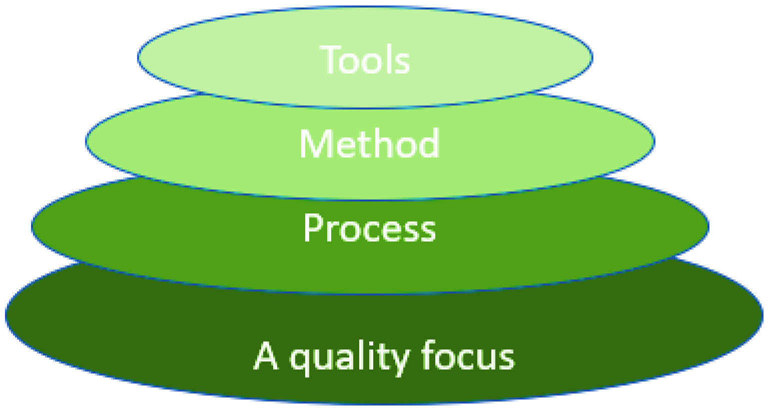
1.a) Define software engineering and explain software Engineering-a layered technology.

Software Engineering is an engineering branch related to the evolution of software product using well-defined scientific principles, techniques, and procedures. The result of software engineering is an effective and reliable software product.

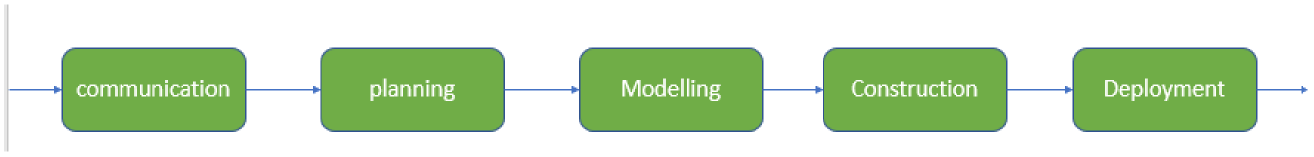
Software engineering is a fully layered technology, to develop software we need to go from one layer to another. All the layers are connected and each layer demands the fulfillment of the previous layer.



Layered technology is divided into four parts:

1. A quality focus: It defines the continuous process improvement principles of software. It provides integrity that means providing security to the software so that data can be accessed by only an authorized person, no outsider can access the data. It also focuses on maintainability and usability.

2. Process: It is the foundation or base layer of software engineering. It is key that binds all the layers together which enables the development of software before the deadline or on time. Process defines a framework that must be established for the effective delivery of software engineering technology. The software process covers all the activities, actions, and tasks required to be carried out for software development.



Process activities are listed below: -

Communication: It is the first and foremost thing for the development of software. Communication is necessary to know the actual demand of the client.

Planning: It basically means drawing a map for reduced the complication of development.

Modeling: In this process, a model is created according to the client for better understanding.

Construction: It includes the coding and testing of the problem.

Deployment: It includes the delivery of software to the client for evaluation and feedback.

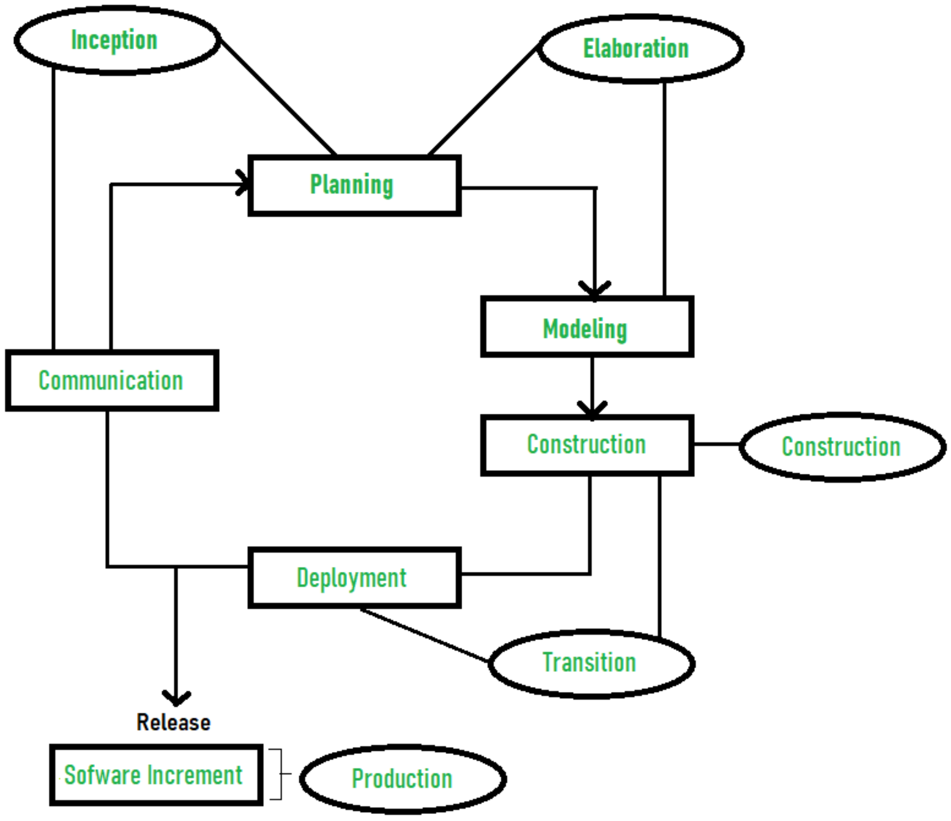
3. Method: During the process of software development the answers to all "how-to-do" questions are given by method. It has the information of all the tasks which includes communication, requirement analysis, design modeling, program construction, testing, and support.

4. Tools: Software engineering tools provide a self-operating system for processes and methods. Tools are integrated which means information created by one tool can be used by another.

1.b) Illustrate unified process model for software development with neat diagram.

Unified Process Model is a software development process for object-oriented models. It is created by Rational corporation and is designed and documented using UML (Unified Modeling Language). This process is included in IBM RMC product. IBM allows us to customize, design, and personalize the unified process. UPM is proposed by Ivar Jacobson, Grady Bootch, and James Rambaugh.

Some characteristics of UPM include use-case driven, Iterative and Incremental by nature, delivered online using web technology, can be customized or tailored in modular and electronic form, etc. UPM reduces unexpected development costs and prevents wastage of resources.



1. Inception –
   - Communication and planning are the main ones.
   - Identifies the scope of the project using a use-case model allowing managers to estimate costs and time required.
   - Customers' requirements are identified and then it becomes easy to make a plan for the project.
   - The project plan, Project goal, risks, use-case model, and Project description, are made.
   - The project is checked against the milestone criteria and if it couldn't pass these criteria then the project can be either canceled or redesigned.
2. Elaboration –
   - Planning and modeling are the main ones.
   - A detailed evaluation and development plan is carried out and diminishes the risks.
   - Revise or redefine the use-case model (approx. 80%), business case, and risks.
   - Again, checked against milestone criteria and if it couldn't pass these criteria then again project can be canceled or redesigned.
   - Executable architecture baseline.

3. Construction –
   - The project is developed and completed.
   - System or source code is created and then testing is done.
   - Coding takes place.
4. Transition –
   - The final project is released to the public.
   - Transit the project from development into production.
   - Update project documentation.
   - Beta testing is conducted.
   - Defects are removed from the project based on feedback from the public.
5. Production –
   - The final phase of the model.
   - The project is maintained and updated accordingly.

Advantages:

1. It provides good documentation; it completes the process in itself.
2. It provides risk-management support.
3. It reuses the components, and hence total time duration is less.
4. Good online support is available in the form of tutorials and training.

Disadvantages:

1. Team of expert professional is required, as the process is complex.
2. Complex and not properly organized process.
3. More dependency on risk management.
4. Hard to integrate again and again.

OR

2.a) Compare Waterfall model with RAD model.

	Waterfall Model	RAD Model
1.	Waterfall model known as Classical/Traditional Model.	RAD stands for Rapid Application Development.
2.	Planning is required in early stage.	There is no such constraint in RAD model.
3.	High assurance is what it aims for.	Its objective is rapid development.
4.	There is high amount risk in waterfall model.	There is low amount risk in RAD model.
5.	In waterfall model large team size is required.	In RAD model small team size is required.
6.	Waterfall model can't handle large project.	RAD model also can't handle large project but usually it is preferred between large and small project.
7.	Any changes can be made in waterfall model only at the beginning.	Any changes can be made in RAD model anytime.
8.	The product of Waterfall model is delivered after the completion of all stages.	The product of RAD model is delivered as soon as possible.
9.	There is long waiting time for running software in waterfall model.	There is less waiting time for running software in RAD model, as its first version is released as soon as possible.
10.	Waterfall model is not compatible with the change in client requirements.	RAD model may work with the change in client requirements.
11.	It is not flexible to changes.	It is flexible to changes.
12.	Customer control over the administrator is very less.	Customer control over the administrator is more in comparison to waterfall model.

2.b) List and explain common process framework for software engineering in detail.

- The process of framework defines a small set of activities that are applicable to all types of projects.
- The software process framework is a collection of task sets.
- Task sets consist of a collection of small work tasks, project milestones, work productivity and software quality assurance points.

1. Software project tracking and control

- In this activity, the developing team accesses project plan and compares it with the predefined schedule.
- If these project plans do not match with the predefined schedule, then the required actions are taken to maintain the schedule.

2. Risk management

- Risk is an event that may or may not occur.
- If the event occurs, then it causes some unwanted outcome. Hence, proper risk management is required.

3. Software Quality Assurance (SQA)

- SQA is the planned and systematic pattern of activities which are required to give a guarantee of software quality.

  For example, during the software development meetings are conducted at every stage of development to find out the defects and suggest improvements to produce good quality software.

4. Formal Technical Reviews (FTR)

- FTR is a meeting conducted by the technical staff.
- The motive of the meeting is to detect quality problems and suggest improvements.
- The technical person focuses on the quality of the software from the customer point of view.

5. Measurement

- Measurement consists of the effort required to measure the software.
- The software cannot be measured directly. It is measured by direct and indirect measures.
- Direct measures like cost, lines of code, size of software etc.
- Indirect measures such as quality of software which is measured by some other factor. Hence, it is an indirect measure of software.

6. Software Configuration Management (SCM)

- It manages the effect of change throughout the software process.

7. Reusability management

- It defines the criteria for reuse the product.
- The quality of software is good when the components of the software are developed for certain application and are useful for developing other applications.

8. Work product preparation and production

- It consists of the activities that are needed to create the documents, forms, lists, logs and user manuals for developing a software.

3.a) What is an Agile Process? State its principles and different methods of Agile.

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments.

The most widely used Agile methods include:
• Scrum
• Lean software development
• Extreme programming
• Crystal
• Kanban
• Dynamic systems development method
• Feature-driven development

Principles:
1. Highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. It welcomes changing requirements, even late in development.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shortest timescale.
4. Build projects around motivated individuals. Give them the environment and the support they need, and trust them to get the job done.
5. Working software is the primary measure of progress.
6. Simplicity the art of maximizing the amount of work not done is essential.
7. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

3.b) List and explain various system elements of computer - based system.

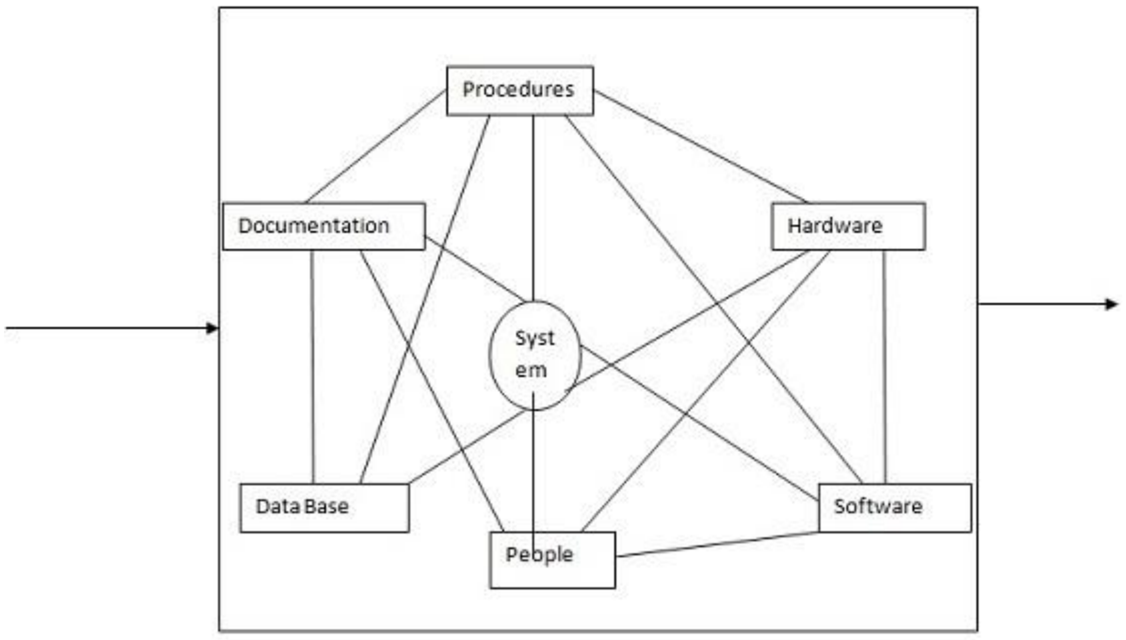
A computer-based system makes use of a variety of system elements.

- *Software: programs, data structures, and related work products.*
- *Hardware: electronic devices that provide computing capabilities.*
- *People: Users and operators of hardware and software.*
- *Database: A large, organized collection of information that is accessed via S/w and persists over time.*
- *Documentation: manuals, on-line help files.*
- *Procedures: the steps that define the specific use of each system element.*

OR

The computer-based system has been introduced in the market, many years ago. It has made its powerful place in some decades of time. That the applications were very general like airline reservation. As time pass by such systems are becoming an essential and important part of every task of any organization. The system engineers associated with such system contributed a lot to make them a great success.

A set or arrangement of elements that are organized to accomplish some method. Procedure or control by processing information. So, the computer-based systems are the collection of its various elements, as shown in figure below.



System Elements

There are total six main elements of any computer-based system. A computer-based system may be the combination of all or some of these elements. They are described below:

1. **Software**: Software may be any computer program, data structure or any associated document to accomplish a required task.
2. **Hardware**: Hardware, element may be any electronic device like CPU, memory keyboard, VDU, IC etc.
3. **Human**: Human factor may be the user or operator of the system.
4. **Database**: A large, organized collection of information which is associated with the software and can be accessed through software.
5. **Documentation**: Report form manual or other descriptive material which describes the various tasks of the system.
6. **Procedures**; A method or a sequence of steps, to be followed which describes the specific use of each system element or the system itself.

It is not necessary that all of these elements should be collectively combined in a particular manner; rather they can be combined in many ways.

One property of the computer-based systems is that even an element can act as a macro element of the large system, means it can act as a small part of a large system. So, this is the system engineer task to define the elements of the computer-based system.

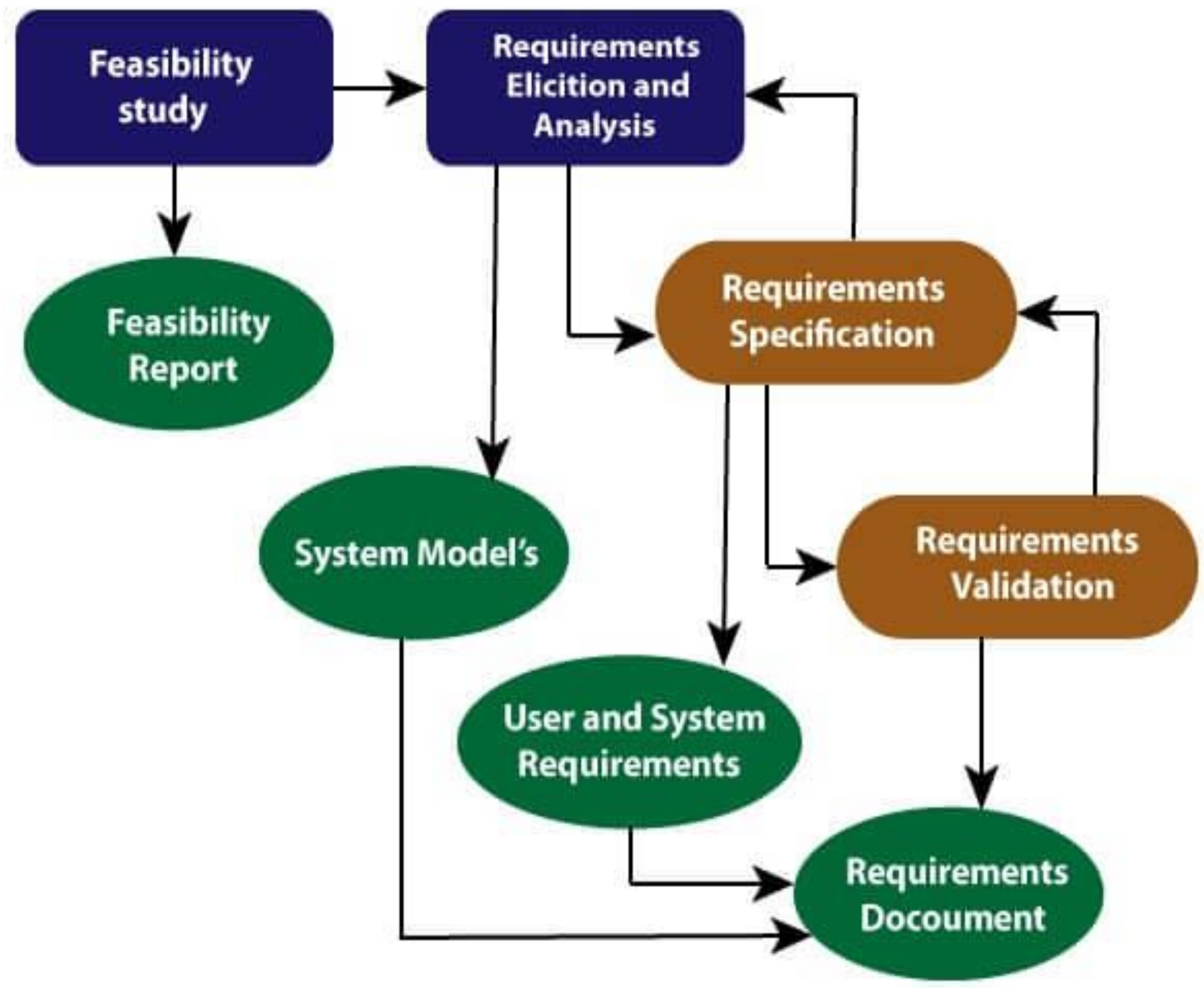
OR

4.a) Describe all the stages of Requirement Engineering in detail.

Requirement Engineering Process

It is a four-step process, which includes -

1. Feasibility Study
2. Requirement Elicitation and Analysis
3. Software Requirement Specification
4. Software Requirement Validation
5. Software Requirement Management



Requirement Engineering Process

1. Feasibility Study:

The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and conformable to established standards.

Types of Feasibility:

1. **Technical Feasibility** - Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.
2. **Operational Feasibility** - Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.
3. **Economic Feasibility** - Economic feasibility decides whether the necessary software can generate financial profits for an organization.

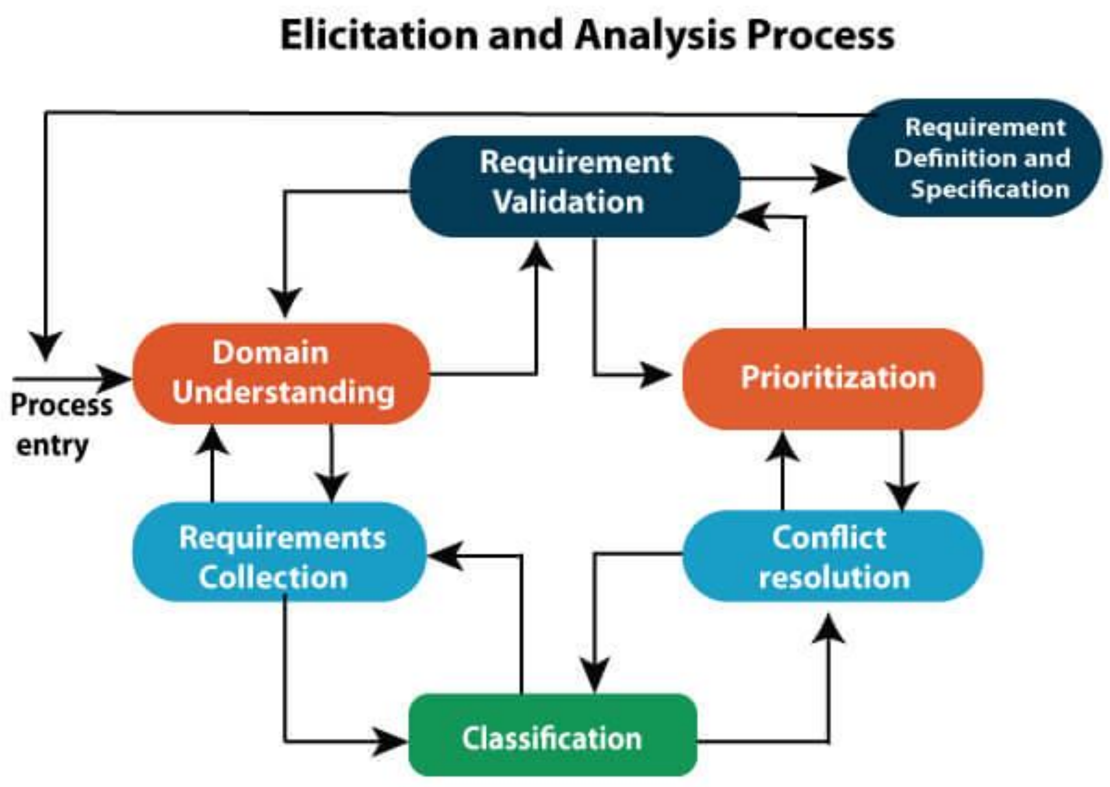
2. Requirement Elicitation and Analysis:

This is also known as the **gathering of requirements**. Here, requirements are identified with the help of customers and existing systems processes, if available.

Analysis of requirements starts with requirement elicitation. The requirements are analyzed to identify inconsistencies, defects, omission, etc. We describe requirements in terms of relationships and also resolve conflicts if any.

Problems of Elicitation and Analysis

- Getting all, and only, the right people involved.
- Stakeholders often don't know what they want
- Stakeholders' express requirements in their terms.
- Stakeholders may have conflicting requirements.
- Requirement change during the analysis process.
- Organizational and political factors may influence system requirements.



3. Software Requirement Specification:

Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources - the requirement received by the customer written in ordinary language. It is the job of the analyst to write the requirement in technical language so that they can be understood and beneficial by the development team.

The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc.

- **Data Flow Diagrams:** Data Flow Diagrams (DFDs) are used widely for modeling the requirements. DFD shows the flow of data through a system. The system may be a company, an organization, a set of procedures, a computer hardware system, a software system, or any combination of the preceding. The DFD is also known as a data flow graph or bubble chart.
- **Data Dictionaries:** Data Dictionaries are simply repositories to store information about all data items defined in DFDs. At the requirements stage, the data dictionary should at least define customer data items, to ensure that the customer and developers use the same definition and terminologies.
- **Entity-Relationship Diagrams:** Another tool for requirement specification is the entity-relationship diagram, often called an "***E-R diagram***." It is a detailed logical representation of the data for the organization and uses three main constructs i.e., data entities, relationships, and their associated attributes.

4. Software Requirement Validation:

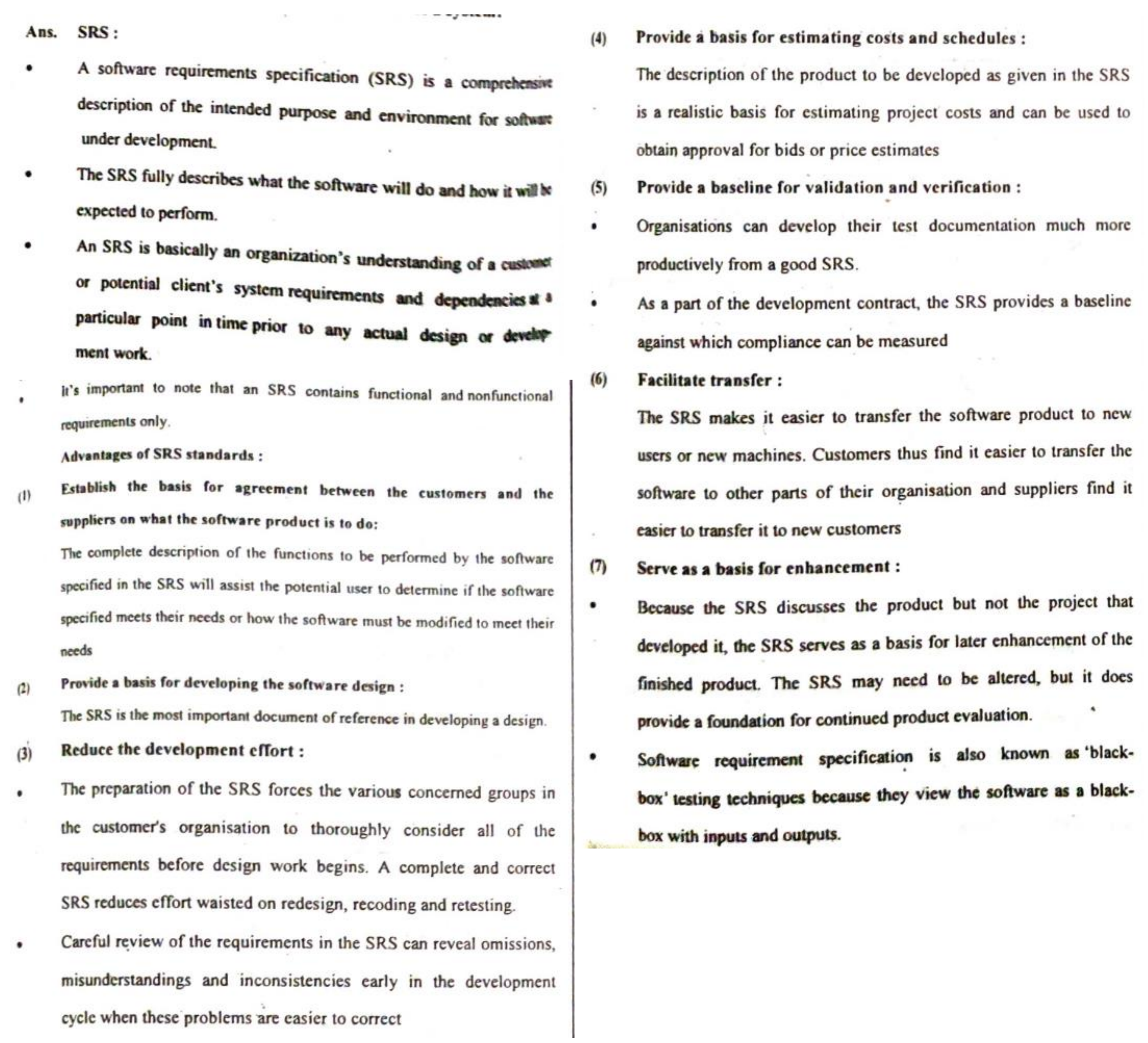
After requirement specifications developed, the requirements discussed in this document are validated. The user might demand illegal, impossible solution or experts may misinterpret the needs. Requirements can be the check against the following conditions:

- If they can practically implement
- If they are correct and as per the functionality and specially of software
- If there are any ambiguities
- If they are full
- If they can describe

Requirements Validation Techniques

- **Requirements reviews/inspections:** systematic manual analysis of the requirements.
- **Prototyping:** Using an executable model of the system to check requirements.
- **Test-case generation:** Developing tests for requirements to check testability.
- **Automated consistency analysis:** checking for the consistency of structured requirements descriptions.

4.b) What is SRS? State significance of SRS.



Ans. SRS :

- A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development.
- The SRS fully describes what the software will do and how it will be expected to perform.
- An SRS is basically an organization's understanding of a customer or potential client's system requirements and dependencies at a particular point in time prior to any actual design or development work.
- It's important to note that an SRS contains functional and nonfunctional requirements only.

Advantages of SRS standards :

(1) **Establish the basis for agreement between the customers and the suppliers on what the software product is to do:**

The complete description of the functions to be performed by the software specified in the SRS will assist the potential user to determine if the software specified meets their needs or how the software must be modified to meet their needs

(2) **Provide a basis for developing the software design :**

The SRS is the most important document of reference in developing a design.

(3) **Reduce the development effort :**

- The preparation of the SRS forces the various concerned groups in the customer's organisation to thoroughly consider all of the requirements before design work begins. A complete and correct SRS reduces effort waisted on redesign, recoding and retesting.
- Careful review of the requirements in the SRS can reveal omissions, misunderstandings and inconsistencies early in the development cycle when these problems are easier to correct

(4) **Provide a basis for estimating costs and schedules :**

The description of the product to be developed as given in the SRS is a realistic basis for estimating project costs and can be used to obtain approval for bids or price estimates

(5) **Provide a baseline for validation and verification :**

- Organisations can develop their test documentation much more productively from a good SRS.
- As a part of the development contract, the SRS provides a baseline against which compliance can be measured

(6) **Facilitate transfer :**

The SRS makes it easier to transfer the software product to new users or new machines. Customers thus find it easier to transfer the software to other parts of their organisation and suppliers find it easier to transfer it to new customers

(7) **Serve as a basis for enhancement :**

- Because the SRS discusses the product but not the project that developed it, the SRS serves as a basis for later enhancement of the finished product. The SRS may need to be altered, but it does provide a foundation for continued product evaluation.
- Software requirement specification is also known as 'black-box' testing techniques because they view the software as a black-box with inputs and outputs.

4.c) What is QFD? What are various types of requirements defined in QFD?

Quality Function Deployment (QFD) is process or set of tools used to define the customer requirements for product and convert those requirements into engineering specifications and plans such that the customer requirements for that product are satisfied.

QFD Processes

There are five important points to QFD which enable it to understand and develop products to suit consumers. It must be practical to produce and at the same time, to provide a competitive advantage:

- Understanding Customer Requirement
- Quality System Thinking + Psychology + Knowledge/Epistemology
- Maximizing Positive Quality That Adds Value
- Comprehensive Quality System for Customer Satisfaction
- Strategy to Stay Ahead of The Game

Quality Functional Deployment consists of four main steps:

1. Identify the customer's vital requirements for the product or service and translate them into design requirements.
2. Develop a service blueprint of an elegant, effective, and efficient delivery process
3. Evaluate alternative designs
4. Implement the newly designed process for delivery of the product or service

The Methodologies of QFD can be quite involved, but are based on the seven parts of the "House of Quality":

1. Customer requirements
2. Importance Weighting & Competitive Evaluation
3. Technical requirements
4. Interrelationships
5. Roof
6. Targets
7. Competition / Importance

5.a) Discuss Object Oriented Modeling in detail.

Intention of object-oriented modeling and design is to learn how to apply object - oriented concepts to all the stages of the software development life cycle. Object-oriented modeling and design is a way of thinking about problems using models organized around real-world concepts. The fundamental construct is the object, which combines both data structure and behavior.

Purpose of Models:

1. Testing a physical entity before building it
2. Communication with customers
3. Visualization
4. Reduction of complexity

Types of Models:

There are 3 types of models in the object-oriented modeling and design are: Class Model, State Model, and Interaction Model. These are explained as following below.

1. **Class Model:**
   The class model shows all the classes present in the system. The class model shows the attributes and the behavior associated with the objects. The class diagram is used to show the class model. The class diagram shows the class name followed by the attributes followed by the functions or the methods that are associated with the object of the class. Goal in constructing class model is to capture those concepts from the real world that are important to an application.
2. **State Model:**
   State model describes those aspects of objects concerned with time and the sequencing of operations – events that mark changes, states that define the context for events, and the organization of events and states. Actions and events in a state diagram become operations on objects in the class model. State diagram describes the state model.
3. **Interaction Model:**
   Interaction model is used to show the various interactions between objects, how the objects collaborate to achieve the behavior of the system as a whole.
   The following diagrams are used to show the interaction model:
   - Use Case Diagram
   - Sequence Diagram
   - Activity Diagram

5.b) Define following software design concepts:

1) Abstraction 2) Pattern 3) Modularity 4) Information Hiding

1. **Abstraction- hide Irrelevant data**
   An abstraction is a tool that enables a designer to consider a component at an abstract level without bothering about the internal details of the implementation. Abstraction can be used for existing element as well as the component being designed.

   Here, there are two common abstraction mechanisms

   1. Functional Abstraction
   2. Data Abstraction.
2. **Pattern- a repeated form**
   The pattern simply means a repeated form or design in which the same shape is repeated several times to form a pattern. The pattern in the design process means the repetition of a solution to a common recurring problem within a certain context.

3. **Modularity- subdivide the system**
   Modularity simply means dividing the system or project into smaller parts to reduce the complexity of the system or project. In the same way, modularity in design means subdividing a system into smaller parts so that these parts can be created independently and then use these parts in different systems to perform different functions. It is necessary to divide the software into components known as modules because nowadays there are different software available like Monolithic software that is hard to grasp for software engineers.
   So, modularity in design has now become a trend and is also important. If the system contains fewer components, then it would mean the system is complex which requires a lot of effort (cost) but if we are able to divide the system into components then the cost would be small.
4. **Information Hiding- hide the information**
   Information hiding simply means to hide the information so that it cannot be accessed by an unwanted party. In software design, information hiding is achieved by designing the modules in a manner that the information gathered or contained in one module is hidden and can't be accessed by any other modules.

OR

6.a) What are the components of Analysis Modelling? Discuss Scenario based modeling.

4 Essential Components of an Analytics Model

1. Data Component

This component consists of everything about data and includes the following:

i) Sources of Data This section deals with all sources of data such as

a) design of experiments or surveys for collecting data
b) purchasing data from organizations that mine and store large datasets
c) use of open dataset
d) simulating raw data to combine it with actual sampled data

ii) Data Preparation and Transformation

This deals with preprocessing raw data to convert it into a form that is ready for analysis or model building and includes topics such as

a) handling missing data
b) data imputation
c) encoding categorical data
d) identification of predictor features and target features

e) data scaling, for example feature standardization or normalization
f) feature selection and dimensionality reduction
g) advanced methods for data transformation such as PCA and LDA

Software that can be used for data preparation and transformation include

- Pandas package
- Excel
- R
- Python

2. Algorithm Component
These are algorithms that are applied to data in order to derive useful and insightful information from the data. The algorithms could be categorized as descriptive, predictive, or prescriptive.

i) Algorithms for Descriptive Analytics
These include packages that can be applied to data for visualization purposes, for example algorithms to produce barplots, line graphs, histograms, scatter plots, pairplots, density plots, qqplots, etc. Some of the most common packages for descriptive analytics include
a) Matplotlib
b) Ggplot2
c) Seaborn

ii) Algorithms for Predictive Analytics
These are algorithms that are used for building predictive models. Some of the most common packages for predictive analytics include

- Sci-kit learn package
- Caret package
- Tensorflow

iii) Algorithms for Prescriptive Analytics
These are algorithms than can be used for prescribing a course of active based on insights derived from data. Some prescriptive analytics algorithms include
a) Probabilistic modeling
c) Optimization methods and operations research
c) Monte-carlo simulations

3. Real World Component

Real world data science projects could be obtained through the following:

a) Kaggle Projects
b) Internships
c) From Interviews

Working on a real-world data science project would enable you to deepen your understanding of the model building workflow, from problem framing, to data analysis, model building & testing, and model application. It would also help you to pick up additional essential skills such as

- Communication skills
- Team player skills
- Presentation skills
- Business acumen

4. Ethical Component

This is the component that has to deal with maintaining high standards of ethics as practicing data scientists. It is important that as a data scientist, you understand the implication of your project outcomes and findings. Be truthful to yourself. Avoid manipulating data or using a method that will intentionally produce bias in results. Be ethical in all phases from data collection, to analysis, model building, testing and application. Avoid fabricating results for the purpose of misleading or manipulating your audience or supervisor. Be ethical in the way you interpret the findings from your data science project.

Scenario Modeling

Scenario modeling examines a range of potential futures, instead of attempting to predict just one future. While you don't have structured data on future performance, like you do with the past, you can use inputs and scenarios to see possible trends that you may encounter in the next few years or decades.
Scenario modeling explores the differences between these possible futures and facilitates investigations into how decisions around those situations would directly and indirectly impact the organization. *It's more about answering "what-if" than "what".*
Scenario modeling differs from conventional forecasting because:

- **Conventional forecasting** analyzes the undertaking of specific projects to identify the 'best-case' scenario or 'most likely' future. You predict the impact of executing a plan versus not executing a plan.
- **Scenario modeling** doesn't just look at the 'best-case' scenario or 'most likely' future. Rather, you produce a multitude of possible futures and compare them against some common basis.

6.b) Discuss E-R model with an example.

Entity relationship (ER) models are based on the real-world entities and their relationships. It is easy for the developers to understand the system by simply looking at the ER diagram. ER models are normally represented by ER-diagrams.

Components

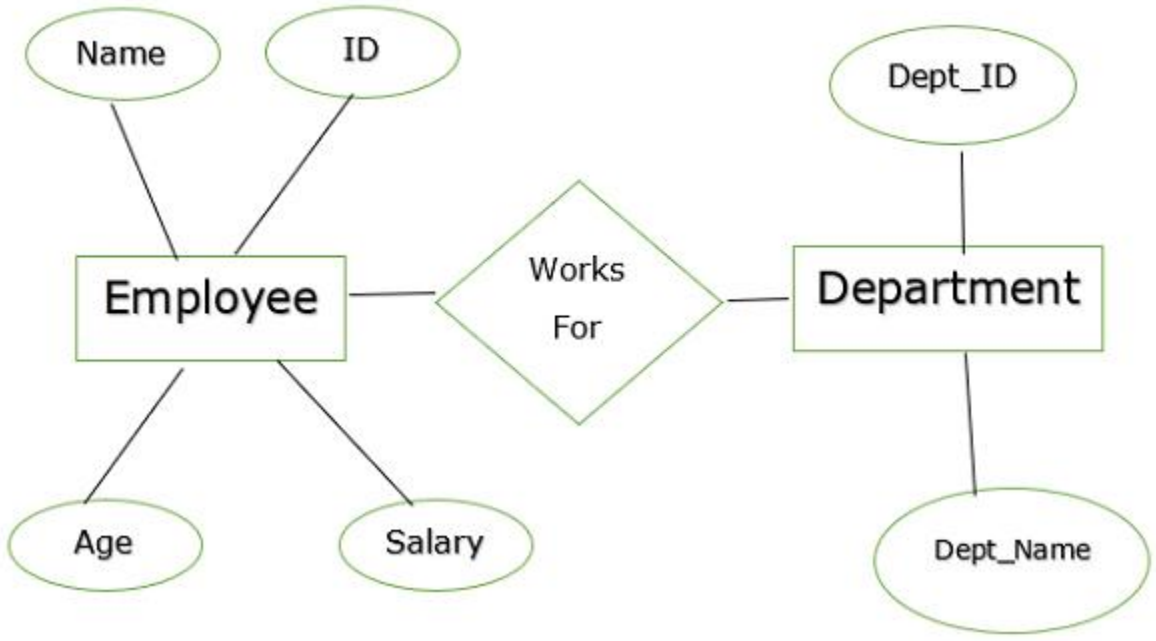
ER diagram basically having three components:

- **Entities** − It is a real-world thing which can be a person, place, or even a concept. For Example: Department, Admin, Courses, Teachers, Students, Building, etc. are some of the entities of a School Management System.
- **Attributes** − An entity which contains a real-world property called an attribute. For Example: The entity employee has the property like employee id, salary, age, etc.
- **Relationship** − Relationship tells how two attributes are related. For Example: Employee works for a department.

An entity has a real-world property called attribute and these attributes are defined by a set of values called domain.

Example

Given below is another example of ER:



In the above example,

Entities − Employee and Department.

Attributes −

- *Employee − Name, id, Age, Salary*
- *Department − Dept_id, Dept_name*

The two entities are connected using the relationship. Here, each employee works for a department.

Features of ER

The features of ER Model are as follows −

- **Graphical Representation is Better Understanding** − It is easy and simple to understand so it can be used by the developers to communicate with the stakeholders.
- **ER Diagram** − ER diagrams are used as a visual tool for representing the model.
- **Database Design** − This model helps the database designers to build the database.

Advantages

The advantages of ER are as follows −

- The ER model is easy to build.
- This model is widely used by database designers for communicating their ideas.
- This model can easily convert to any other model like network model, hierarchical model etc.
- It is integrated with the dominant relational model.

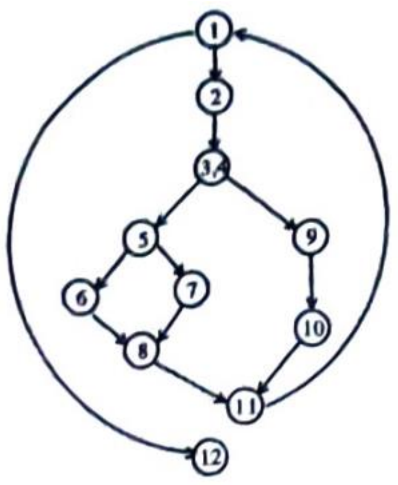
Disadvantages

The disadvantages of ER are as follows −

- There is no industry standard for developing an ER model.
- Information might be lost or hidden in the ER model.
- There is no Data Manipulation Language (DML).
- There is limited relationship representation.

7.a) What do you mean by cyclomatic complexity?

Calculate the cyclomatic complexity for the following graph.



Cyclomatic complexity is a source code complexity measurement that is being correlated to a number of coding errors. It is calculated by developing a Control Flow Graph of the code that measures the number of linearly-independent paths through a program module.

7.b) What are the different quality factors used to measure software quality? Explain.

1. **Correctness** – The extent to which a software meets its requirements specification.
2. **Efficiency** – The amount of hardware resources and code the software, needs to perform a function.
3. **Integrity** – The extent to which the software can control an unauthorized person from the accessing the data or software.
4. **Reliability** – The extent to which a software performs its intended functions without failure.
5. **Usability** – The extent of effort required to learn, operate and understand the functions of the software.
6. **Maintainability** – The effort required to detect and correct an error during maintenance phase.
7. **Flexibility** – The effort needed to improve an operational software program.
8. **Testability** – The effort required to verify a software to ensure that it meets the specified requirements.
9. **Portability** – The effort required to transfer a program from one platform to another.
10. **Re-usability** – The extent to which the program's code can be reused in other applications.
11. **Interoperability** – The effort required to integrate two systems with one another.

OR

8.a) Discuss following testing strategies.

i) Unit Testing - It focuses on the smallest unit of software design. In this, we test an individual unit or group of interrelated units. It is often done by the programmer by using sample input and observing its corresponding outputs.
Example:
a) In a program we are checking if the loop, method, or function is working fine
b) Misunderstood or incorrect, arithmetic precedence.
c) Incorrect initialization

ii) Integration Testing - The objective is to take unit-tested components and build a program structure that has been dictated by
design. Integration testing is testing in which a group of components is combined to produce output.
Integration testing is of four types: (i) Top-down (ii) Bottom-up (iii) Sandwich (iv) Big-Bang
Example:
(a) Black Box testing: - It is used for validation.
In this, we ignore internal working mechanisms and focus on what is the output?
(b) White box testing: - It is used for verification.
In this, we focus on internal mechanisms i.e., how the output is achieved?

iii) Regression Testing - Every time a new module is added leads to changes in the program. This type of testing makes sure that the whole component works properly even after adding components to the complete program.
Example: In school, record suppose we have module staff, students and finance combining these modules and checking if on integration of these modules works fine in regression testing.

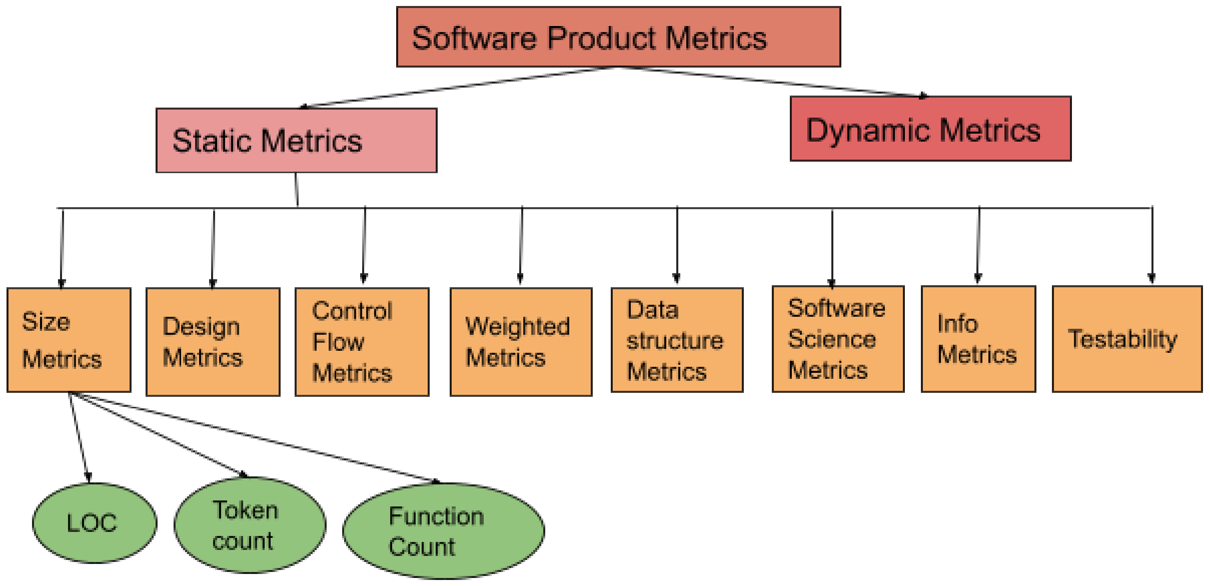
iv) Smoke Testing - This test is done to make sure that the software under testing is ready or stable for further testing. It is called a smoke test as the testing of an initial pass is done to check if it did not catch the fire or smoke in the initial switch on.
Example: If the project has 2 modules so before going to the module make sure that module 1 works properly

v) System Testing - This software is tested such that it works fine for the different operating systems. It is covered under the black box testing technique. In this, we just focus on the required input and output without focusing on internal working. In this, we have security testing, recovery testing, stress testing, and performance testing
Example: This includes functional as well as nonfunctional testing.

8.b) Illustrate size-oriented metric?

Size-Oriented Metrics concentrates on the size of the program created.
Size-oriented metrics are used to collect direct measures of software output and quality.
We can calculate Size-Oriented Metrics by averaging out quality and productivity.
The size of the program that has been generated is taken into account by Point Metrics.

For software initiatives, the organization creates a simple size measurement record. It is based on previous organizational experiences. It's a metric for determining how good a piece of software is.
Flow Chart for Size-Oriented Metrics:



Size-Oriented Metrics can also measure and compare programmers' productivity. It is a direct evaluation of a piece of software. The size estimation is based on the computation of **lines of code**. A line of code is defined as one line of text in a source file.
Set of Size Measures
The following is a simple set of size measures that can be developed:

- Quality = Errors per KLOC
- Cost = $ per KLOC
- Documentation = Pages of documentation per KLOC
- Errors = Errors per person-month
- Productivity = LOC per person-month

Here KLOC means **Thousands of Lines of Code**.

9.a) Discuss RMMM.

(1) **Risk Mitigation :**

- When the software team adopts a proactive approach to risk, avoidance is always the best strategy. This is achieved by developing a plan for risk mitigation.
- For example, assume that high staff turnover is noted as a project risk, r1. Based on past history and management intuition, the likelihood of high turnover is estimated to be 0.70 (70 percent, rather high) and the impact, x1 is projected at level 2. That is high turnover will have a critical impact on project cost and schedule.
- To mitigate this risk, project management must develop a strategy for reducing turnover. Among the possible steps to be taken are meet with current staff to determine causes for turnover.
- Mitigate those causes that are under our control before the project starts.
- Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave.
- Organize project teams so that information about each development activity is widely dispersed.
- Define documentation standards and establish mechanisms to be sure that documents are developed in a timely manner.
- Assign a backup staff member for every critical technologist.

(2) **Risk Monitoring :**

- When the project proceeds, risk monitoring activities commence.
- The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely.
- In the case of high staff turnover, the following factors can be monitored:

(i) General attitude of team members based on project pressures.

(ii) The degree to which the team has jelled.

(iii) Interpersonal relationships among team members.

(iv) Potential problems with compensation and benefits.

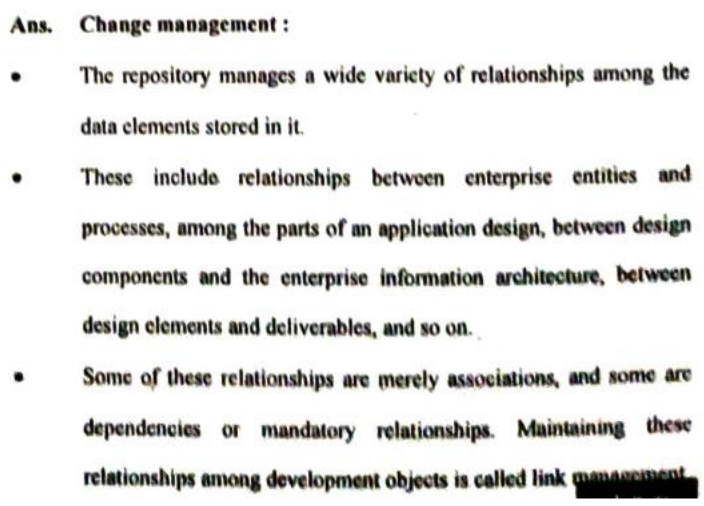
(v) The availability of jobs within the company and outside it.

(vi) In addition to monitoring these factors, the project manager should monitor the effectiveness of risk mitigation steps.

(3) **Risk management :**

- Risk management and contingency planning assumes that mitigation efforts have failed and that the risk has become a reality.
- If the mitigation strategy has been followed, backup is available, information is documented, and knowledge has been dispersed across the team.
- In addition, the project manager may temporarily refocus resources to those functions that are fully staffed, enabling newcomers who must be added to the team to "get up to speed."
- It is important to note that RMMM steps incur additional project cost. For example, spending the time to "backup" every critical technologist costs money.
- Part of risk management is to evaluate when the benefits accrued by the RMMM steps are outweighed by the costs associated with implementing them.
- In essence, the project planner performs a classic cost/benefit analysis.
- If risk aversion steps for high turnover will increase both project cost and duration by an estimated 15 percent, but the predominant cost factor is "backup" management may decide not to implement this step.
- For a large project, 30 or 40 risks may identified. If between three and seven risk management steps are identified for each, risk management may become a project in itself. For this reason, we adapt the Pareto 80–20 rule to software risk.

9.b) What is Change Management? Discuss Software Configuration Management.



Ans. Change management :

- The repository manages a wide variety of relationships among the data elements stored in it.
- These include relationships between enterprise entities and processes, among the parts of an application design, between design components and the enterprise information architecture, between design elements and deliverables, and so on.
- Some of these relationships are merely associations, and some are dependencies or mandatory relationships. Maintaining these relationships among development objects is called link [redacted]

System Configuration Management (SCM) is an arrangement of exercises which controls change by recognizing the items for change, setting up connections between those things, making/characterizing instruments for overseeing diverse variants, controlling the changes being executed in the current framework, inspecting and revealing/reporting on the changes made. It is essential to control the changes in light of the fact that if the changes are not checked legitimately then they may wind up undermining a well-run programming. In this way, SCM is a fundamental piece of all project management activities.

Processes involved in SCM –
Configuration management provides a disciplined environment for smooth control of work products. It involves the following activities:

1. **Identification and Establishment**
2. **Version control**
3. **Change control**
4. **Configuration auditing**
5. **Reporting**

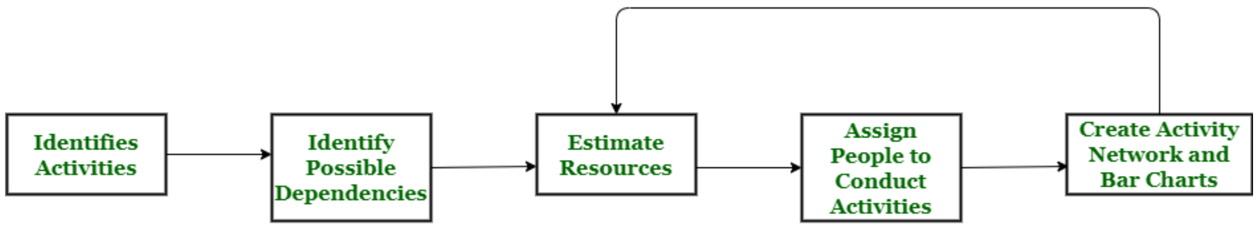
SCM Tools –
Different tools are available in market for SCM like: CFEngine, Bcfg2 server, Vagrant, SmartFrog, CLEAR CASETOOL (CC), SaltStack, CLEAR QUEST TOOL, Puppet, SVN- Subversion, Perforce, TortoiseSVN, IBM Rational team concert, IBM Configuration management version management, Razor, Ansible, etc. There are many more in the list.
It is recommended that before selecting any configuration management tool, have a proper understanding of the features and select the tool which best suits your project needs and be clear with the benefits and drawbacks of each before you choose one to use.

9.c) What is Project Scheduling? Explain the quality factors of project scheduling.

A schedule in your project's time table actually consists of sequenced activities and milestones that are needed to be delivered under a given period of time.

Project schedule simply means a mechanism that is used to communicate and know about that tasks are needed and has to be done or performed and which organizational resources will be given or allocated to these tasks and in what time duration or time frame work is needed to be performed. Effective project scheduling leads to success of project, reduced cost, and increased customer satisfaction.

Scheduling in project management means to list out activities, deliverables, and milestones within a project that are delivered. It contains more notes than your average weekly planner notes. The most common and important form of project schedule is Gantt chart.


Identifies Activities
Identify Possible Dependencies
Estimate Resources
Assign People to Conduct Activities
Create Activity Network and Bar Charts

Project Scheduling Process

Process:
The manager needs to estimate time and resources of project while scheduling project. All activities in project must be arranged in a coherent sequence that means activities should be arranged in a logical and well-organized manner for easy to understand. Initial estimates of project can be made optimistically which means estimates can be made when all favorable things will happen and no threats or problems take place.

Problems arise during Project Development Stage:

- People may leave or remain absent during particular stage of development.
- Hardware may get failed while performing.
- Software resource that is required may not be available at present, etc.

The project schedule is represented as set of charts in which work-breakdown structure and dependencies within various activities are represented. To accomplish and complete project within a given schedule, required resources must be available when they are needed. Therefore, resource estimation should be done before starting development.

Resources required for Development of Project:

- Human effort
- Sufficient disk space on server
- Specialized hardware
- Software technology
- Travel allowance required by project staff, etc.
-

Advantages of Project Scheduling:
There are several advantages provided by project schedule in our project management:

- It simply ensures that everyone remains on same page as far as tasks get completed, dependencies, and deadlines.
- It helps in identifying issues early and concerns such as lack or unavailability of resources.
- It also helps to identify relationships and to monitor process.
- It provides effective budget management and risk mitigation.

OR

10.a) What are Risks in Software development? Explain Risk Projection.

Software development is a multi-stage approach of design, documentation, programming, prototyping, testing etc. which follows a Software Development Life Cycle (SDLC) process. Different tasks are performed based on SDLC framework during software development. Developing and Maintaining software project involves risk in each step.
Risk and importance of risk management:
Risk is uncertain events associated with future events which have a probability of occurrence but it may or may not occur and if occurs it brings loss to the project. Risk identification and management are very important task during software project development because success and failure of any software project depends on it.
Various Kinds of Risks in Software Development:

1. **Schedule Risk:**
   Schedule related risks refers to time related risks or project delivery related planning risks. The wrong schedule affects the project development and delivery.
   Some reasons for Schedule risks –
   - Time is not estimated perfectly
   - Improper resource allocation
   - Tracking of resources like system, skill, staff etc.
   - Frequent project scope expansion
   - Failure in function identification and its' completion

2. **Budget Risk:**
   Budget related risks refers to the monetary risks mainly it occurs due to budget overruns.
   Some reasons for Budget risks –
   - Wrong/Improper budget estimation
   - Unexpected Project Scope expansion
   - Mismanagement in budget handling
   - Cost overruns
   - Improper tracking of Budget
3. **Operational Risks :**
   Operational risk refers to the procedural risks means these are the risks which happen in day-to-day operational activities during project development due to improper process implementation or some external operational risks.
   Some reasons for Operational risks –
   - Insufficient resources
   - Conflict between tasks and employees
   - Improper management of tasks
   - No proper planning about project
   - Less number of skilled people
   - Lack of communication and cooperation
   - Lack of clarity in roles and responsibilities
   - Insufficient training
4. **Technical Risks :**
   Technical risks refers to the functional risk or performance risk which means this technical risk mainly associated with functionality of product or performance part of the software product.
5. **Some reasons for technical risks –**
   - Frequent changes in requirement
   - Less use of future technologies
   - Less number of skilled employees
   - High complexity in implementation
   - Improper integration of modules
6. **Programmatic Risks:**
   Programmatic risks refers to the external risk or other unavoidable risks. These are the external risks which are unavoidable in nature. These risks come from outside and it is out of control of programs.
   Some reasons for Programmatic risks –
   - Rapid development of market
   - Running out of fund / Limited fund for project development
   - Changes in Government rules/policy
   - Loss of contracts due to any reason

Risk projection, also called risk estimation, attempts to rate each risk in two ways—the likelihood or probability that the risk is real and the consequences of the problems associated with the risk, should it occur. The project planner, along with other managers and technical staff, performs four risk projection activities:

1. Establish a scale that reflects the perceived likelihood of a risk,
2. Delineate the consequences of the risk,
3. Estimate the impact of the risk on the project and the product, and
4. Note the overall accuracy of the risk projection so that there will be no misunderstandings.

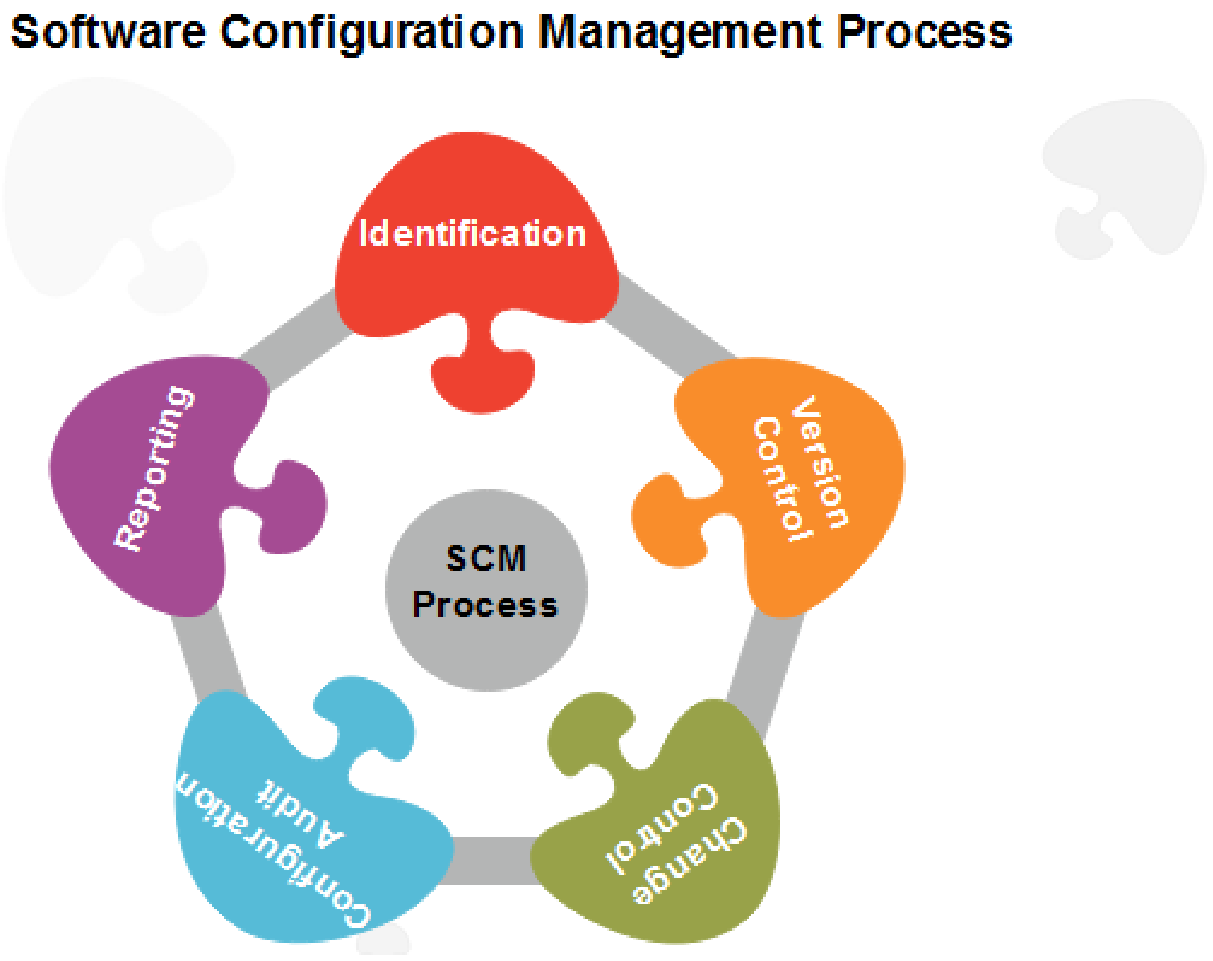
- Developing a Risk Table
- Assessing Risk Impact
- Risk Assessment

10.b) What are the Functions of an SCM Repository.

SCM Repository Functions

It uses the tools which keep that the necessary change has been implemented adequately to the appropriate component.

The SCM process defines a number of tasks:



1. **Identification**
   Basic Object: Unit of Text created by a software engineer during analysis, design, code, or test.
   Aggregate Object: A collection of essential objects and other aggregate objects. Design Specification is an aggregate object.
   The interrelationships between configuration objects can be described with a Module Interconnection Language (MIL).

2. **Version Control**
   Version Control combines procedures and tools to handle different version of configuration objects that are generated during the software process.

3. **Change Control**
   Change Control is Vital. But the forces that make it essential also make it annoying.
   The "check-in" and "check-out" process implements two necessary elements of change control-**access control** and **synchronization control**.
   Access Control governs which software engineers have the authority to access and modify a particular configuration object.
   Synchronization Control helps to ensure that parallel changes, performed by two different people, don't overwrite one another.

4. **Configuration Audit**
   SCM audits to verify that the software product satisfies the baselines requirements and ensures that what is built and what is delivered.

5. **Status Reporting**
   Configuration Status reporting (sometimes also called status accounting) providing accurate status and current configuration data to developers, testers, end users, customers and stakeholders through admin guides, user guides, FAQs, Release Notes, Installation Guide, Configuration Guide, etc.

10.c) Explain with example, how FP based estimation is performed.

Function Point Analysis (FPA) is a method or set of rules of Functional Size Measurement. It assesses the functionality delivered to its users, based on the user's external view of the functional requirements. It measures the logical view of an application, not the physically implemented view or the internal technical view.

The Function Point Analysis technique is used to analyze the functionality delivered by software and Unadjusted Function Point (UFP) is the unit of measurement.

Example: Compute the function point, productivity, documentation, cost per function for the following data:

1. Number of user inputs = 24
2. Number of user outputs = 46
3. Number of inquiries = 8
4. Number of files = 4
5. Number of external interfaces = 2
6. Effort = 36.9 p-m
7. Technical documents = 265 pages
8. User documents = 122 pages
9. Cost = $7744/ month

Various processing complexity factors are: 4, 1, 0, 3, 3, 5, 4, 4, 3, 3, 2, 2, 4, 5.

Solution:

Measurement Parameter	Count		Weighing factor
1. Number of external inputs (EI)	24	*	4 = 96
2. Number of external outputs (EO)	46	*	4 = 184
3. Number of external inquiries (EQ)	8	*	6 = 48
4. Number of internal files (ILF)	4	*	10 = 40
5. Number of external interfaces (EIF) Count-total →	2	*	5 = 10 378

So, sum of all f_i (i ← 1 to 14) = 4 + 1 + 0 + 3 + 5 + 4 + 4 + 3 + 3 + 2 + 2 + 4 + 5 = 43

FP = Count-total * [0.65 + 0.01 *$\sum(f_i)$]
= 378 * [0.65 + 0.01 * 43]
= 378 * [0.65 + 0.43]
= 378 * 1.08 = 408

$$\text{Productivity} = \frac{\text{FP}}{\text{Effort}} = \frac{408}{36.9} = 11.1$$

Total pages of documentation = technical document + user document
= 265 + 122 = 387pages

Documentation = Pages of documentation/FP
= 387/408 = 0.94

$$\text{Cost per function} = \frac{\text{cost}}{\text{productivity}} = \frac{7744}{11.1} = \$700$$