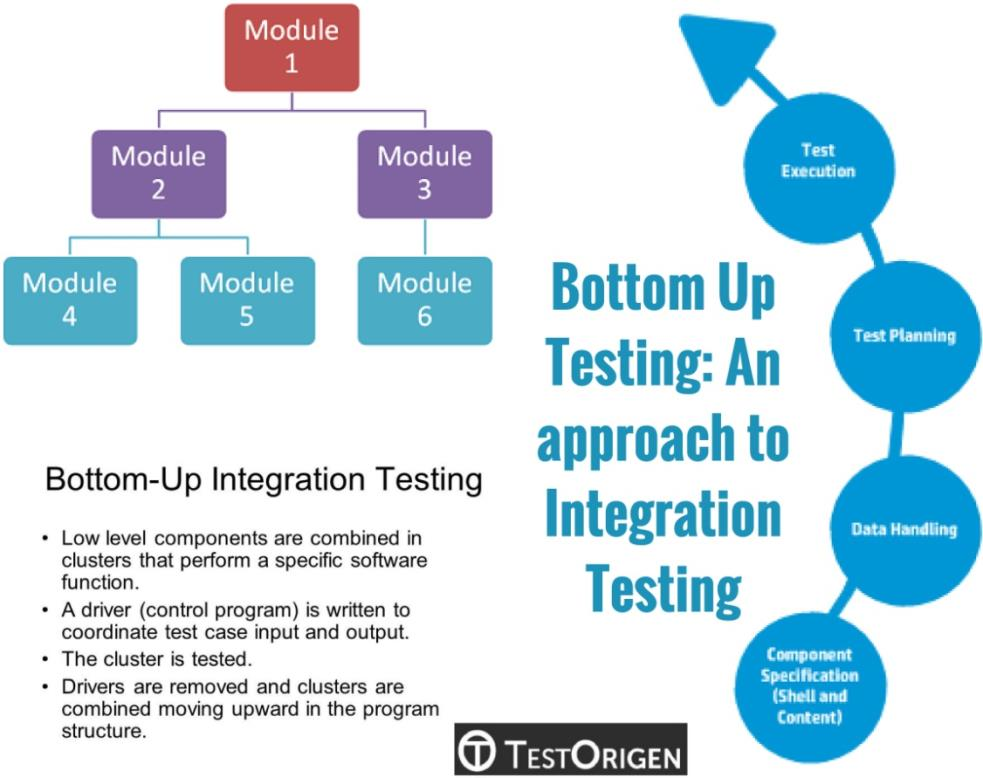
1. Big Bang
2. incremental Approaches or types:
3. Top down approach
4. Bottom up approach

iii) Sandwich

1. **Top down approach**



1. **Bottom up approach**



**Iii) Sandwich:** Combination of top down and bottom up

**Stubs and drivers:**

Stubs: dummy module used in top down approach

Drivers: dummy module used in bottom up approach

**3) System Testing:** → Testing over all functionality of the application with respective

client requirements.

➔Black box testing technique

➔Conducted by testing team

**System Testing focuses on**

1. Graphical User interface testing
2. Functional testing

iii) System Integration Testing

iv) Nonfunctional testing

v) Usability testing

**i) GUI (Graphical user interface testing):**

➔It is the process of testing the system gui.

➔To check the screens, menus, buttons, icons, toolbar, menu,bar, dialog box, window, tabs, combo box, textbox, edit box,

radio button, tree view, progress bar, table, scroll bar, spell check

➔Tester validate the application look and feel, shortcut keys and navigations

GUI Objects:➔screens, menus, buttons, icons, toolbar, menu bar, dialog box,

window, tabs, combo box, textbox, edit box, radio button, tree

view, progress bar, table, scroll bar, spell check

1. It checks if all the basic elements are available in the page or not

2. Spelling of the object

3. Alignments

4. Mandatory fields

5. Background color and font size

6. Button alignment issues

7. Inconsistent space between labels or text boxes

8. Broken labels i.e. single line label getting displayed in two lines

9. Misalignment among text boxes, info icons, labels or dropdowns

10. Overlapping of fields

11. Incomplete fields

12. Data on the page is misaligned; some time-shifted upwards or downwards

13. In any browser, while selecting some action, the corresponding action is not happening

14. Re sizing not working as expected

15. Session expiry time either very short or very long for some browsers

16. Browser specific issues – Few fields are not editable after entering data in one browser but editable in another browser

**ii) Functional Testing:**→ Checking the functional behavior of the application by giving

inputs.

1. Input Domain coverage(BVA, ECP)
2. Error handling or exception handling
3. Calculation /Manipulation
4. Links execution
5. Cookies & session

**1. Input Domain coverage(Validate Techniques):**

→Boundary value Analysis

→Equivalence clause partitioning

**BVA:** LB -1 LB LB+1 UB-1 UB UB+1

4 5 6 99 100 101

4 → invalid value (abc)

5, 6→ valid value(abcd, abcde)

99,100 →valid values(adgh….99, asdf…100)

101 → invalid value (adgjh…101)

**ECP:** invalid partition valid partition invalid partition

1-3 7-98 102 – infinite

1 63 110

**Data’s:**

i) Blank data → Negative test case

ii) Valid data (BVA, ECP) → positive test case

iii) Invalid data(BVA, ECP) → Negative test case

iv) Illegal data→ Negative test case

**4. Error Handling**

o Validating error message thrown by the application when

we provide invalid data.

o The error message should be clear and easy to

understand to the user.

**5. Calculation /Manipulation Testing:**

o Validate mathematical calculations.

**6. Links Executions:**→Links existence → Links place in proper location

→Links executions → links is navigate to appropriate page

or not

**7. Cookies and Session:**

Cookies:

Temporary internet files which are created at client side

when we open the website

Session:

Sessions are time slots which are allowed to the user at the

server side

iii) **System Integration Testing:**

SIT Testing is to validate that all software module

dependencies are functionally correct and that data integrity is

maintained between separate modules for the entire solution.

**iv) Non Functional Testing:**

Checking performance of the Application.

**i) Performance Testing**

→ Load Testing

→ Stress Testing

→ Volume Testing

➔Load Testing:

Testing speed of the application while increasing the

load gradually till the customer expected number.

➔Stress Testing:

Testing speed of the application while

increasing/decreasing the load on the application to

check any where its breaking.

➔Volume Testing:

Check how much volumes of data are able to handle by

the system.

ii) **Security Testing:**

System protected from the unauthorized access.

Types:

Authentication

Access control

Encryption/Decryption

**iii) Compatibility Testing:** To check different os, h/w and s/w

**iv)Usability Testing:**

This testing validates application provides context sensitive help.

**4) UAT (user acceptance Testing):**

After completion of system testing UAT team conducts acceptance

testing in two levels.

i) Alpha testing → testing performed in developing environment

(developer ,client)

ii) Beta testing → testing performed in client environment

(developer,tester,client)

**Testing Terminology:**

i) Adhoc Testing

ii) Monkey Testing

iii) Smoke Testing

iv) Retesting

v) Sanity Testing

vi) Regression Testing

vii) Exploratory Testing

viii) End to End Testing

**Adhoc Testing:**

a) S/w testing performed without planning and

documentation.

b) Without SRS (Software requirement specification) tester to

test the application

**Monkey Testing:**

Test the functionality randomly without knowledge of application and

test cases is called monkey testing.

**Smoke Testing:**

It is initial testing to perform checking the build stability.

To check major functionalities

Done by the tester

**Retesting:**

Same Functions to be tested repetitively.

Executing the same test cases whether the bug is fixed or not

**Sanity testing:**

Build receiving with minor changes in the code or functionality due to

this changes any bug introduce in the build or not.

**Regression testing:**

New Functions added in the existing functions or bug fixes to introduce

any bugs in the existing functionalities or not.

**Exploratory Testing:**

Exploring the application and understanding the functionalities adding

or modifying the existing test cases for better testing is called

exploratory testing.

**End to End testing:**

Testing the overall functionalities of the application including the

integration among all the modules is called end to end testing.

**Testing Process:**

**Software Testing Life cycle:**

i) TM(Test Manager)→ TL(Test Lead)→Testers

Client -→ Business Analysis, -→ BRS (Business Requirement

specification)

BRS → Software Architect

(SRS & FRS) -→ (Software Requirement specification –testing)

(Functional requirement specification → developer)

**Can you explain Software Testing Life cycle?**

• Once received the SRS my Test Manager and Team lead

analysis the requirements.

• Your TM and TL will prepare the Test plan and RTM.

• As a tester I received the SRS and TP from my TL.

• Analysis the requirement and identified the Test Scenarios.

• As a tester will write the test cases.

• Review the Test cases.

• Once I received build from developing team I will executed

my test cases.

• Identify the defects.

• Performing Defect tracking process.

• Once bugs are fixed retesting and close all the bugs.

• Updating RTM.

• Team lead prepared Test summary report.

• Project or product will be release to the client.

**Testing Phases:**

i) Requirement analysis

ii) Test plan Design

iii) Test case Design

iv) Test Execution

v) Test log

vi) Defect Tracking process

vii) Test closure (As a tester RTM update, AS a Team Lead Test

summary report)

Requirement: Signup

UN: min 5 char max 15 char

First letter should be alpha and caps

Numeric values accepted

Special char and blank space not allowed

Error msg display “please enter the valid username”

Test Data Expected Result Actual Result

status

UN: Blank Error msg no error msg

Fail

UN: Admin it accepted as per expected result

pass

UN: admin Error msg its navigate to home page

fail

UN:123Admin Error msg

UN: @Admin Error msg

Un: Admin123 it accepted

Un: Admin@ Error msg

**Bug life cycle: (Defect tracking process)**

**Priority:**

How fastly bug has been fixed.

Client point of view

Priority (Highest, High, Medium, Low) (p1, p2, p3)

**Severity:**

How badly the bug affected the application or build.

Severity levels: (Block, Critical, Highest, High, Medium, Low)

**Defect Leakage:**

After project release to the client, if client identified any bug is called

Defect Leakage.

**Defect Clustering:**

Small module to identify most numbers of bugs is called defect

clustering.

**Defect Triage:**

Large number of defects and limited testers of verify them, defect

triage helps trying to get as many defects resolved based on defect

parameters like severity and priority.

**Defect parameters:**

Defect review

Defect assessment

Defect assignment

**RTM: (Requirement Traceability matrix)**

Mapping the requirements with test cases and defects.

Two types:

i) Requirement traceability matrix (forward traceability matrix)

ii) Defect traceability matrix (backward traceability matrix)

**Documents:**

SRS

Test plan

Test case

Bug report

RTM

**SRS Contents:**

i) Introduction

ii) Scope

iii) Objective

iv) Environment

v) Functional Requirement

vi) Non Functional Requirement

vii) Use case

viii) Energy flow diagram

ix) Summary

**Test plan Content:**

i) Scope

ii) Strategy

iii) Environment

iv) Control (Review meeting, Change Request meeting, Defect Review meeting)

v) Function to be tested

vi) Resources and responsibilities

vii) Deliverables (Bug report, updating RTM, Test Summary report)

viii) Entry criteria and Exit criteria

ix) Suspension criteria and resuspension criteria

x) Summary

**Test case Content:**

i) S.No

ii) Test Scenario

iii) Test Scenario id

iv) Test case id

v) Priority

vi) Condition

vii) Action

viii) Test Data

ix) Expected Result

x) Actual Result

xi) Status

**Bug report Content:**

i) Bug id

ii) Test Scenario id

iii) Test case id

iv) Bug description

v) Priority

vi) Severity

vii) Build version

viii) Release version

ix) Status

**RTM Content:**

i) S.no

ii) Module

iii) Sub Module

iv) Requirement id

v) Test Strategy

vi) Test Scenario

vii) Test Scenario id

viii) Test case id

ix) Defect id

x) Status

xi) Comments

xii) Release date

**Testing Strategy:**

i) GUI

ii) Functional

iii) System Integration

iv) Non Functional

**Testing Environments:**

i) Dev (developer)

ii) QA

iii) UAT

iv) Staging

v) Production

A test environment is a server that allows you to run the test cases you have defined.

The test environment includes more than just setting up a server to run tests on. It

also involves hardware and network configuration.

In other words, **a test environment enables you to create identical environments**

**every time you need to test your product.** It’s the most important tool for a testing

engineer in order to have confidence in the testing results.

**DEV (Development Environment)** – This is the environment the developers will

write the code in. Before I say anything about DEV I have to say this to software

testers: **never test in dev**. This is the golden rule. It is right next to don’t let the

developers run the test for QA. I did not add this as best practice since it is simply

the only approach that can be taken. The reason you cannot test in DEV is developers

are always checking in code and working on defects. They will do their unit tests,

but as a software tester you need a non-moving platform to verify if things are

working as expecting. On top of that you need to test in a production like

environment and dev will not offer the data or the set up to perform this normally.

In dev you also can have complexities with multiple projects being worked on in one

environment. A lot of times this is managed through branching and merging the

code. As developers are always checking in code and working on defects, it is

expected that the environment will be stable but not as stable as any of the other

environments.**QA (Testing/ Quality Assurance Environment)** – Once the developer performs the

unit test cases, the code will be moved into QA to start testing. Often you will have

a few environments for testing. For example you will have one set up for system

testing and another that is used for performance testing and yet another that is used

for user acceptance testing (UAT). This is caused by the unique needs for each type

of testing. The reason you cannot test in DEV is developers are always checking in

code and working on defects. You need a non-moving platform to verify if things

are working as expecting.

**STAGING** – As you move the code from one environment to another, you will use

the staging environment. It is not likely that you will ever need to use the staging

environment for software testing, unless you are performing pre-production

implementation testing in this environment.

**PROD (Production)** – This is the external application where it is available to the

client. Production should not be updated until the scheduled go live. If it will be

updated prior to go live escalate. No testing will take place in production prior to go

live. Production does not have to be available to clients to be impacting to the rest

of the system. If the code is live the integration impacts can be seen. So do not let

the code get moved to prod without completing the testing in QA.

**Test Metrics:**

i)

No of Requirements

ii)

Avg no of test cases written per requirement

iii) Total no of test cases written all requirements

iv) Total no of test cases executed

v)

No of test cases passed

vi) No of test case failed

vii) No of test case blockedviii) No of test case not executed

ix) Total no of defects identified

x)

Critical defects count

xi) High defect count

xii) Medium defect count

xiii) Low defect count

xiv) Customer defects

xv) No of defects found in UAT

**Agile process:**

**Agile** is a **software development methodology** to build

software incrementally using short iterations of 1 to 4 weeks so that

the **development** is aligned with the changing business needs.

Agile frame works:

i) Scrum,

ii) Lean,

iii) Kanban,

iv) XP

