Module 1

Introduction and Fundamental

Que – 1 what is Software Testing ?

Ans – Software testing is a process which is used to identify the correctness, completeness and quality of the developed software

– Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free.

Que – 2 what is Manual Testing ?

Ans – Manual testing is a type of software in which test case are executed manually by a tester without using any automated tools.

- Manual testing concepts dose not require knowledge of any testing tools. One of the software testing is "100% Automation is not possible". This makes Manual Testing is Vital.

Que – 3 what is Automation Testing ?

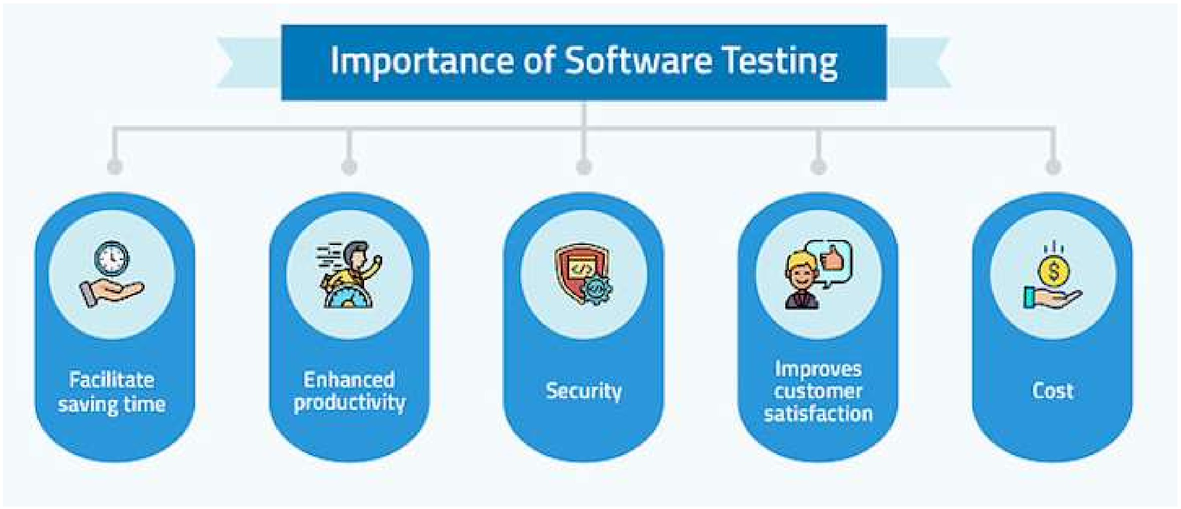
Ans – Automation testing is a type of software in which test case are executed by a tester with using of any types of automated tools Like QTP, Selenium etc.

Que – 4 why Software Testing is necessary ?

Ans – Testing is necessary because we all make mistakes. Some of those mistakes are unimportant, but some of them are expansive or dangerous. We need to check everything and anything we produce because things can always go wrong – human make mistakes all the time.

– Software Testing is important because if there are any bugs of errors in the software, it can be identified early and can be solved before delivery of the software product.

– If software testing is not performed then there will be a chance to lead the software failure.


Importance of Software Testing
Facilitate saving time
Enhanced productivity
Security
Improves customer satisfaction
Cost

<u>Que – 5 Testing Activities?</u>

<u>Ans –</u> 1) Planning and Control

2) Choosing test condition

3) Designing test Cases

4) Checking the results

5) Evaluating completion Criteria

6) Reporting the testing process

7) Finalizing and Closer

8) Reviewing of the documents

Que – 6 When to start Software Testing ?

Ans – Testing should be performed at every development stages of the product like Requirements gathering, Designing and Coding & Deployment. If testing is not performed on any of this stage, then the problems may change the whole structure.

Que –7 when to stop Software Testing ?

Ans - ~ All the high priority bugs are fixed

~ The rate at which bugs are found is too small.

~ The testing budget is exhausted.

~ The project duration is completed.

~ The risk in the project is under acceptable limit.

Que –8 Testing Objectives

Ans - 1) Finding defect

2) Preventing defects

3) Gaining confidence in and providing information about the level of quality.

4) Both dynamic testing and static testing can be used.

5) To preventing defects to be introduced into code to affect the whole project life cycle.

6) Review of the document throughout the life cycle

.

Que –8 7 Key Principles

Ans - 1) Testing shows the presence of Defects.

2) Exhaustive Testing is Impossible.

3) Early testing.

4) Defect Clustering.

5) The pesticide paradox.

6) Testing is context Dependent.

7) Absence of Error Fallacy.

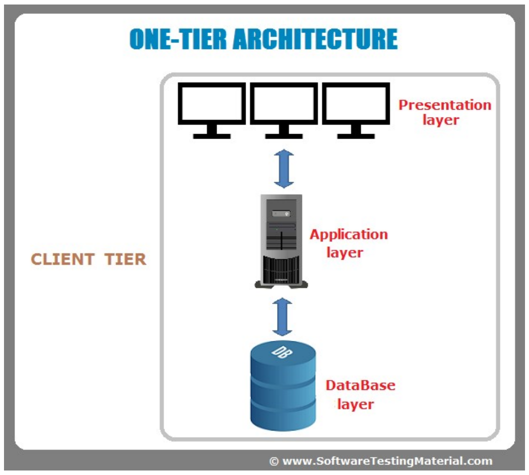
Que –9 Software Architecture

Ans - Software Architecture consists of One Tier, Two Tier, Three Tier, and N-Tier architectures.

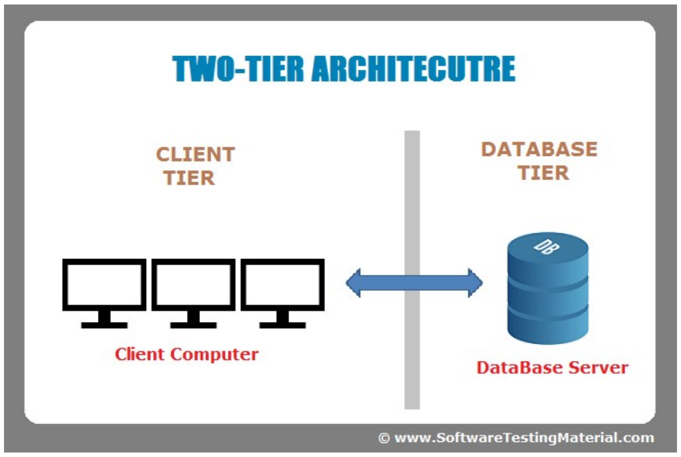
- A “tier” can also be referred to as a “layer”.

- Three layers are involved in the application namely Presentation Layer, Business Layer, and Data Layer

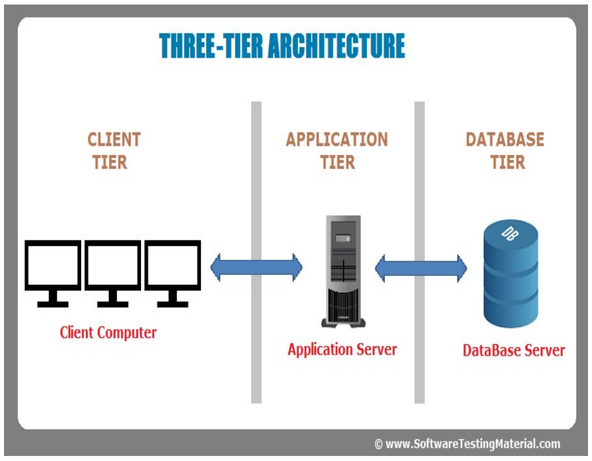
One – Tier Architecture

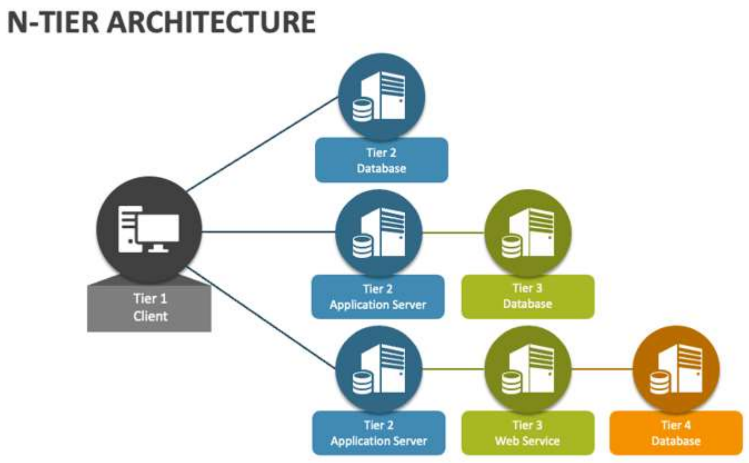


Two – Tier Architecture



Three – Tier Architecture





<u>Que –9 SDLC (Software Devlopment Life Cycle)</u>

Ans - SDLC is a structure imposed on the development of a software product that defines the process for planning, implementation, testing, deployment and ongoing maintenance and support.

- SDLC is a methodology or step-by-step approach to produce software with high quality, lowest cost in the shortest possible time by defining the phase like planning, analysis and design, coding and implementation, and testing and maintenance.

SDLC Phases :-

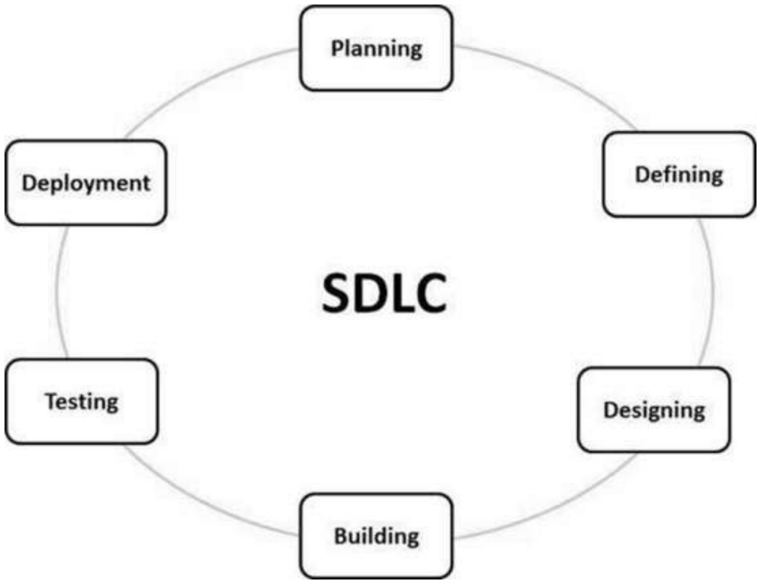
1. Requirement Collection

   Three types of problems arise

   1. lack of clarity
   2. Requirement confusion
   3. Requirement Amalgamation

2. Analysis
3. Design (low level Design & High level Design)
4. Implementation / Coding
5. Testing
6. Maintenance

   - Corrective maintenance
   - Adaptive maintenance
   - Perfective maintenance



Que –9 Software Development Methodoligies / Models : -

1) **Waterfall Model**
2) **V – Model**
3) **Iterative and Incremental Model**
4) **Spiral Model**

[1] Waterfall Model

“The waterfall Model is a classical software lifecycle that model the software development as a step by step “ waterfall “between the various development phase”

Application (when to use ?)

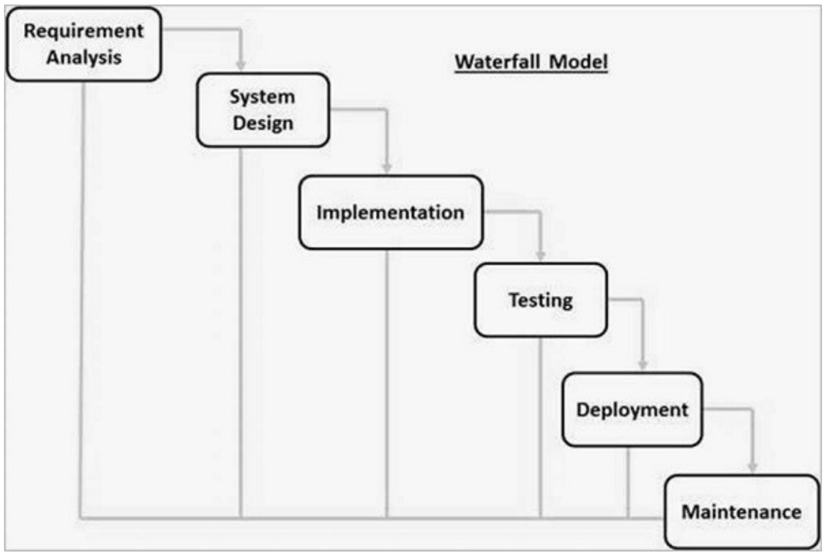
“The waterfall Model is a classical software lifecycle that model the software development as a step by step “waterfall“between the various development phase”

Pros:-

1. Simple and easy to understand and use.
2. Phase are processed and completed one at a time.
3. Work well for smaller projects where requirements are very well understood.
4. Process and results are well documented

Cons :-

1. Lack of flexibility (No changing requirement & no parallel work)
2. High amount of risk and uncertainty.
3. Poor model for long and on-going projects.
4. Not good model for complex & object oriented projects.



[2] V Model

The V- model is SDLC model where execution of processes happens in a sequential manner in V – shape. It is also known as Verification and Validation model.

Under V- model, the corresponding, testing phase of the development phase is planned in parallel.

V MODEL

Varification	Validation
Verification is a process which is performed at development level.	Validation is a process which is performed at testing level.
Verification Phases are: – Business Requirement Analysis – System Design / System Requirement – Architecture Design (Technical	Validation Phases are: – Unit Testing – Integration Testing – System Testing – Acceptance Testing

– Specification) – Module Design (program Specification	
It is the process of evaluating product of development to check whether the specified requirements meet or not.	It is the process of evaluating product of development to check whether it satisfied business requirements or not.
Verification can be achieved by asking "Are you building a product right ?	Validation can be achieved by asking "Are you building a right product ?
The evaluation of verification can be achieved by planning, Requirement Specification, Design, Specification, Code Specification, and test cases.	The evaluation of validation can be achieved as an actual product.
Verification activities are Reviews and Inspection	Validation activity is Testing.

Application (when to use ?)

- Requirements are well defined , clearly documented and fixed
- Product definition is stable.
- Technology is not dynamic and is well understood by the project team.
- There are no ambiguous or undefined requirements.
- The project is shorts.

Pros:

- Phases are completed one time.
- Work well for smaller projects where requirements are very well understood.
- Simple and easy to understand and use.
- Each phase has specification deliverables and a review process.

Cons:

1. Not a good model for complex and object-oriented projects.
2. Poor model for long and ongoing projects.
3. High risk of changing requirements.
4. Difficult to go back and change the functionalities.

[3] Iterative Incremental Model

"The basic idea behind this method is a to develop a system through repeated cycles (iterative) and in smaller portions at a time (Incremental)." Here, the development in always integrative and all activates progress in parallel.

Application (when to use ?)

- Requirements of the complete system are clearly defined and understood
- Major requirements must be defined.
- There is a time to market constraint.
- There are some high risk features and goals which may change in the future.
- A new technology is being used and is learnt by the development team while working on project.

Pros:

1. Results are obtained early and periodically.
2. Parallel development can be planned.
3. Progress can be measured.
4. Easier to manage risk – High risk part is done first.
5. Testing and debugging during smaller iteration is easy.
6. Facilitates the customer evaluation & feedback.
7. Module by module working.

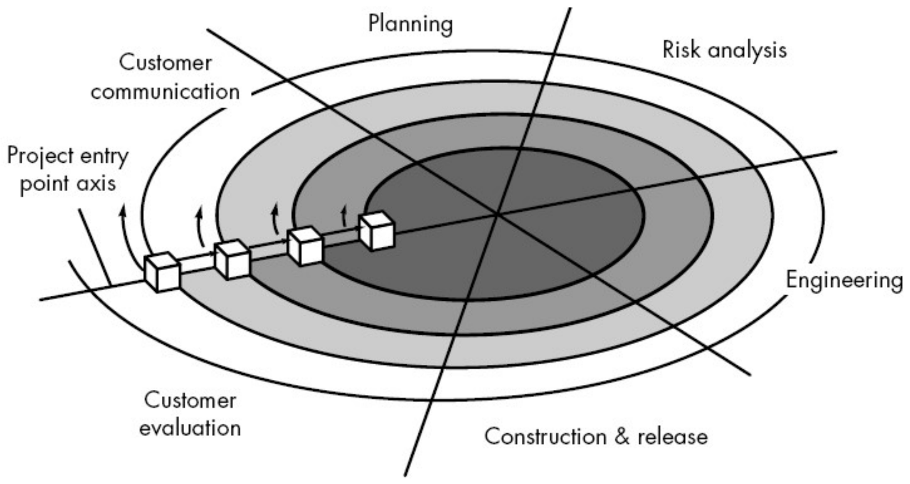
8. Less error changes.
9. Maximum user interactions.
10. Cost balancing (less costly)
11. Useful for early product release demand.
12. Useful for large product.
13. Flexible to change.
14. Risk analysis is better.

Cons:

1. More resources may be required.
2. More management attention is required.
3. Not suitable for smaller projects.
4. Highly skilled talent is required for risk analysis.
5. Project`s progress is highly dependent upon the risk analysis phase.
6. Management complexity is more.
7. End of project may not be known which a risk is.

[4] Spiral Model

“Spiral Model is very widely used in the software industry as it is in synch with the natural development process of any product learning with maturity also involves minimum risk for the customer as well as the development firms.”



A. Planning / Requirment Gathering / Feasibility Study
B. Risk Analysis / Design
C. Engineering / Coding
D. Customer Evaluation / Testing

Application (when to use ?)

- For medium to high – risk projects.
- Long term projects commitment.
- Customer is not sure of their requirements which are usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phase to get enough customer feedback.

Pros:

1. Highly flexible model.
2. Allows extensive use of prototypes.
3. Requirements can be captured more accurately.
4. User can see the software early.
5. Changing requirements is possible.
6. Monitoring is easy and effective.
7. Better risk management.

Cons:

1. Management is more complex.
2. End of project may not be known early.
3. Process is complex.
4. Not an ideal fir for smaller or low – risk projects
5. Spiral may go indefinitely.

[5] Agile Model

“Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.’’

Agile Mnifesto Principles / 4 values of Agile Manifesto

- **Individuals and Interaction** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over following a plan

Pros:

1. Very realistic approach
2. Rapid delivery.
3. Functionality can be developed rapidly
4. Resource requirements are minimum.
5. Little or no planning required
6. Promotes teamwork and cross training
7. suitable for fixed or changing requirements
8. Gives flexibility to developers

Cons:

1. More risk of sustainability, maintainability and extensibility.
2. Depend heavily on customer interaction.
3. Very high individual dependency.
4. Minimum documentation generated.
5. Not useful for small projects.
6. Not stables for hand handling complex dependencies.

Use Case:

A use case is the specification of the sequence of the action, including variants that a system can perform interacting with with the actors of the system.

Que –9 What is SRS (Software Requirement Specification)

SRS is a completed description of the behavior of the system to be developed.

Types of Requirements :-

- Customer Requirements
- Functional Requirements
- Non - Functional Requirements

Software Product:-

"A Software application that is developed by a company with its own budget."

Que – 10 What is 00PS :-

Object Oriented Programming is viewed as a collection of objects. It is used o structure the software program into simple reusable code. Here it is referred as Functional testing or Black Box Testing.

Que –11 What is Class :-

Class is a collection of a data member (variables) and member function with its behavior. Class is a blueprint or a template to describe the properties and behavior of the object.

Que –12 What is Object :-

An object is the basic unit of OPP which is accessed by its properties called data member and member function. Its creates the memory for the class.

Que –13 What is Encapsulation:-

A wrapping up of data and function into a single unit is called Encapsulation. It hide/include private access of data member & member function.

Abstractions:-

Abstraction is the representation of the essential feature of an object. Also called data hiding.

Que –14 What is Polymorphism:-

An ability to take one name having many different forms.

Compile time Polymorphism: (Operator Overloading).

Method name should be same in single class but its behavior (Arguments & Data type) is different.

Run time Polymorphism (Operator Overriding)

Method name should be same in Super class and sub class its behavior different.

Que –14 What is Inheritance:-

One class (Super, Base) inherits the properties of another class (Sub, Derived).

Types of Inheritance

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance
4. Hybrid Inheritance
5. Multiple Inheritance