OBJECT ORIENTED SOFTWARE ENGINEERING

IMPORTANT QUESTIONS

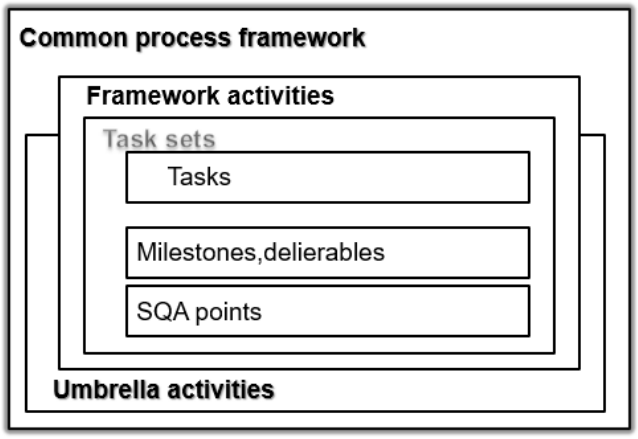
UNIT-1

1. Explain about process frame work with

UMBRELLA activities? Ans:

Process Framework:

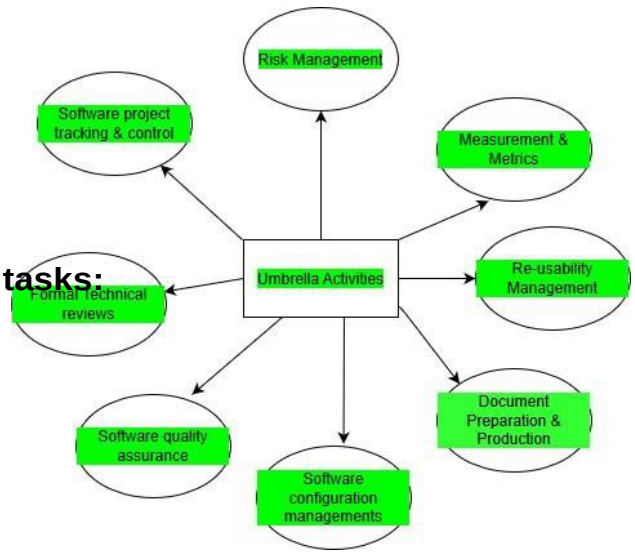
- Establishes the foundation for a complete software process
- Identifies a number of framework activities applicable to all software projects
- Also include a set of umbrella activities that are applicable across the entire software process.



Umbrella activities

Software engineering is a collection of interconnected phases. These steps are expressed or available in different ways in different software process models. Umbrella activities are a series of steps or procedures followed by a software development team to maintain the progress, quality, changes, and risks of complete development tasks. These steps of umbrella activities will evolve through the phases of the generic view of software development.

The activities in the software development process are supplemented by many general activities. Generally, common activities apply to the entire software project and help the software development team manage and track progress, quality, changes, and risks.



Umbrella activities consist of different tasks:

- Software Project Tracking and Control
- Formal Technical Reviews
- Software Quality Assurance
- SCM or Software configuration

management

- Document Preparation and Production
- Re-usability Management
- Measurement and Metrics
- Risk Management

Software project tracking and control: This activity allows the software team to check the progress of software development. Before the actual development starts, make a software development plan and develop on this basis, but after a certain period of time, it is necessary to analyze the development progress to find out what measures need to be taken. It must be accepted at an appropriate time after the completion of development, testing, etc. The test results may need to reschedule the development time.

Risk management: Risk management is a series of steps to help software development teams understand and manage uncertainty. It is a very good idea to identify it, assess the likelihood of it happening, assess its impact, and develop an "if the problem does happen" contingency plan.

Software quality assurance: As its name suggest this defines and conducts the activities required to ensure software quality. The quality of the software, such as user experience, performance, workload flexibility, etc., must be tested and verified after reaching the specified milestones, which reduces the tasks at the end of the development process, which must be performed by a dedicated team so that the development can continue.

Technical reviews: It assesses software engineering work products in an effort to uncover and remove errors before they are propagated to the next activity. Software engineering is done in clusters or modules, after completing each module, it is good practice to review the completed module to find out and remove errors so their propagation to the next module can be prevented.

Measurement: This includes all measurements of all aspects of the software project. Define andcompile process, project, and product metrics to help the team deliver software that meets the needs of stakeholders; it can be used in conjunction with all other frameworks and general operations.

Software configuration management: It manages the impact of changes throughout the software development process. Software Configuration Management (SCM) is a set of activities designedto manage changes by identifying work products that can be changed, establishing relationships between them, and defining mechanisms for managing different versions of them. Work product.

Reusability management: Define the standards for the reuse of work products (including software components), and develop mechanisms to implement reusable components. This includes the approval of any part of a backing-up software project or any type of support provided for updates or updates in the future. Update the software according to user/current time

requirements.

Work product preparation and production: It encompasses the activities required to create work products such as models, documents, logs, forms, and lists.

2.a) **Define Software Myth? Explain briefly about the types of myths?**

Ans:

SOFTWARE MYTHS:

- Widely held but false view
- Propagate misinformation and confusion
- Three types of myth
  - Management myth
  - Customer myth
  - Practitioner's myth

- Management myth

- Myth(1) -The available standards and procedures for software are enough.
- Myth(2) -Each organization feel that they have state-of-art software development tools since they have latest computer.
- Myth(3) -Adding more programmers when the work is behind schedule can catch up.
- Myth(4) -Outsourcing the software project to third party, we can relax and let that party build it.

- Customer myth

- Myth(1) - General statement of objective is enough to begin writing programs, the details can be filled in later.
- Myth(2) -Software is easy to change because software is flexible

- Practitioner's myth

Myth(1) -Once the program is written, the job has been done.

Myth(2) -Until the program is running, there is no way of assessing the quality.Myth(3) -The only deliverable work product is the working program

Myth(4) -Software Engineering creates voluminous and unnecessary documentation and invariably slows down software development.

b) **What is software Engineering and Explain the characteristics of software engineering? Ans:**

Definition of Engineering

-Application of science, tools and methods to find cost effective solution to problems

Definition of SOFTWARE ENGINEERING

- SE is defined as systematic, disciplined and quantifiable approach for the development, operation and maintenance of software.

Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and work efficiently on real machines.

Characteristics of Good Software

Every software must satisfy the following attributes:

- Operational
- Transitional

- Maintenance

Operational

This characteristic let us know about how well software works in the operations which can bemeasured on:

- Budget
- Efficiency
- Usability
- Dependability
- Correctness
- Functionality
- Safety
- Security

Transitional

This is an essential aspect when the software is moved from one platform to another:

- Interoperability
- Reusability
- Portability
- Adaptability

Maintenance

This aspect talks about how well software has the capabilities to adapt itself in the quickly changing environment:

- Flexibility
- Maintainability
- Modularity
- Scalability

3. Explain briefly on

***(a)* the incremental model**

Ans: THE INCREMENTAL PROCESS MODEL

- Linear sequential model is not suited for projects which are iterative in nature
- Incremental model suits such projects
- Used when initial requirements are reasonably well-defined and compelling need to provide limited functionality quickly
- Functionality expanded further in later releases
- Software is developed in increments
- Requirements of the system are clearly understood
- When demand for an early release of a product arises
- When software engineering team are not very well skilled or trained
- When high-risk features and goals are involved
- Such methodology is more in use for web application and product based companies

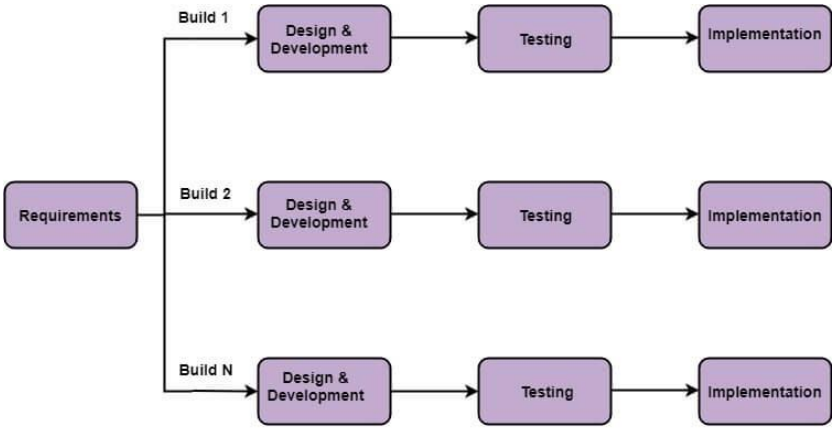


Fig: Incremental Model

***(b)* AGILE MODEL with neat diagram?**

Ans: 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 Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like −

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.

At the end of the iteration, a working product is displayed to the customer and important stakeholders.

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each buildis incremental in terms of features; the final build holds all the features required by the customer.

***4.a)* Illustrate about spiral model from evolutionary process model? Ans:**

THE SPIRAL MODEL

When to use Spiral Model?

- When deliverance is required to be frequent.
- When the project is large
- When requirements are unclear and complex
- When changes may require at any time

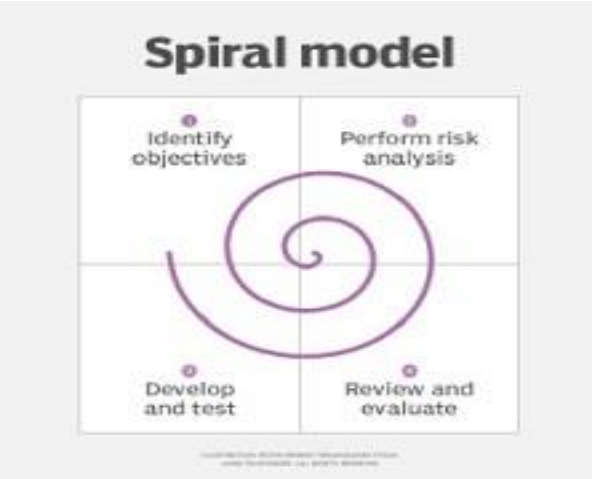
- Large and high budget projects

Advantages

- High amount of risk analysis
- Useful for large and mission-critical projects.

Disadvantages

- Can be a costly model to use.
- Risk analysis needed highly particular expertise
- Doesn't work well for smaller projects.



b) Distinguish between verification & validation? Ans:

Sr. No.	Key	Verification	Validation
1	Definition	Verification is the process in which product or system is evaluated in development phase to find out whether it meets the specified requirements or not.	On other hand Validation is the process in which product or system is evaluated at the end of the development process to determine whether software meets the customer expectations and requirements or not.
2	Objective	The main objective of Verification process is to make sure that the system being develop is as per the requirements and design specifications of the customer and if it deviates from it then make it correct in the development phase itself.	On other hand the objective of Validation is to make sure that the product which has been developed is actually meet up the user's requirements or not. And if it is not then make it to the level of acceptance in re development phase.
3	Activities	Main activities which defines the Verification process are Reviews of specification and product development, Meetings about diversification and inspections.	On other hand activities under Validation process are typically different type of testing such as Black Box testing, White Box testing, Grey box testing etc. which ensure

Sr. No.	Key	Verification	Validation
			the defect free delivery of product as per specification document.
4	Type	Verification is the process where execution of code is not take place and hence it comes under static testing.	On other hand during Validation execution of code take place and thus it comes under dynamic testing.
5	Sequence	Verification is carried out before the Validation.	On other hand Validation activity is carried out just after the Verification.
6	Performer	Verification is carried out by Quality assurance team.	On other hand Validation is executed on software code with the help of testing team.

***5.* What are the types of process models in software engineering and Explain about Water fall model with help of SDLC steps?**

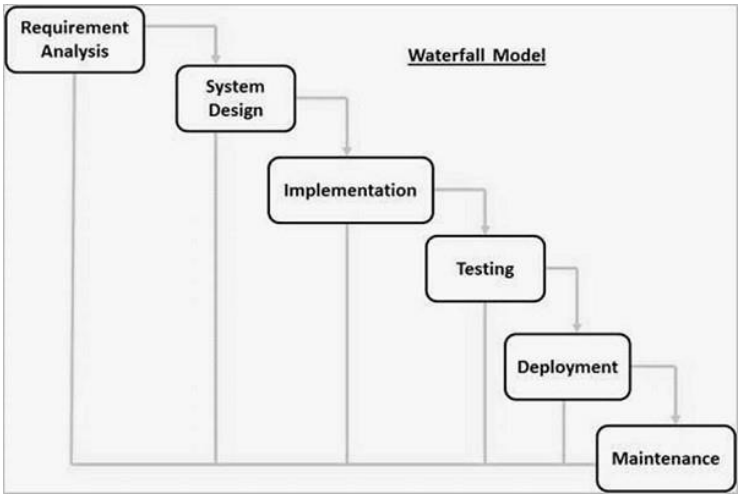
Ans: A software process model is an abstraction of the software development process. The models specify the stages and order of a process. So, think of this as a representation of the **orderof activities** of the process and the **sequence** in which they are performed.

- Waterfall model
- V model
- Incremental model
- RAD model
- Agile model
- Iterative model
- Prototype model
- Spir

al model

Waterfall Model - Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase actsas the input for the next phase sequentially. The following illustration is a representation of the different phases of the Waterfall Model.



The sequential phases in Waterfall model are −

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developedand tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions arereleased. Maintenance is done to deliver these changes in the customer environment.

Unit-2

1. **Elaborate about Functional and Non- Functional requirements in software engineering? Ans:**

FUNCTIONAL REQUIREMENTS

Functional requirements define a function that a system or system element must be qualified to perform and must be documented in different forms. The functional requirements describethe behavior of the system as it correlates to the system's functionality.

- Should be both complete and consistent
- Completeness -- All services required by the user should be defined
- Consistent -- Requirements should not have contradictory definition
- Difficult to achieve completeness and consistency for large system

NON-FUNCTIONAL REQUIREMENTS

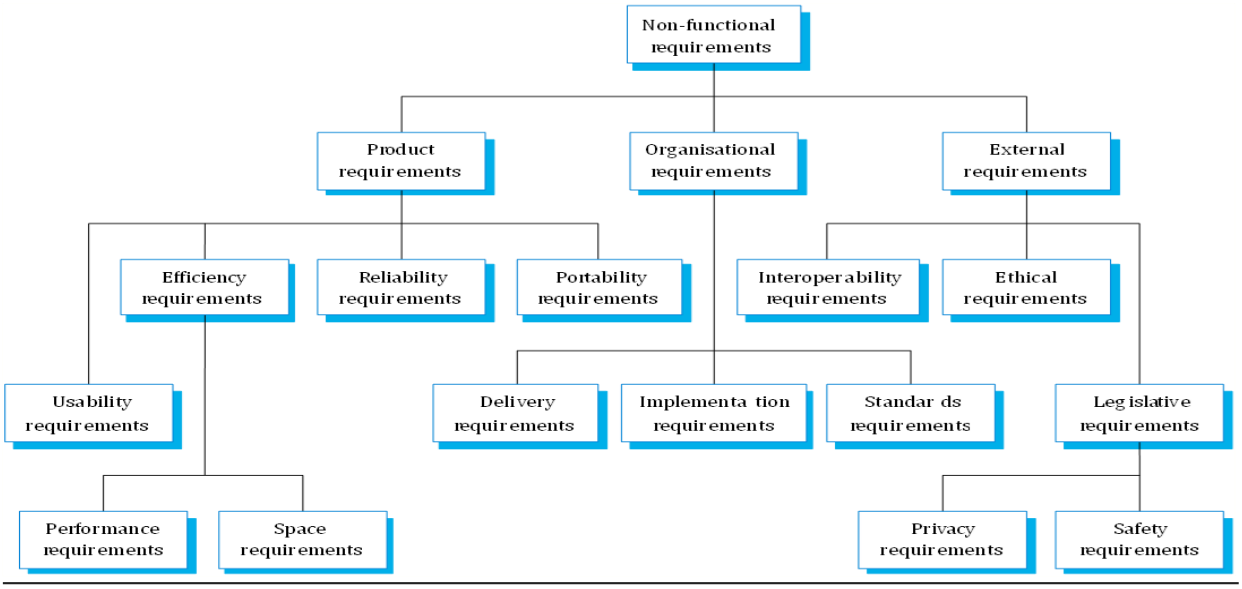
Non-functional requirements are not related to the software's functional aspect. They can be the necessities that specify the criteria that can be used to decide the operation instead of specific behaviors of the system. Basic non-functional requirements are - usability, reliability, security, storage, cost, flexibility, configuration, performance, legal or regulatory requirements,etc.

- **Types of Non-functional Requirements**
- 1.Product Requirements -Specify product behavior
  -Include the following
    - Usability
    - Efficiency
    - Reliability
    - Portability

2. Organisational Requirements --Derived from policies and procedures
--Include the following:

- Delivery
- Implementation
- Standard

 3.External Requirements



2.What is the major distinction between user requirements and system requirements? Ans:

3.Explain about Software Requirement Document and illustrate IEEE requirements standards? Ans:

System Requirements Document is also known as system requirement specification. System requirements document is a set of documentation that describes the behavior and features of a software or system. It comprises of various elements that attempt to characterize the functionality needed by the client to satisfy their users. In other words, the system requirements document (SRD) describes the system-level performance and functional requirements for asystem.

System Requirements Document or System Requirements Specification is defined as a documentwhich defines what the software will do and how it will be required to perform, and it also defines the functionality the software needs to satisfy all stakeholders (users, business) requirements.

While the SRD capacities as an outline for dealing with the extent of a project, it eventually characterizes the functional and non-functional requirements of the system. The documentdoesn't layout design or technology solutions. These decisions will be taken by the developers later.

The well-written system requirement document should:

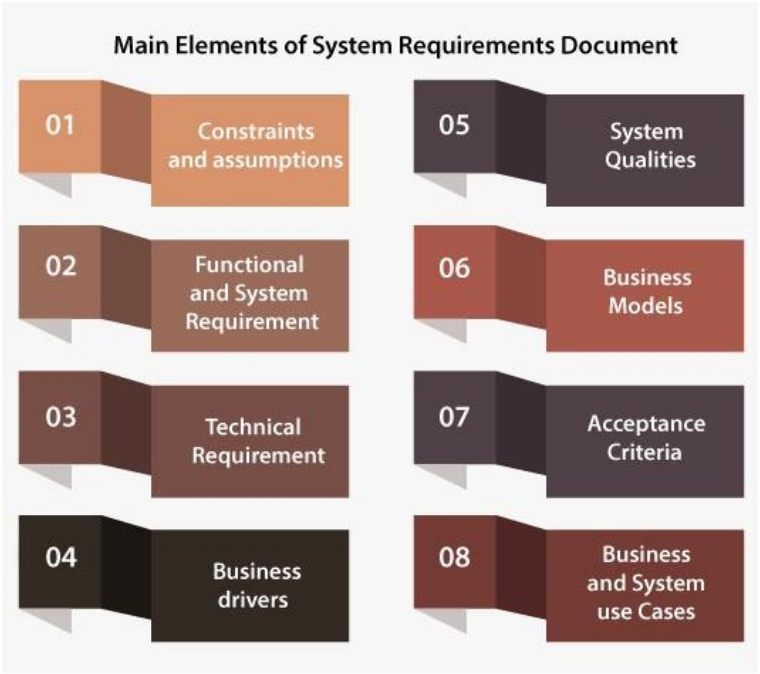
- Split the problem into manageable parts.
- Notify design specifications i.e., SRD needs to comprise adequate

information on softwarerequirements to present an effective design.

- Gives feedback to the customer or client.
- For testing and validation, serve as a reference.

According to the IEEE standards, SRDs must cover the following important topics.

- Quality
- Security/Privacy
- Functional capabilities
- Safety
- Performance levels
- Constraints and Limitations
- Data Structure/Elements
- Reliability
- Interfaces

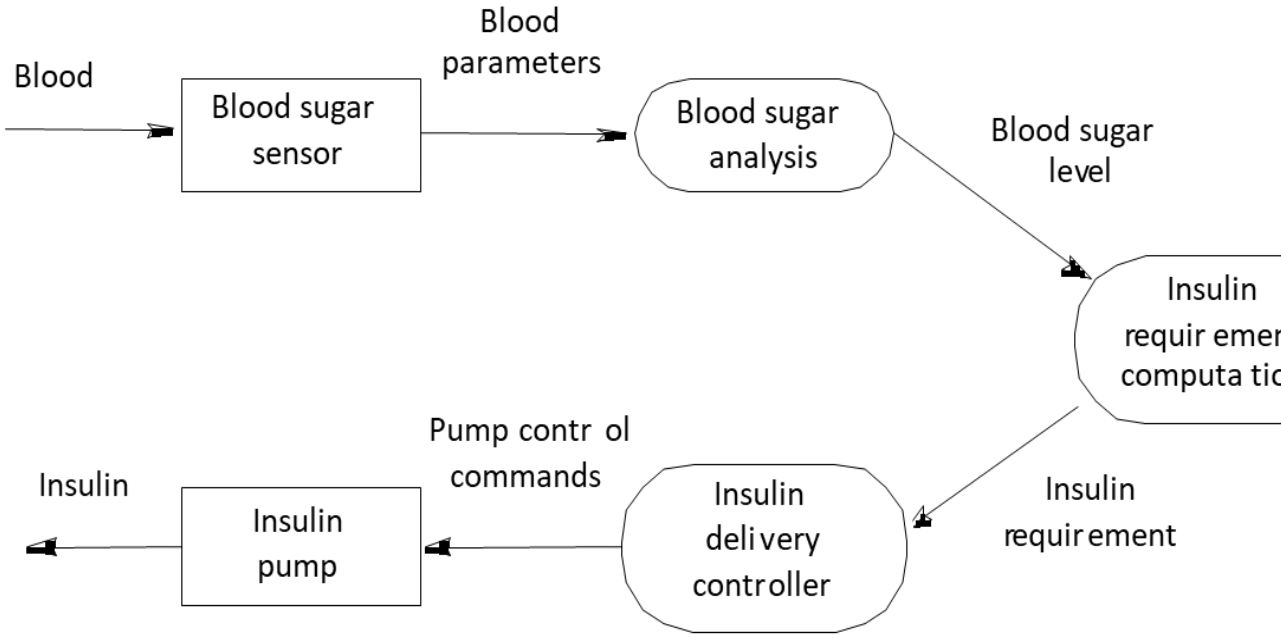


4. Define about Behavioural model and explain one type with neat diagram? Ans:

Behavioral Model is specially designed to make us understand behavior and factors that influence behavior of a System. Behavior of a system is explained and represented with the help of a diagram. This diagram is known as State Transition Diagram. It is a collection of states and events.It usually describes overall states that a system can have and events which are responsible for a change in state of a system.

So, on some occurrence of a particular event, an action is taken and what action needs to be takenis represented by State Transition Diagram.

Insulin pump DFD



Advantages :

Behavior and working of a system can easily be understood without any effort.Results are more accurate by using this model.
This model requires less cost for development as cost of resources can be minimal.It focuses on behavior of a system rather than theories.
Disadvantages :

This model does not have any theory, so trainee is not able to fully understand basic principle andmajor concept of modeling.
This modeling cannot be fully

automated. Sometimes, it's not easy to
understand overall result.
Does not achieve maximum productivity due to some technical issues or any errors.

5. Explain about the various types of Object model and elaborate one type with a neat diagram? Ans:

OBJECT MODELS:

- An object oriented approach is commonly used for interactive systems development
- Expresses the systems requirements using objects and developing the system in an objectoriented PL such as c++
- A object class: AN abstraction over a set of objects that identifies common attributes
- Objects are instances of object class
- Many objects may be created from a single class
- Analysis process
- -- Identifies objects and

object classesObject class in
UML

- --Represented as a vertically oriented rectangle with three sections
- (a) The name of the object class in the top section
- (b) The class attributes in the middle section
- (c) The operations associated with the object class are in lower section.

TYPES OF

OBJECT MODELS

INHERITANCE

MODELS

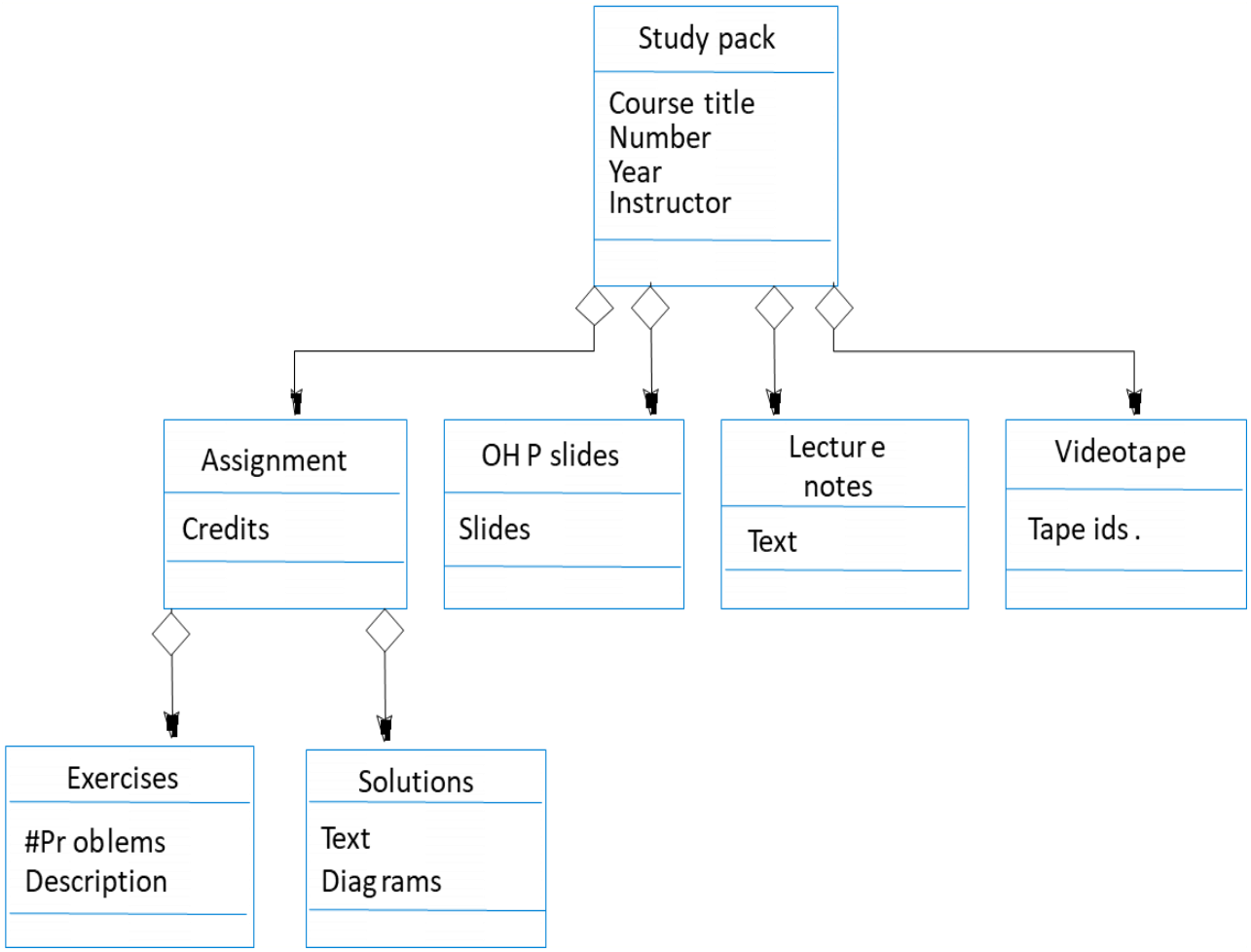
- A type of object oriented model which involves in object classes attributes
- Arranges classes into an inheritance hierarchy with the most general object class at the topof hierarchy

OBJECT-BEHAVIORAL MODEL

- Shows the operations provided by the objects
- Sequence diagram of UML can be used for

behavioral modeling**OBJECT AGGREGATION**

- Some objects are grouping of other objects
- An aggregate of a set of other objects
- The classes representing these objects may be modeled using an object aggregationmodel
- A diamond shape on the source of the link represents the composition


Study pack
Course title
Number
Year
Instructor
Assignment
Credits
OH P slides
Slides
Lectur e notes
Text
Videota pe
Tape ids .
Exercises
#Pr oblems
Description
Solutions
Text
Diag rams

UNIT 3

1. (a)List out the guidelines and quality attributes of design process

QUALITY GUIDELINES

• Uses recognizable architectural styles or patterns
• Modular; that is logically partitioned into elements or subsystems
• Distinct representation of data, architecture, interfaces and components
• Appropriate data structures for the classes to be implemented
• Independent functional characteristics for components
• Interfaces that reduces complexity of connection
• Repeatable method

QUALITY ATTRIBUTES

•
The FURPS quality attributes represent a target for all software design:
•
Functionality is assessed by evaluating the feature set and capabilities of the program, the generality of the functions that are delivered, and the security of the overall system.
•
Usability is assessed by considering human factors, overall aesthetics, consistency and documentation.
•
Reliability is evaluated by measuring the frequency and severity of failure, the accuracy of output results, and the mean – time –to- failure (MTTF), the ability to recover from failure, and the predictability of the program.
•
Performance is measured by processing speed, response time, resource consumption, throughput, and efficiency
•
Supportability combines the ability to extend the program (extensibility), adaptability, serviceability- these three attributes represent a more common term maintainability

(b). Define about cohesion and coupling?

VI.

Ans) Functional Independence:

The concept of functional independence is a direct outgrowth of modularity and the

concepts of abstraction and
information hiding.
Independence is assessed using two qualitative criteria: cohesion and coupling. Cohesion is an indication of the
relative functional strength of a module. Coupling is an indication of the relative interdependence among modules. Cohesion
is a natural extension of the information hiding.
A cohesion module performs a single task, requiring little interaction with other components in other parts of a program. Stated simply, a cohesive module should do just one thing.
Coupling is an indication of interconnection among modules in a software structure. Coupling depends on the
interface complexity between modules, the point at which entry or reference is made to a module, and what data pass across
the interface. In software design, we strive for lowest possible coupling. Simple connectivity among modules results in
software that is easier to understand and less prone to a —ripple effect‖, caused when errors occur at one location and propagates
throughout a system.
VII.
Refactoring :
Refactoring is a reorganization technique that simplifies the design of a component without changing its function or
behavior.
The designer may decide that the component should be refactored into 3 separate components, each exhibiting high cohesion.
The result will be software that is easier to integrate, easier to test, and easier to maintain.
They define characteristics of a well- formed design class.
High cohesion: A cohesive design class has a small, focused set of responsibilities and single- mindedly applies attributes and
methods to implement those responsibilities.
Low coupling: Within the design model, it is necessary for design classes to collaborate with one another. However,
collaboration should be kept to an acceptable minimum. If a design model is highly coupled the system is difficult to
implement, to test, and to maintain over time. In general, design classes within a subsystem should have only limited
knowledge of classes in other subsystems. This restriction, called the law of Demeter, suggests that a method should only sent
messages to methods in neighboring classes.

2.What are the types of design concepts and elaborate any five types of design concepts?
DESIGN CONCEPTS
•

M.A Jackson :software engineer is to recognize the difference between getting a program to work, and getting it right.

1. Abstractions
2. Architecture
3. Patterns
4. Modularity
5. Information Hiding
6. Functional Independence
7. Refinement
8. Re-factoring
9. Design Classes

DESIGN CONCEPTS

•

ABSTRACTION

•

Many levels of abstraction

•

Highest level of abstraction : Solution is slated in broad terms using the language of the problem environment

•

Lower levels of abstraction : More detailed description of the solution is provided

• Procedural abstraction

-- Refers to a sequence of instructions that a specific and limited function

• Data abstraction

-- Named collection of data that describe a data object

DESIGN CONCEPTS

• ARCHITECTURE

--Structure organization of program components (modules) and their interconnection

•

Architecture Models

(a). Structural Models

-- An organised collection of program components

(b). Framework Models

-- Represents the design in more abstract way

(c). Dynamic Models

-- Represents the behavioral aspects indicating changes as a function of external events

(d). Process Models

-- Focus on the design of the business or technical process

PATTERNS

• Provides a description to enables a designer to determine the followings :

(a). Whether the pattern is applicable to the current work

(b). Whether the pattern can be reused

(c). Whether the pattern can serve as a guide for developing a similar but functionally or structurally different pattern.

MODULARITY

• Divides software into separately named and addressable components, sometimes called modules

• Modules are integrated to satisfy problem requirements

• Consider two problems p1 and p2. If the complexity of p1 is cp1 and of p2 is cp2 then effort to solve p1=cp1 and effort to solve p2=cp2

• If cp1>cp2 then ep1>ep2

• The complexity of two problems when they are combined is often greater than the sum of the perceived complexity when each is taken separately

• Based on Divide and Conquer strategy : it is easier to solve a complex problem when broken into sub-modules

INFORMATION HIDING

• Information contained within a module is inaccessible to other modules who do not need such information

• Achieved by defining a set of Independent modules that communicate with one another only that information necessary to achieve S/W function

• Provides the greatest benefits when modifications are required during testing and later

• Errors introduced during modification are less likely to propagate to other location within the S/W

FUNCTIONAL INDEPENDENCE

•

A direct outgrowth of Modularity. abstraction and information hiding.

•

Achieved by developing a module with single minded function and an aversion to excessive interaction with other modules.

•

Easier to develop and have simple interface

•

Easier to maintain because secondary effects caused b design or code modification are limited, error propagation is reduced and

reusable modules are possible.
•
Independence is assessed by two quantitative criteria:
•
(1) Cohesion
•
(2) Coupling
•
Cohesion
•
-- Performs a single task requiring little interaction with other components
•
Coupling
•
--Measure of interconnection among modules
•
Coupling should be low and cohesion should be high for good design

REFINEMENT & REFACTORING
•
REFINEMENT
• Process of elaboration from high level abstraction to the lowest level abstraction
• High level abstraction begins with a statement of functions
• Refinement causes the designer to elaborate providing more and more details at successive level of abstractions
• Abstraction and refinement are complementary concepts.
• Refactoring
• Organization technique that simplifies the design of a component without changing its function or behavior.
• Examines for redundancy, unused design elements and inefficient or unnecessary algorithms
DESIGN CLASSES
•
Class represents a different layer of design architecture.
•
Five types of Design Classes
•
1. User interface class
•
-- Defines all abstractions that are necessary for human computer

interaction
•
2. Business domain class
•
-- Refinement of the analysis classes that identity attributes and services to implement some of business domain
•
3.Process class
•
-- implements lower level business abstractions required to fully manage the business domain classes
•
4.Persistent class
•
-- Represent data stores that will persist beyond the execution of the software

•

4.Differentiate between white-box and black-box testing?
Differences between Black Box Testing vs White Box Testing:

Black Box Testing	White Box Testing
is a way of software testing in which the internal	It is a way of testing the software in which the tes
tructure orthe program or the code is hidden and	has knowledgeabout the internal structure or the
othing is known about	or the program of the
.	software.
mplementation of code is not needed for black box esting.	Code implementation is necessary for white box
is mostly done by software testers.	It is mostly done by software developers.
o knowledge of implementation is needed.	Knowledge of implementation is required.
can be referred as outer or external software esting.	It is the inner or the internal software testing.
is functional test of the software.	It is structural test of the software.
his testing can be initiated on the basis of equirement	This type of testing of software is started after d design
pecifications document.	document.
o knowledge of programming is required.	It is mandatory to have knowledge of programmi
is the behavior testing of the software.	It is the logic testing of the software.
is applicable to the higher levels of testing of oftware.	It is generally applicable to the lower levels of so testing.
is also called closed testing.	It is also called as clear box testing.
is least time consuming.	It is most time consuming.
is not suitable or preferred for algorithm testing.	It is suitable for algorithm testing.
Can be done by trial and error ways and methods.	Data domains along with inner or internal boundaries can be better tested.
Example: search something on google by using keywords	**Example:** by input to check and verify loops

3.a) Discuss about the architectural styles and patterns?

ARCHITECTURAL STYLES

Describes a system category that encompasses:

• (1) a set of components

• (2) a set of connectors that enables “communication and coordination

• (3) Constraints that define how components can be integrated to form the system

• (4) Semantic models to understand the overall properties of a system

Types of Architectures

• Data centered Architecture

• Data flow architecture

• Call and return architecture

• Object oriented architecture

• Layered Architecture

Data centered Architecture

• Data-centered architecture consists of different components that communicate through shared data repositories.

• The most well-known examples of the data-centered architecture is a database architecture, in which the common database schema is created with data definition protocol – for example, a set of

related tables with fields and data types in

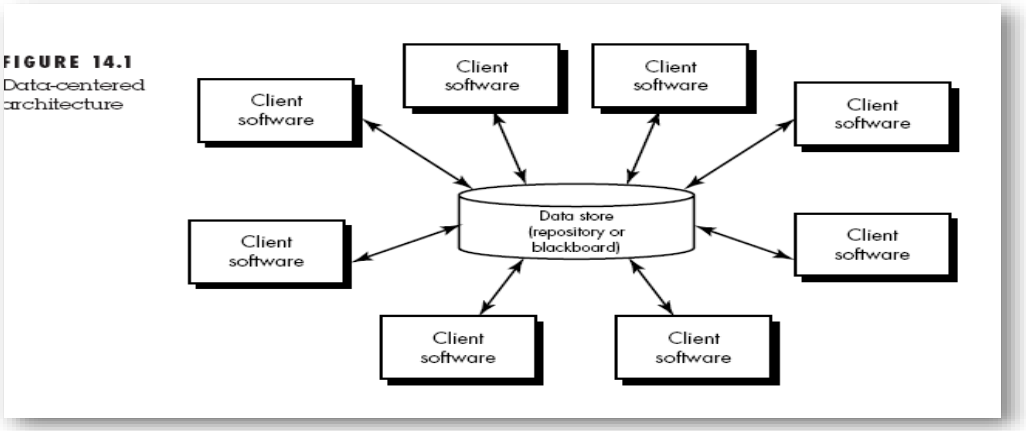


FIGURE 14.1
Data-centered architecture

Data-flow architectures

- Shows the flow of input data, its computationalcomponents and output data.
- Structure is also called pipe and Filter.
- Pipe provides path for flow of data.
- Filters manipulate data and work independent ofits neighboring filter.
- If data flow degenerates into a single line oftransform, it is termed as batch sequential.

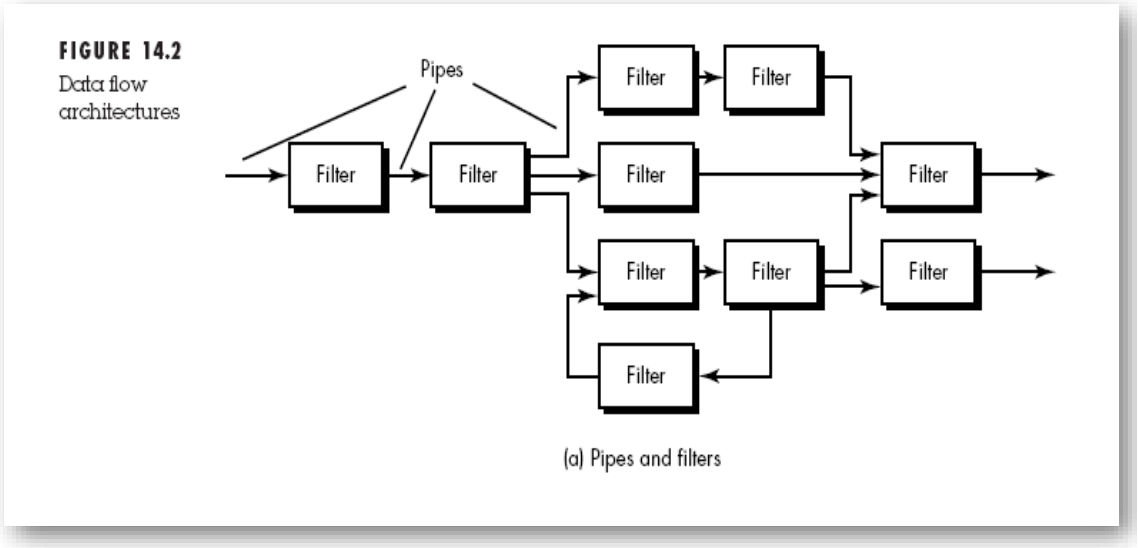


FIGURE 14.2
Data flow architectures

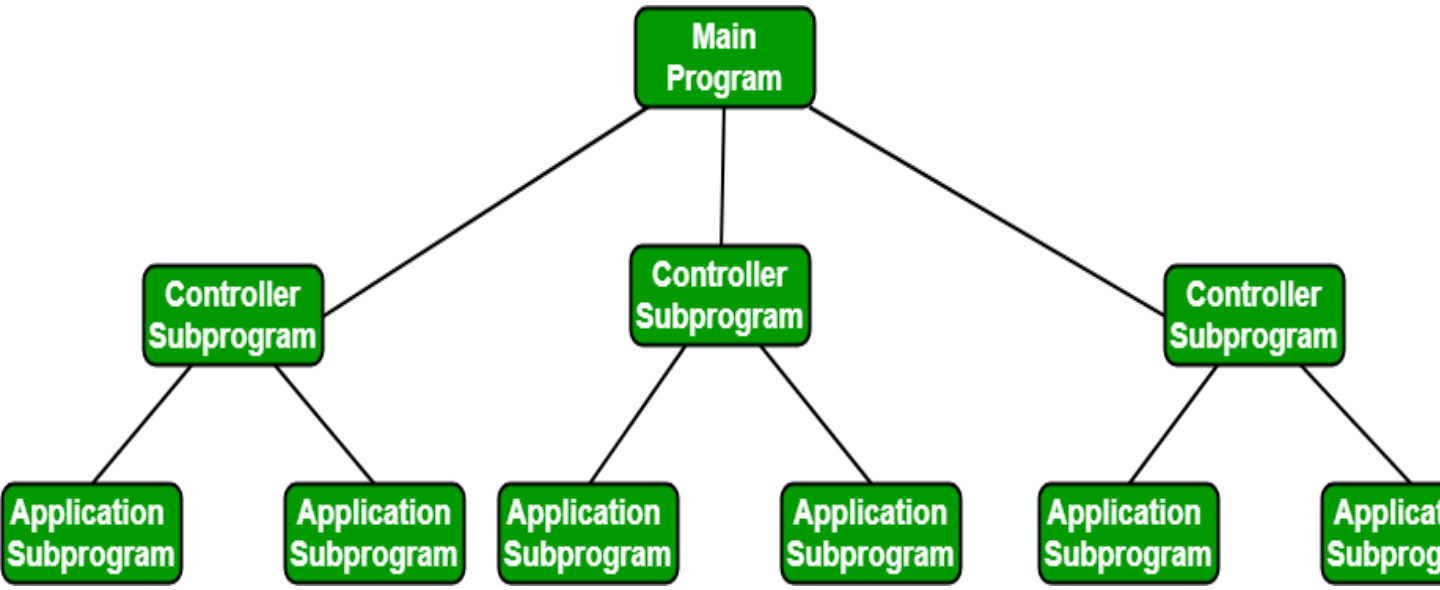
(a) Pipes and filters

Call and return architectures

- Achieves a structure that is easy to modify and scale
- Two sub styles

(1) Main program/sub program architecture

(2) Remote procedure call architecture



- **Main program or Subprogram architectures:** The main program structure decomposes into number of subprograms or function into a control hierarchy. Main program contains numberof subprograms that can invoke other components.

- Remote procedure call
  architecture: *This components is used topresent in a main program or sub programarchitecture distributed among multiple computers on a network.*

Object-oriented architectures

- The components of a system encapsulatedata and the operations.
- Communication and coordination betweencomponents is done via message

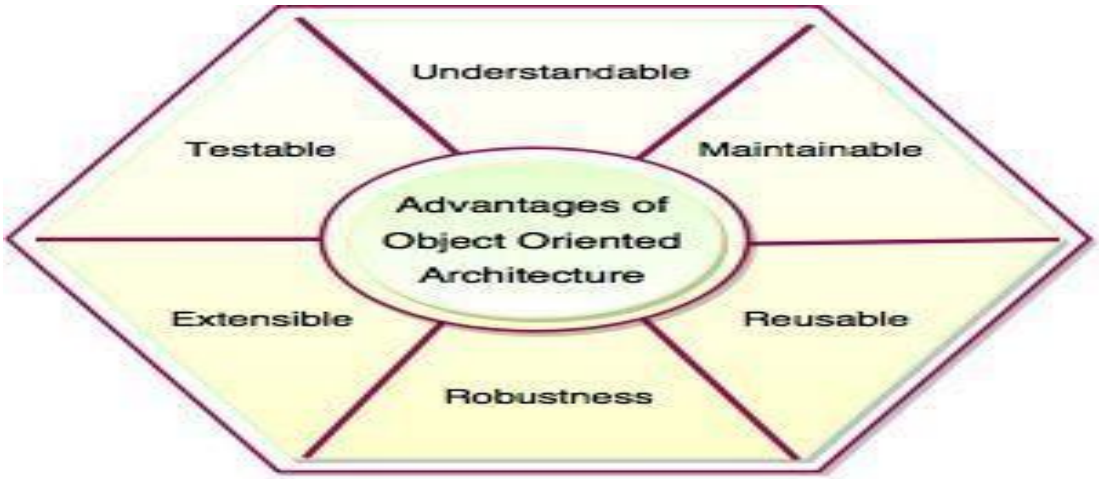


Fig. Advantages of Object Oriented Architecture

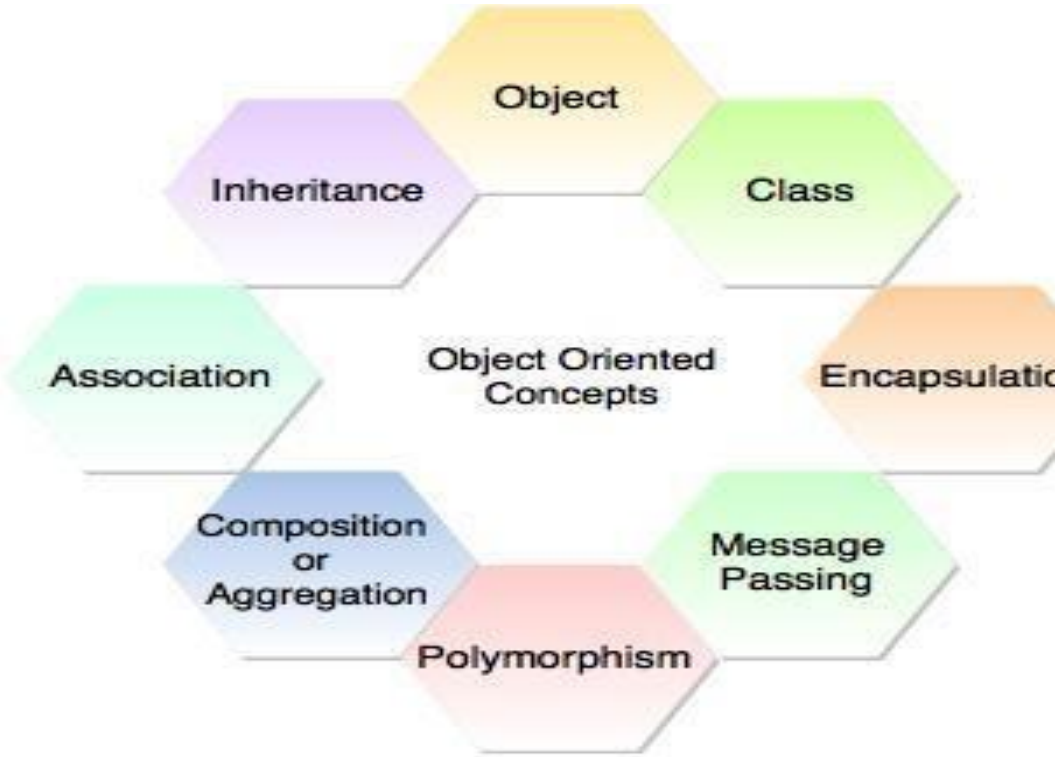
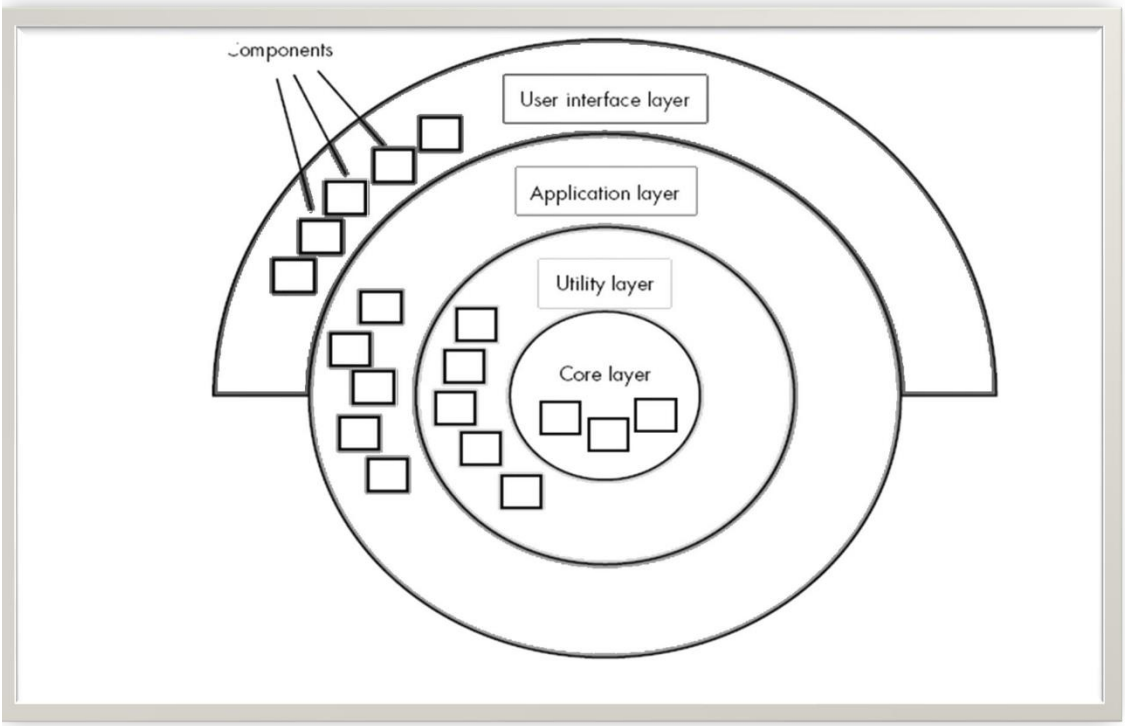


Fig. Object Oriented Concepts

Layered architectures

- A number of different layers are defined
- Inner Layer(interface with OS)
- Intermediate Layer (Utility services andapplication function)
- Outer Layer (User interface)

FIG: Layered Architecture



ARCHITECTURAL PATTERNS

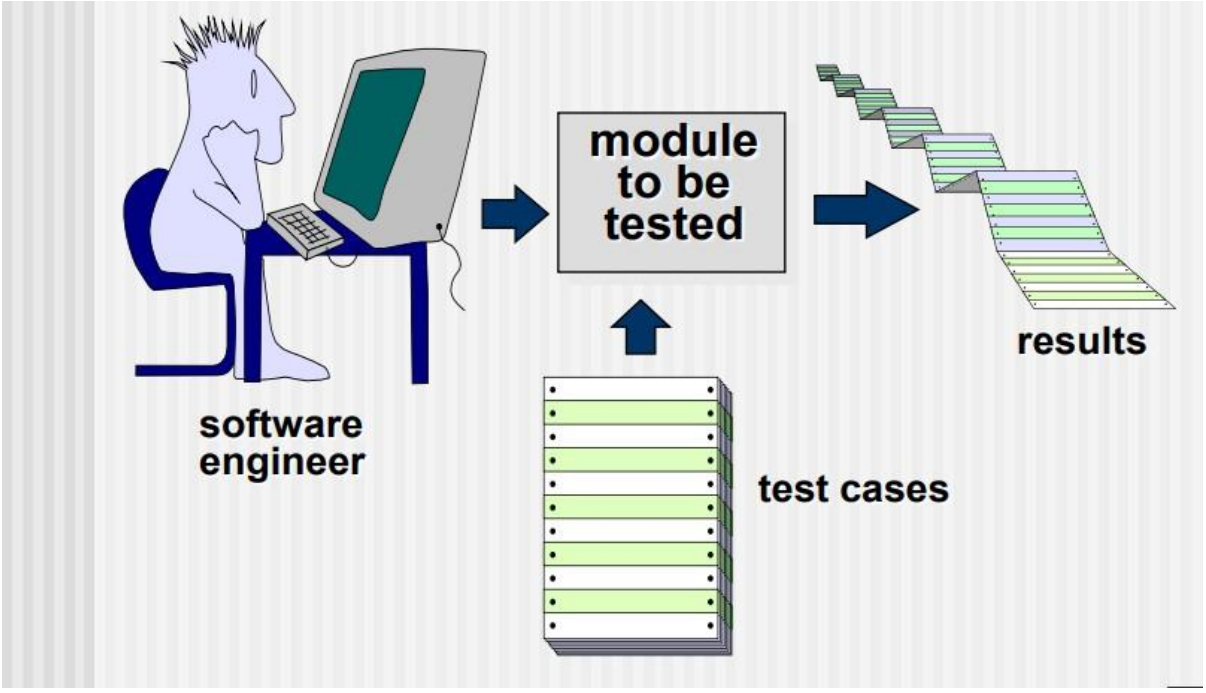
- A template that specifies approach for some behavioral characteristics of the system
- Patterns are imposed on the architectural styles
- Pattern Domains
- 1.Concurrency
- --Handles multiple tasks that simulates parallelism.
- --Approaches(Patterns)
- (a) Operating system process management pattern
- (b) A task scheduler pattern
- 2.Persistence
- --Data survives past the execution of the process
- --Approaches (Patterns)
- (a) Data base management system pattern
- (b) Application Level persistence Pattern(word processing software)
- 3.Distribution
- -- Addresses the communication of system in a distributed environment
- --Approaches(Patterns)
- (a) Broker Pattern
- -- Acts as middleman between client and server.

3) b) What are the types of diagrams in UML and draw any 3 diagrams for ATM withdrawal transaction?

5) Explain about the importance of test strategies for conventional software?

Testing Strategies

- Testing is the process of exercising a program with the specific intent of findingerrors prior to delivery to the end user.



Testing Strategies for Conventional Software

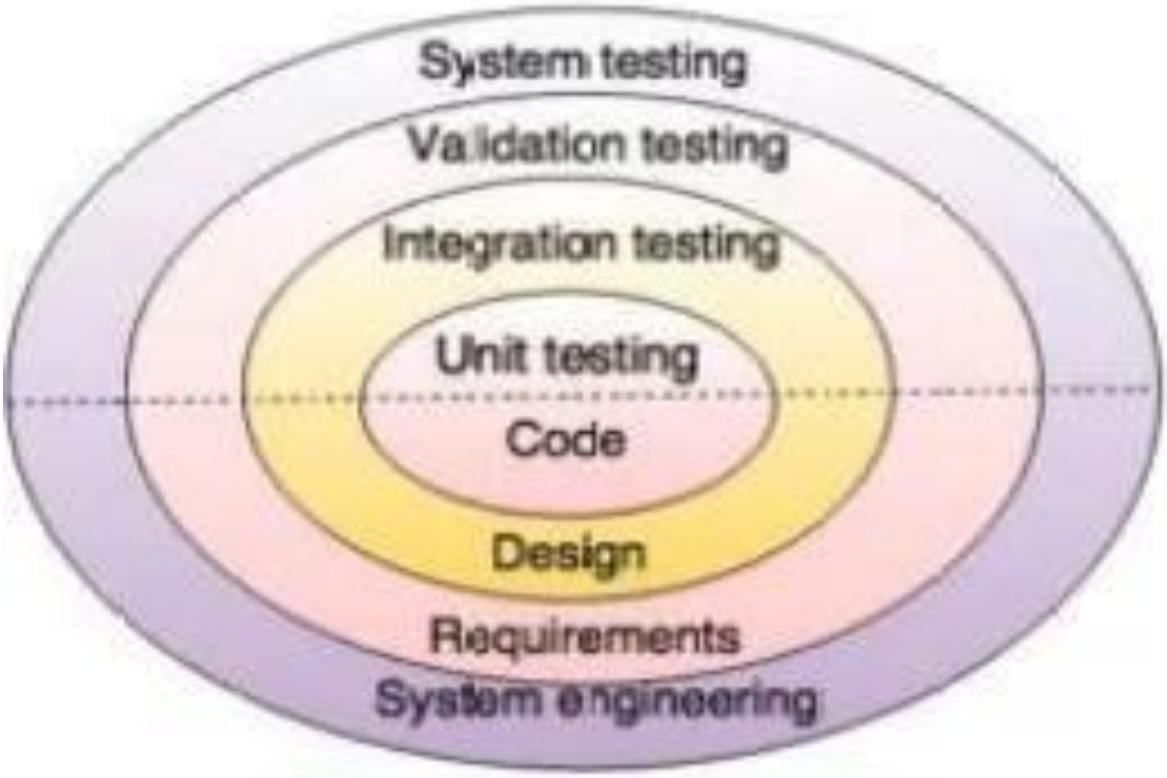
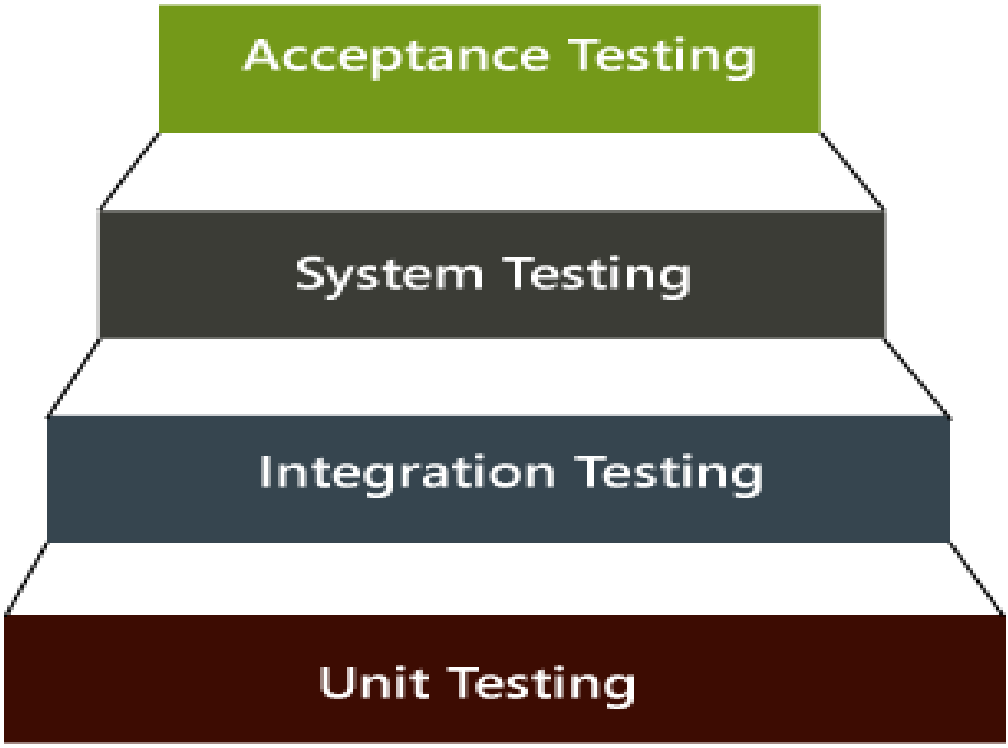


Fig. - Testing Strategy

Hierarchy of Testing Levels

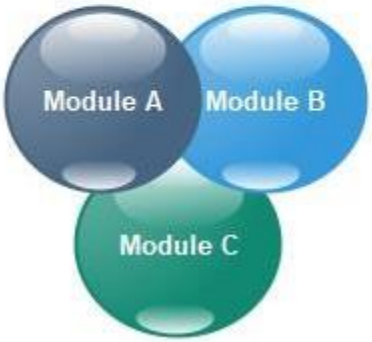


- It is the first level of functional testing. The aim behindunit testing is to validate unit components with its performance.
- A unit is a single testable part of a software system andtested during the development phase of the application software.
- Whenever the application is ready and given to the Test engineer, he/she will start checking every component of the module or module of the application independently orone by one, and this process is known as **Unit testing** or **components testing**.

- Integration testing is the second level of the software testing process comes afterunit testing.
- uses modules for testing purpose, and these modules are combined and tested inintegration testing.
- The goal of integration testing is to check the correctness of communication amongall the modules.

Integration testing

Tested in Unit Testing



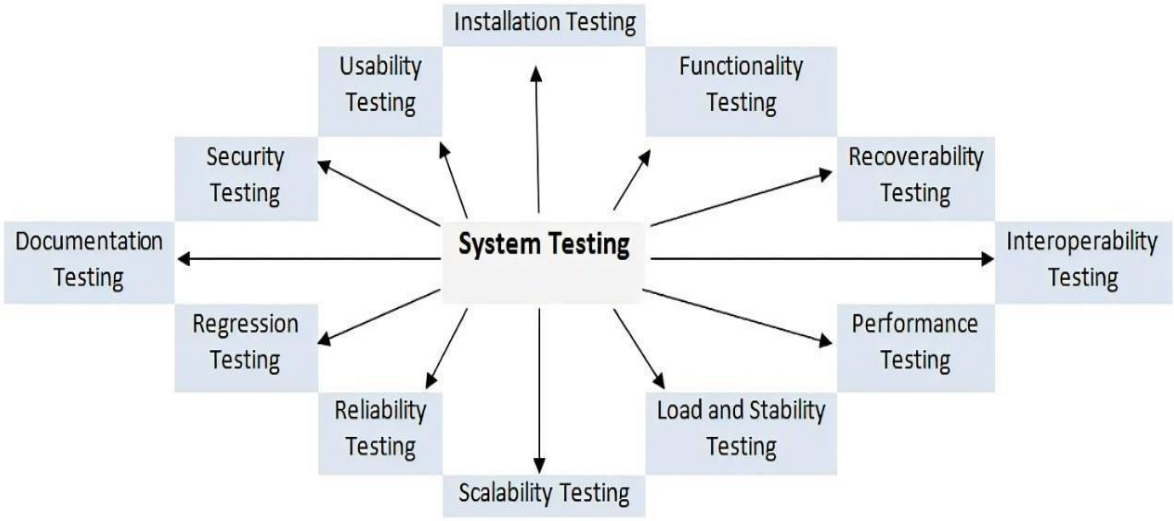
Under Integration Testing

System Testing

- System Testing includes testing of a fully integrated software system.
- To check the end-to-end flow of an application or the software is known as **System testing**.

- ***System testing in software engineering*** is of three types:

- **α-testing** → It is performed by development team responsible to test the system, mostly testing team. It is responsibility of the development team to make sure that point to point functionality as mentioned in SRS document is working properly and anything isn'tmissed. If anything is breaking then it must be fixed and properly tested.
- **ß-testing** → It is performed by a friendly set of customers to makesure that everything is all fine before finally submitting product for client evaluation.
- **Acceptance testing** → It is performed by customer to confirmwhether the product is as per their requirement or not.

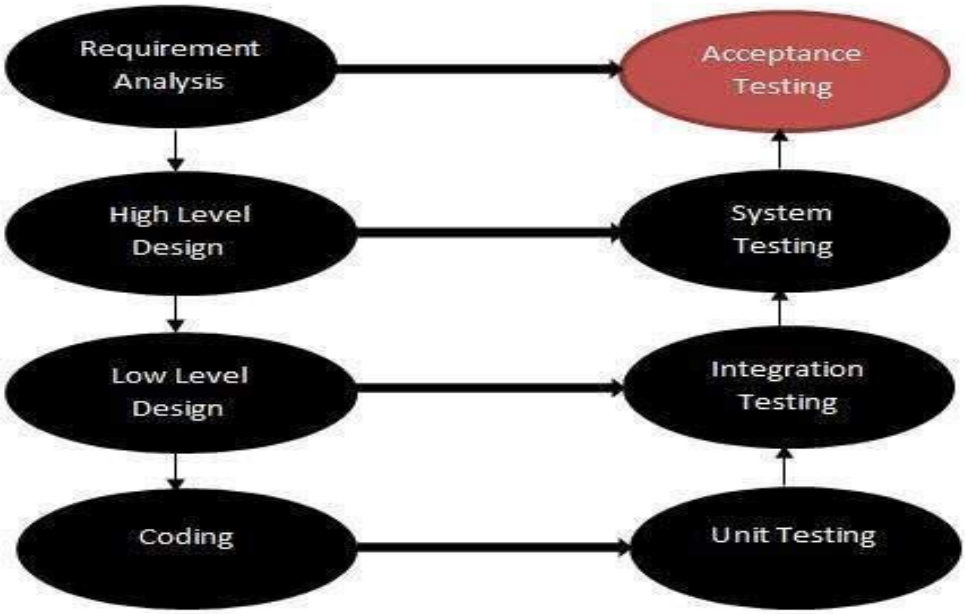


System Testing - © www.SoftwareTestingHelp.com

Acceptance testing

- It is the fourth and last level of softwaretesting.
- Acceptance testing, a testing technique performed to determine whether or not the software system has met the requirement specifications.
- User acceptance testing (UAT) is a type oftesting, which is done by the customer before accepting the final product.

Acceptance testing



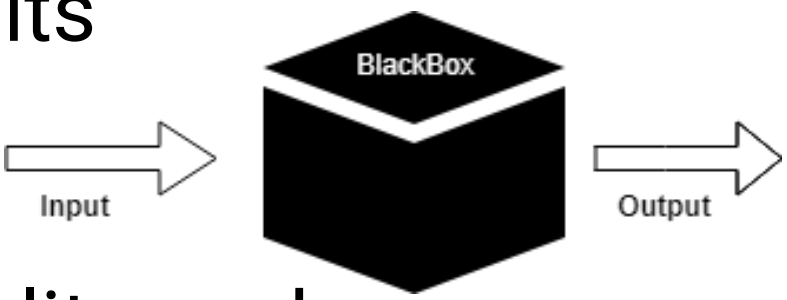
Acceptance testing

There are various forms of acceptancetesting:

- User acceptance Testing
- Business acceptance Testing
- Alpha Testing
- Beta Testing

Black box testing

- Black box testing is a technique does notrequire programming knowledge of the software.
- In this method, tester selects a function and gives input value to examine its



functionality, and checks whether the function is giving expected output or not.

Generic steps of black boxtesting

- In 1st step software is examined in the beginning.
- In the second step will take valid and invalidinput values to check that the software is processing them correctly or incorrectly.
- In the third step, checks error estimation.
- The fourth phase includes the execution of alltest cases.
- In the fifth step, the tester compares the expected output against the actual output.
- In the sixth step, there is any flaw in the

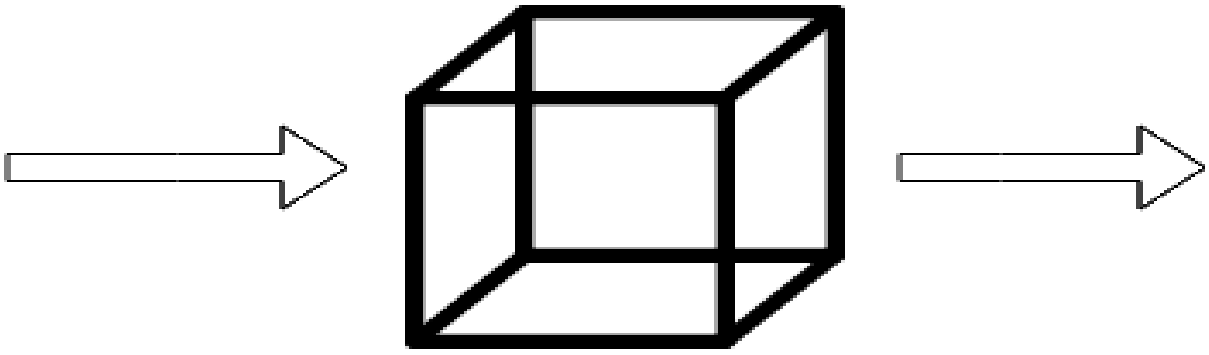
White Box Testing

- White box testing which also known asglass box.
- It tests internal coding and infrastructure ofa software.
- It is based on inner workings of an application.
- White box testing optimizes code so hidden errors can be identified.

Generic steps of white box

testing

- Developers will perform the white box testing automatically because it saves timerather than performing manually.



Whitebox Testing

UNIT 4

2) How to identify the risk in software?

Risk Identification

We identify the risk and list them out based on 2 approaches.

1.Generic risk identification

2.Product specific risk identification

Steps in Risk identification:

1.Preparation of risk item check list

2.Creating risk components and drivers list

1. Preparation of risk item check list

Risk items can be identified based on below components

-Product size(Risk associated with s/w size)

-Business impact(Condition of management)

-Customer characteristics(Choices of customer)

-Process definition(Which s/w follows organizational standards)

-Development environment(Availability of development tools)

-Staff size and experience

-Technology to be built

2. Creating risk components and drivers list:

RC and DL are prepared along with their probability of occurrence.

Risk Components:

Performance risk—the degree of uncertainty that the product will meet its requirements and be fit for its intended use.

Cost risk—the degree of uncertainty that the project budget will be maintained.

Support risk—the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.

Schedule risk—the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.

3)Define RMMM plan? Explain about RMMM Plan in detail?

RMMM

- R-Risk
- M-Mitigation
- M-Monitoring
- M-Management

Risk Mitigation: Preventing the risk from occurrence.

1.Communicate with staff and find the risks.

1. Eliminate the causes for risks.
2. Prepare a policy so that project will continue even if staff leaves in between.
3. Maintain all relevant documents. 5.Conduct reviews on time.

•

Risk Monitoring: This will occurred after risk has happened.

The project manager will monitor the following:

1. Behaviour of the team when making them pressure.
2. Spirit of teamwork(how people performs in team) 3.Cooperation among team
3. Problems that may occur.

4. Availability of job in and out of organization.

•

Risk Management:

1.It will done by the project manager when the risk becomes reality.

2.If mitigation is done properly Management becomes easy.

RMMM plan

•

It is a document in which all risk analysis activities are described.

•

Each risk is described using RIS(Risk info sheet).

Risk Information sheet

Project name:			
RiskID:	Date:	Probability:	Impact:
Origin:		AssignedTo:	
Description: *Refinement/Context:* *Mitigation/Monitoring:* *Trigger/Contigency plan:*			
Status:			
Organization:			*ClosingDate:*

- Probability: How frequently the risk is occurring. It can range from 10% to 100%.
- Impact: It is an “effect” on project. It can be negligible, marginal, critical or catastrophic.

- Origin: From where the risk has identified.
- AssignedTo: Who should rectified the risk and who should work on the risk.Personname has to assign.
- Description:It should be simple and everyone can read it. Description is a summaryrelated to risk.

- Refinement/Context: External factors (modifications in system)effecting the risk.
- Mitigation/Monitoring: Mitigation is used for risk prevention. If already occurredwe have to eliminate.
- Trigger/contigency plan: Plan will be given to team.
- Status: status of the risk(Incomplete or completed or in progress).
- Date:On which date you are updating the status of the risk.
- Originator: The name of the team member who identified the risk.If again riskencountered we will get down to thatperson to solve the risk.
- Closing Date: Closing date of the risk to be mentioned.

4) Explain about Quality concepts and Quality assurance?

Quality concepts

4 Quality concepts

1. Software Quality

2.Quality control

3.Quality assurance

4.Cost of quality

1.Software quality:

S/w quality is defined as how well the software works.

S/w quality is of 2 types

a. Quality of design

b. Quality of conformance

a.Quality of design

Characteristics of items or tools used by designer.

b.Quality of conformance

Degree to which Design specifications which are designed by the user are followed in manufacturing the product or software.

2.Quality control:

The process in which several activities are conducted to maintain the quality of the product.

several activities means Inspection, reviewing ,testing can be done either manually, automatically or semi-automatically.

3.Quality assurance

Here planned and systematic activities are done which are needed to provide high degree of reliability(confidence) in software.

4.Cost of Quality

Total cost required to obtain quality and conduct quality related activities. In simple way we can tell that how much cost you are spending to get quality product.

• Types of costs:

a. Prevention cost—Formal technical reviews, trainings

b. Appraisal cost—Testing, Inspection

c. Failure cost ---Review ,rework i)internal failure cost
ii)external failure cost.
i)Internal failure cost:
Defects found before delivering product to the customer.
ii)external failure cost:
Defects found after delivering product to the customer.

Software Quality Assurance

•

Software quality assurance is a planned and systematic plan of all actions necessary to provide adequate confidence that an item or product conforms to establish technical requirements.

•

SQA Activities
#1) Creating an SQA Management Plan:
#2) Setting the Checkpoints:
#3) Apply software Engineering Techniques:
#4) Executing Formal Technical Reviews:
#5) Having a Multi-Testing Strategy:
#6) Maintaining Records and Reports:
#7) Manage Good Relations:
#8) Performing SQA Audits:

• **Creating an SQA Management Plan:**
The foremost activity includes laying down a proper plan regarding how the SQA will be carried out in your project.
Setting the Checkpoints: This ensures regular quality inspection and working as per the schedule.

• **Apply software Engineering Techniques:** Prepare the project estimation using techniques like WBS (work breakdown structure), SLOC (source line of codes), and FP(functional point) estimation.

• **Executing Formal Technical Reviews:**
A meeting is conducted with the technical staff to discuss the actual quality requirements of the software and the design quality of the prototype. This activity helps in detecting errors in the early phase of SDLC and reduces rework effort in the later phases.

• **Maintaining Records and Reports**:
It is crucial to keep the necessary

documentation related to SQA and share the required SQA information with the stakeholders. The test results, audit results, review reports, change requests documentation, etc. It should be kept for future reference.

• **Manage Good Relations**:In fact, it is very important to maintain harmony between the QA and the development team.

• We often hear that testers and developers often feel superior to each other. This should be avoided as it can affect the overall project quality.

• **Having a Multi-Testing Strategy:** We mean that one should not rely on any single testing approach, instead, multiple types of testing should be performed so that the software product can be tested well from all angles to ensure better quality.

5)a)Discuss about ISO 9000 Quality Standards ?

ISO (International Standards Organization).

- ISO is an independent, non-governmental membership organization, which stands for International OrganizationforStandardization.
- It works with institutes and companies from around the world to help develop product and technology standards.
- ISO is derived from the Greek "isos" meaning equal, andthe members of ISO represent national standardsorganizations from 162 countries.
- The main objective of ISO is to facilitate trade, but its focusis on safety, process improvement, and quality in various areas.
- Types of ISO 9000 Quality Standards

- The ISO 9000 series of standards is based on the assumption that if a proper stage is followed for production, then good quality products are bound to follow automatically. The types of industries to which the variousISO standards apply are asfollows.
- **ISO 9001:** This standard applies to the organizations engaged in design, development, production, and servicing of goods.This is the standard thatapplies to most software development organizations.
- **ISO 9002:** This standard applies to those organizations which do not
- design products but are only involved in the production. Examples of thesecategory industries contain steel and car manufacturing industries that buy the product and plants designs from external sources and are engagedin only manufacturingthose products. Therefore, ISO 9002 does not apply to software development organizations.
- **ISO 9003:** This standard applies to organizations that are involved only in the installation and testing of the products. Forexample, Gas companies.

b) Write some of the Formal Technical Reviews in quality management?

•

Formal Technical reviews(FTR):

-It is an SQA activity performed by software engineer.

-It will identifies the logic of the software, functionality of the software and also implementation of the software.

-Also checks the all requirements are implemented or not.

-Also checks the s/w according to standards

•

4-steps of FTR:

1. Review meeting

-Short duration

-advance preparation
-3/4/5 people involved

2.Review reporting and record keeping

-Done by reviewer
-prepares a report

3.Review guidelines:

---establish in adavance
---Distribute to all reviewers
---All should agree then only implemented

4.Sample driven review

All software work product samples are inspected.It means which sample having more and which having less errors.

1) Elaborate the concepts of Risk management Reactive vs Proactive Risk strategies? What is software risks ?

Risk Management

- **Risk:**

"risk" is a problem that could cause some loss or threaten theprogress of the project, but which has not happened yet. "Tomorrow problems are today's risk."

Risk Management is the system of identifying addressing andeliminating these problems before they can damage the project.

Risk Management:is an important part of project planning activities. It involves identifying and estimating the probabilityof risks with their order of impact on the project.

Software Risks

- Risk identification and management are the main concerns in every softwareproject. Effective analysis of software risks will help in effective planning andassignment of work.
- Risk always involves two characteristics:
- Uncertainty—the risk may or may not happen; that is, there are no 100% probablerisks
- Loss—if the risk becomes a reality, unwanted consequences or losses will occur

Categories of Risks: 1.Schedule Risk 2.Budget Risk 3.Operational Risks 4.Technical Risks 5.Programmatic Risks 6.Predictable 7.Unpredictable

- **Technical risks** threaten the quality and timeliness of the software to be produced
- Identify potential design, implementation,interface, verification, and maintenance problems.
- Specification ambiguity, technical uncertainty,technical obsolescence, and "leading-edge" technology are risk factors.
- Technical risks occur because the problem isharder to solve than we thought it would be.

- **Known risks** are those that can be uncovered after careful evaluation of the project plan, the business and technical environment in which theproject is being developed, and other reliable information sources.
- **Predictable risks** are extrapolated from pastproject experience.
- **Unpredictable risks** are the joker in the deck. They can and do occur, but they are extremelydifficult to identify in advance.You are not thinking the risk but it will happen

Business risks threaten the viability of the software to be built.

- Top five business risks :
- (1) building an excellent product or system that no one really wants(market risk)
- (2) building a product that no longer fits into the overall business strategy for the company (strategic risk)i.e Not following companiespolicies
- (3) building a product that the sales force doesn't understand howto sale(sales risk).
- (4) losing the support of senior management due to a change in focus or a change in people (management risk).i.e senior managersleft the company
- (5) losing budgetary or personnel commitment (budget risks).

Reactive vs Proactive Risk strategies

- **Reactive(Past)**: Generally Reactive means “Respond to incident or accidents that have already happened.

- Mistakes are addressed as they are noticed.
- “Don’t worry I will think of something”. Most of the team and managers rely on this approach.

- Then the team takes action to correct the problem rapidly.
- In this we will get bad results.
- Lessions learned from accidents occurred.
- No preventive care
- Corrected only after occurrence.
- If we are not solved the project is in danger.

- **Proactive(Present):** It identifies hazardous conditions through analysis of organization's process.

- Begins before risk occurs
- First identify the risk
- Then we assess the impact of the risk on s/w.
- Then risks are prioritized.
- High priority risks are managed first
- Continuous hazard monitoring .
- Systematic data collection.
- Potential risks are identified
- Their probability and impact are assessed
- They are ranked by importance
- Not all risks can be avoided

UNIT 5

1.What are the software project management activities explain in detail?

Ans) Software Project Management consists of many activities, that includes planning of the project, deciding the scope of product, estimation of cost in different terms, scheduling of tasks, etc.

1. **The list of activities are as follows:**
2. Project planning and Tracking
3. Project Resource Management
4. Scope Management
5. Estimation Management
6. Project Risk Management
7. Scheduling Management
8. Project Communication Management
9. Configuration Management

Now we will discuss all these activities -

1. Project Planning: It is a set of multiple processes, or we can say that it a task that performed before the construction of the product starts.

2. Scope Management: It describes the scope of the project. Scope management is important because it clearly defines what would do and what would not. Scope Management create the project to contain restricted and quantitative tasks, which may merely be documented and successively avoids price and time overrun.

3. Estimation management: This is not only about cost estimation because whenever we start to develop software, but we also figure out their size(line of code), efforts, time as well as cost.

If we talk about the **size**, then Line of code depends upon user or software requirement.

If we talk about **effort**, we should know about the size of the software, because based on the size we can quickly estimate how big team required to produce the software.

If we talk about **time**, when size and efforts are estimated, the time required to develop the software can easily determine.

And if we talk about cost, it includes all the elements such as:

Size of software
Quality
Hardware
Communication
Training
Additional Software and tools
Skilled manpower

4. Scheduling Management: Scheduling Management in software refers to all the activities to complete in the specified order and within

time slotted to each activity. Project managers define multiple tasks and arrange them keeping various factors in mind.

Find out multiple tasks and correlate them.
Divide time into units.
Assign the respective number of work-units for every job.
Calculate the total time from start to finish.
Break down the project into modules.

5. Project Resource Management: In software Development, all the elements are referred to as resources for the project. It can be a human resource, productive tools, and libraries.

Resource management includes:
Create a project team and assign responsibilities to every team member
Developing a resource plan is derived from the project plan.
Adjustment of resources.

6. Project Risk Management: Risk management consists of all the activities like identification, analyzing and preparing the plan for predictable and unpredictable risk in the project.

Several points show the risks in the project:
The Experienced team leaves the project, and the new team joins it.
Changes in requirement.
Change in technologies and the environment.
Market competition.

7. Project Communication Management: Communication is an essential factor in the success of the project. It is a bridge between client, organization, team members and as well as other stakeholders of the project such as hardware suppliers.

From the planning to closure, communication plays a vital role. In all the phases, communication must be clear and understood. Miscommunication can create a big blunder in the project.

8. Project Configuration Management: Configuration management is about to control the changes in software like requirements, design, and development of the product.

2. Identify some of the objectives and goals in software project management?

Goals

- A goal is an achievable outcome that is generally broad and longer term while an objective is shorter term and defines measurable actions to achieve an overall goal.

- Types of goals

There are three main types of goals: time- bound, outcome-oriented, and process-orientedgoals.

- Time-bound goals

Time-bound goals are focused on setting timely actions. This means they are driven by deadlinesand target dates. They provide a high-level explanation for what your team should be striving toward.

- Outcome-oriented goals

Outcome-oriented goals are focused on the endresult. Rather than focusing on specific deadlines, outcome-oriented goals look to accomplish the action above all else.

- Process-oriented goals

Process-oriented goals focus on achieving newinternal systems and processes. Instead of focusing on a specific outcome, process- oriented goals prioritize the work and how it’saccomplished.

Objectives

- There are three main types of objectives, eachfeaturing unique perspectives when it comes to completing tasks.

Strategic objectives

Tactical objectives

Operational objectives

Strategic objectives

- Strategic objectives ensure team members have a clear project direction and are alignedon the project's purpose and overall timeline.

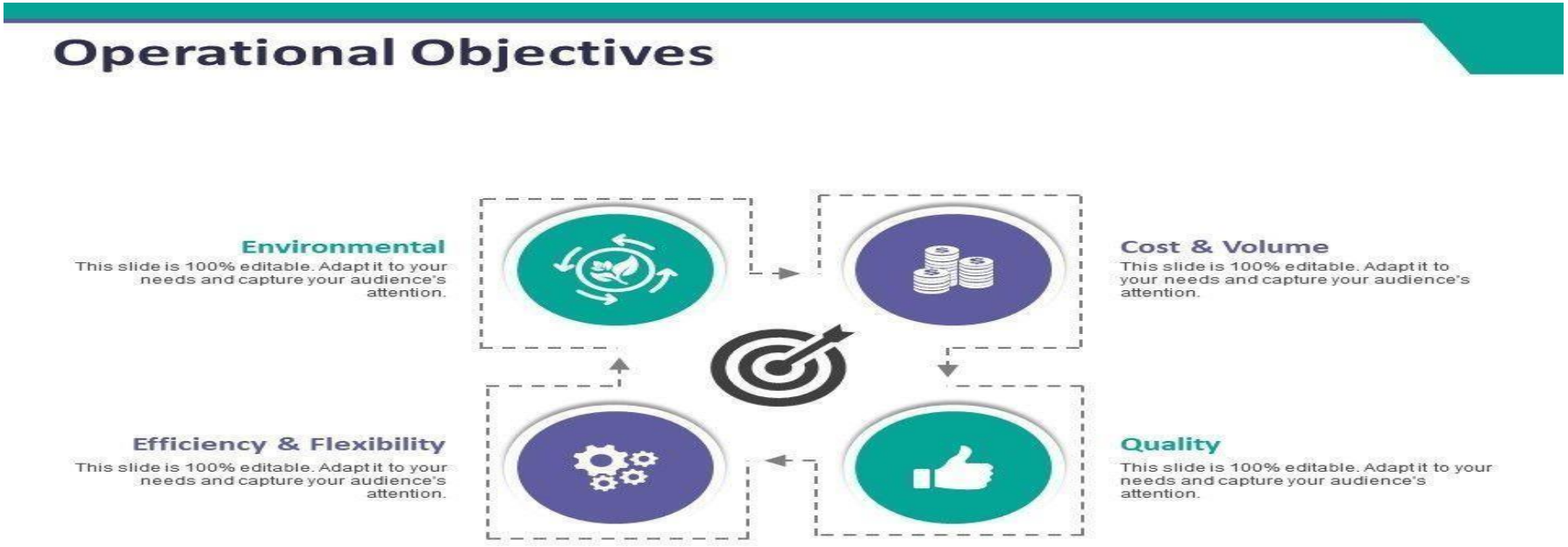
Example:

What we are doing for it?What is required?

How we will do it?

Tactical objectives

- This type of objective looks at the results ofshort-term tasks.
- short-term tasks can set for 6 to 12 months.



Operational objectives

- Operational objectives contribute to daily, weekly, and monthly goals by organizing task schedules and aligning different departments.

3.a) Explain in detail about project scope ?

Project Scope

- The project scope is the **total amount of workthat needs to be done to complete a project**.

Four Steps

- Step 1 Identify Project Needs
- Step 2: Identify Project Objectives
- Step 3: Identify Project Expectations
- Step 4: Identify Project Constraints

- **Project Scoping Process**

Planning: The first step in the project scoping process is to understand what needs to be doneand plan projects accordingly.

Controlling: During this phase it is essential thatyou control and monitor everything efficiently.

Closing:

- Auditing of all project deliverables
- Assessment of project outcomes
- Compare project outcome to estimated outcome

b) Discuss some of the challenges in software project management?

Ans)

1. Unclear and undefined expectations
2. Time constraint
3. Changing project requirements and priorities
4. Poor communication
5. Skills management
6. Changing technologies
7. Keeping everyone on the same page
8. Motivating team members
9. Steep learning curve
10. Project cancellation
11. Estimation
12. High competition
13. Upgrade to a new system
14. Quality testing
15. Managing risks

Unclear and undefined expectations

Defining goals is crucial for a successful project. However, sometimes the project managers might fail to gather requirements clearly from the clients, which further complicates the progress. The expectations and objectives become ambiguous, and the output deviates from the actual results. The time and resources you spent will be for nothing. More than that, it will affect your image and reputation in the market.

Moreover, if the goals are unclear and unrealistic, project delay is one thing, but it will double your cost, clients' quality and expectations will not meet, demotivate your whole team, etc.

Time constraint

"Less time, more work." is typical in the software development process. But what makes this a common challenge is the unrealistic deadlines from the clients sometimes, which leads to a race against time. As a result, it affects the quality of the software. Because of high

competition and changing technology, it has become even more challenging for the development team and the project managers to cope with tight deadlines.

Moreover, unrealistic deadlines create pressure, which leads to more problems during the development process.

Changing project requirements and priorities

One of the substantial challenges in software project management is changing requirements and priorities. The needs of the clients may change as the project progresses. And when that happens, extra time, cost, and resources need to be allocated, which further impacts the project.

Moreover, planning, reviewing, and implementing the new requirements into the existing ones take a toll on the development team and the project manager.

Poor communication

Poor communication with clients, other stakeholders, and the development team affects the overall project.

First, miscommunication with clients will prevent getting the requirements, which will complicate the working process.

Second, it will create conflicts within the team. If members don’t communicate with each other, then the roles and responsibilities will overlap with each other.

Moreover, it is important to communicate and coordinate with each other for smooth operation.

Skills management

One of the biggest challenges of a project manager is finding the right team with the skills and experience to finish the job. If the members are not skilled enough, then the chances of failure increases. It puts the project managers in a risky situation.

Changing technologies

The IT industry is one of the fastest-growing industries. So, rapid change in technologies isn’t a new thing. Today’s technology may not work tomorrow. So, it is essential to get acquainted with the latest ones.

Moreover, the market is now full of hundreds of software management tools with tons of features. So, the project managers are responsible for selecting the best ones that are right for their company and the team.

Keeping everyone on the same page

Keeping everyone together is crucial for a successful project. However, it becomes very challenging for the project managers to do it since each individual is from a different background and has distinct skills. So, it is the responsibility to find ways to keep them on the same track.

If everyone shares the same goal and is on the same page, it will make things go smoothly. Everybody will know their roles and responsibilities.

Motivating team members

Sometimes, unrealistic expectations and tight deadlines make the team demotivated. Plus, conflict among the members and the problems faced by them affects the project. So, the project managers have to keep them motivated and resolve the dispute.

Project cancellation

One of the most significant challenges the project managers have to face is the cancellation of the project mid-way. This is the worst phase they have to face. And there can be various reasons.

Estimation

It plays a crucial role in the success of the project. However, unrealistic estimation affects the overall project as well as the performance of the team.

These are the common challenges in software project management project managers face. To cope with them and handle the projects seamlessly, they require skills, experience, and knowledge.

4. What are project activities in project management?

- The activity planning stage is very important for any type of project. Let's take a look at a project planning activities list for activity planning.
- Determine project requirements
- Identify all database
- Determine cost estimates
- Revise the risk assessment
- Define and identify critical success factors
- Write up the project .

- Create a detailed project plan
- Begin the project

Project Planning

Software project planning is task, which is performed before the production of software actually starts. It is there for the software production but involves no concrete activity that has any direction connection with software production; rather it is a set of multiple processes, which facilitates software production. Project planning may include the following:

Scope Management It defines the scope of project; this includes all the activities, process need to be done in order to make a deliverable software product. Scope management is essential because it creates boundaries of the project by clearly defining what would be done in the project and what would not be done. This makes project to contain limited and quantifiable tasks, which can easily be documented and in turn avoids cost and time overrun.

During Project Scope management, it is necessary to -

- Define the scope
- Decide its verification and control
- Divide the project into various smaller parts for ease of management.
- Verify the scope
- Control the scope by incorporating changes to the scope

Project Estimation

For an effective management accurate estimation of various measures is a must. With correct estimation managers can manage and control the project more efficiently and effectively.

Project estimation may involve the following:

- **Software size estimation**

  Software size may be estimated either in terms of KLOC (Kilo Line of Code) or by calculating number of function points in the software. Lines of code depend upon coding practices and Function points vary according to the user or software requirement.

- **Effort estimation**

The managers estimate efforts in terms of personnel requirement and man-hour required to produce the software. For effort estimation software size should be known. This can either be derived by managers' experience, organization's historical data or software size can be converted into efforts by using some standard formulae.

- **Time estimation**

  Once size and efforts are estimated, the time required to produce the software can be estimated. Efforts required is segregated into sub categories as per the requirement specifications and interdependency of various components of software. Software tasks are divided into smaller tasks, activities or events by Work Breakthrough Structure (WBS). The tasks are scheduled on day-to-day basis or in calendar months.

  The sum of time required to complete all tasks in hours or days is the total time invested to complete the project.

- **Cost estimation**

  This might be considered as the most difficult of all because it depends on more elements than any of the previous ones. For estimating project cost, it is required to consider -

  - Size of software
  - Software quality
  - Hardware
  - Additional software or tools, licenses etc.
  - Skilled personnel with task-specific skills
  - Travel involved
  - Communication
  - Training and support

5. Elaborate about effort estimation and infrastructure in project planning?

Ans)Effort estimation is the process of forecasting how much effort is required to develop or maintain a software application. This effort is traditionally measured in the hours worked by a person, or the money needed to pay for this work.

Effort estimation is used to help draft project plans and budgets in the early stages of the software development life cycle. This practice enables a project manager or product owner to accurately predict costs and allocate resources accordingly.

Effort estimation in Agile

Though effort estimation can be used in a traditional software development approach, it is more commonly associated with the Agile methodology. Here, the product owner must manage a list of project deliverables, known as a backlog. They will estimate the effort required to complete each item. Rather than using time or cost estimates, they will look at user stories and story points.

- A user story is a tool used to describe a software feature from the perspective of the consumer
- A story point is a unit that measures the amount of work in implementing a user story, taking into account the level of difficulty involved and the potential risk

A product owner will compare the features of their new project with a previous one to determine the complexity of their user story and assign suitable story points.

Agile effort estimation techniques

There are many different Agile effort estimation techniques to choose from. Here are three of the most common ones:

1. **Planning Poker:** In this method, team members sit together in a circle to assign values to story points. Each individual will have a set of cards with the numerical values that can be assigned: 0, 1, 2, 3, 5, 8, 13, 20, 40, and 100. The product owner will read out a user story to the team members. They will have a discussion and then decide which value it should have. If everyone is in agreement, the final estimate is decided. If not, the team will discuss further until a consensus is reached.
2. **T-shirt Sizes:** Here, story points take the form of sizes: extra-small (XS), small (S), medium (M), large (L), and extra-large (XL). Estimators will determine the sizes to get a quick and rough estimate that can be converted to numbers later.
3. **Dot voting:** This approach enables team members to sort items in the product backlog from low to high priority. User stories are posted on a board and estimators get four or five dots to use as votes. The one with the most dots is deemed the highest-priority item, and so on.

What Is Infrastructure Project Management?

Infrastructure project management focuses solely on infrastructure projects. However, it uses all the same standard methodologies and processes as other types of project management.

What are infrastructure projects?

Infrastructure is the basic structures, systems, and services required for operation. Think of infrastructure as the fundamentals that everything else is built upon.

The infrastructure of a community is called economic infrastructure. It includes roadways, sewers, railways, power lines, etc.

Here are some examples of economic infrastructure projects:

- Developing a new highway
- Installing underground power lines
- Replacing a section of the sewer system

There are also business and technology infrastructures. Business infrastructure can include facilities, operating systems, communication tools, and security systems. Some business infrastructure project examples are:

- Upgrading the phone lines in the building
- Installing a new sprinkler system
- Renovating the production facility

Technology infrastructure consists of hardware systems, software, network connections, and servers. These are some examples of technology infrastructure projects:

- Installing a new backup server
- Replacing all the computer hardware
- Upgrading the payroll system software

A technology company may have one project manager who only does hardware projects and another that only does software projects. While the basics are the same, specialization allows you to gain experience in specific project types.

This specialization can result in the following:

- Improved planning due to past lessons learned
- Better ability to problem solve due to experience with similar issues
- A greater understanding of the technical aspects of the project
- A reputation as an expert, which can lead to greater trust from your team and stakeholders