Software Engineering Questions

Q1. What is Software Engineering? What are 4 differences and 4 similarities between software engineering and conventional engineering?

Ans. Software Engineering is the process of designing, developing, testing, and maintaining software. It is a systematic and disciplined approach to software development that aims to create high-quality, reliable, and maintainable software.

1. Software engineering includes a variety of techniques, tools, and methodologies, including requirements analysis, design, testing, and maintenance.
2. It is a rapidly evolving field, and new tools and technologies are constantly being developed to improve the software development process.
3. By following the principles of software engineering and using the appropriate tools and methodologies, software developers can create high-quality, reliable, and maintainable software that meets the needs of its users.
4. Software Engineering is mainly used for large projects based on software systems rather than single programs or applications.
5. The main goal of Software Engineering is to develop software applications for improving quality, budget, and time efficiency.
6. Software Engineering ensures that the software that has to be built should be consistent, correct, also on budget, on time, and within the required requirements.

Key Principles of Software Engineering

1. Modularity
2. Abstraction
3. Encapsulation
4. Reusability
5. Maintenance
6. Testing
7 .Design Patterns
8. Agile methodologies
9. Continuous Integration & Deployment

Main Attributes of Software Engineering

1. Efficiency
2. Reliability
4. Reusability
5. Maintainability

Here are four differences and four similarities between software engineering and conventional engineering:

Differences

1. **Nature of Products**:
   - **Software Engineering**: Produces intangible products (software applications, systems).
   - **Conventional Engineering**: Produces tangible products (buildings, machinery, infrastructure).
2. **Development Process**:
   - **Software Engineering**: Often follows iterative and agile methodologies, allowing for flexibility and continuous improvement.
   - **Conventional Engineering**: Typically follows a more linear and structured approach (such as the waterfall model), with clear phases like design, construction, and testing.
3. **Testing and Quality Assurance**:
   - **Software Engineering**: Testing can be done at various stages, and bugs can often be fixed post-deployment through updates.
   - **Conventional Engineering**: Testing is usually more rigorous before deployment, as physical structures must meet safety and regulatory standards.

4. **Tools and Technologies**:
   - **Software Engineering**: Utilizes programming languages, development frameworks, and software tools (IDEs, version control systems).
   - **Conventional Engineering**: Uses physical tools and materials (CAD software, construction tools, physical modeling).

Similarities

1. **Problem-Solving Focus**:
   - Both fields emphasize solving complex problems through systematic approaches and methodologies.
2. **Project Management**:
   - Both require effective project management skills to ensure timely delivery and adherence to budgets and specifications.
3. **Collaboration**:
   - Both involve teamwork and collaboration among various stakeholders, including engineers, clients, and other professionals.
4. **Regulatory Compliance**:
   - Both fields must adhere to standards and regulations relevant to their respective industries, ensuring safety, quality, and functionality of the final product.

Q2. Explain software components and characteristics. What is software crisis? What are different software quality attributes?

Ans. Software is the set of instructions in the form of programs to govern the computer system and to process the hardware components. To produce a software product the set of activities is used. This set is called a software process.

Components of Software

There are three main components of the software:

1. **Program:** A computer program is a list of instructions that tell a computer what to do.
2. **Documentation:** Source information about the product contained in design documents, detailed code comments, etc.
3. **Operating Procedures:** Set of step-by-step instructions compiled by an organization to help workers carry out complex routine operations.

Other Software Components are:

1. Code	**2.Data**
3. User interface	**4.Libraries**
5. Documentation	**6. Test cases**
7. Configuration files	**8.Build and deployment scripts**
8. Metadata	

Software Crisis

The term "software crisis" refers to a set of problems that were faced by the software industry in the 1960s and 1970s, such as:

1. **Size and Cost:** Day to day growing complexity and expectation out of software. Software are more expensive and more complex.
2. **Quality:** Software products must have good quality.
3. **Delayed Delivery:** Software takes longer than the estimated time to develop, which in turn leads to cost shooting up.
4. **High costs and long development times:** software projects were taking much longer and costing much more than expected.
5. **Low quality**: software was often delivered late, with bugs and other defects that made it difficult to use.
6. **Lack of standardization**: there were no established best practices or standards for software development, making it difficult to compare and improve different approaches.
7. **Lack of tools and methodologies**: there were few tools and methodologies available to help with software development, making it a difficult and time-consuming process.

Software Quality attributes

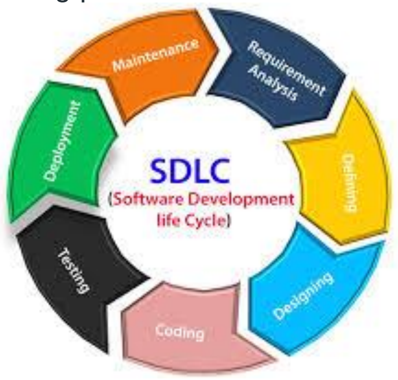
Software quality attributes are measurable or testable properties of a software system used by quality architects. These properties help to determine whether the software satisfies the stakeholders' requirements and needs. Below is the list of the most important software architecture quality attributes.

Portability: A software device is said to be portable, if it can be freely made to work in various operating system environments, in multiple machines, with other software products, etc.
Usability: A software product has better usability if various categories of users can easily invoke the functions of the product.
Reusability: A software product has excellent reusability if different modules of the product can quickly be reused to develop new products.
Correctness: A software product is correct if various requirements as specified in the SRS document have been correctly implemented.
Maintainability: A software product is maintainable if bugs can be easily corrected as and when they show up, new tasks can be easily added to the product, and the functionalities of the product can be easily modified, etc.

Q3. What do you mean by SDLC? Describe different phases of SDLC.

Ans. Software development life cycle (SDLC) is a structured process that is used to design, develop, and test good-quality software. SDLC, or software development life cycle, is a methodology that defines the entire procedure of software development step-by-step.
There are several reasons why organizations use the Software Development Life Cycle (SDLC) when developing software applications:
1.To ensure that the software is of high quality
2.To manage risks and costs
3.To improve communication and collaboration
4.To improve efficiency and productivity
5.To increase the likelihood of a successful project outcome

The SDLC typically includes the following phases:



1. Requirements gathering and analysis: This phase involves gathering information about the software requirements from stakeholders, such as customers, end-users, and business analysts.

2. Design: In this phase, the software design is created, which includes the overall architecture of the software, data structures, and interfaces. It has two steps:

- **High-level design (HLD):** It gives the architecture of software products.
- **Low-level design (LLD):** It describes how each and every feature in the product should work and every component.

3. Implementation or coding: The design is then implemented in code, usually in several iterations, and this phase is also called as Development.
things you need to know about this phase:
- This is the longest phase in SDLC model.
- This phase consists of Front end + Middleware + Back-end.
- **In front-end:** Development of coding is done even SEO settings are done.
- **In Middleware:** They connect both the front end and back end.
- **In the back-end:** A database is created.

4. Testing: The software is thoroughly tested to ensure that it meets the requirements and works correctly.

5. Deployment: After successful testing, The software is deployed to a production environment and made available to end-users.

6. Maintenance: This phase includes ongoing support, bug fixes, and updates to the software.

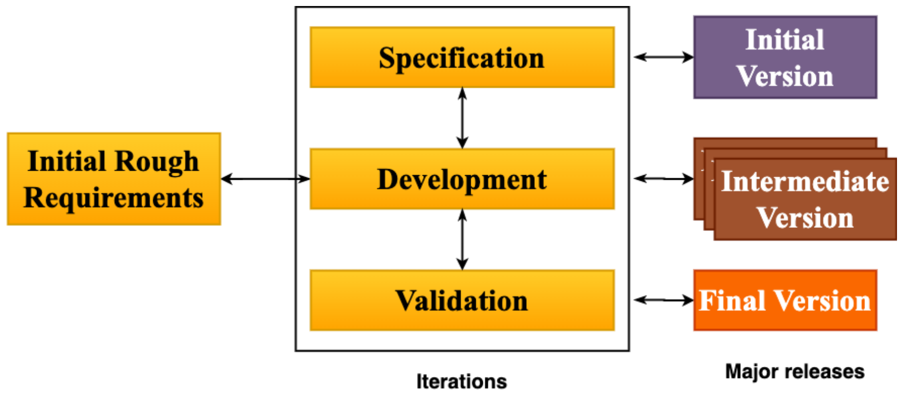
There are **different methodologies** that organizations can use to implement the SDLC, such as **Waterfall, Agile, Scrum, V-Model** and **DevOps.**

Q4. Describe the following models-

(i) Evolutionary development model
(ii) Iterative enhancement model
(iii) Agile software development model.

Ans. (i) Evolutionary development model

The Evolutionary development model divides the development cycle into smaller, incremental waterfall models in which users can get access to the product at the end of each cycle. The evolutionary model is a combination of the Iterative and Incremental models of the software development life cycle.



Advantages Evolutionary Model

1. Adaptability to Changing Requirements
2. Early and Gradual Distribution
4. User Commentary and Involvement
5. Improved Handling of Difficult Projects

Disadvantages Evolutionary Model

1. Communication Difficulties
2. Dependence on an Expert Group
3. Increasing Management Complexity
4. Greater Initial Expenditure**.**

(ii) Iterative enhancement model

In software development, the Iterative Enhancement Model stands out due to its incremental and iterative nature, it is also known as an **incremental model**. This approach focuses on incremental development and improvement rather than trying to complete a software product in a single phase. This model is based on segmenting the project into smaller units, or iterations, with a set of tasks completed.

The Iterative Model allows the accessing earlier phases, in which the variations made respectively. The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.

The various phases of Iterative model are as follows:

1. Requirement gathering & analysis
2. Design**.**
3. Implementation
4. Testing
5. Deployment
6. Review
7. Maintenance

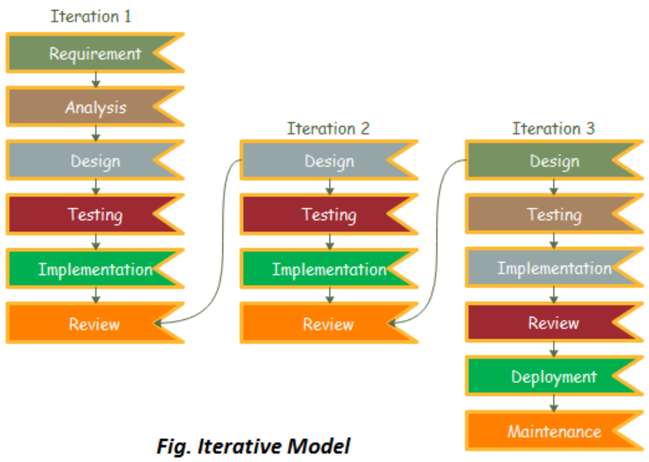


Fig. Iterative Model

When to use the Iterative Model?

1. When requirements are defined clearly and easy to understand.
2. When the software application is large.
3. When there is a requirement of changes in future.

(iii) Agile Software Development

Agile Software Development is a [software development methodology]() that values flexibility, collaboration, and customer satisfaction. It is based on the Agile Manifesto, a set of principles for software development that prioritize individuals and interactions, working software, customer collaboration, and responding to change.

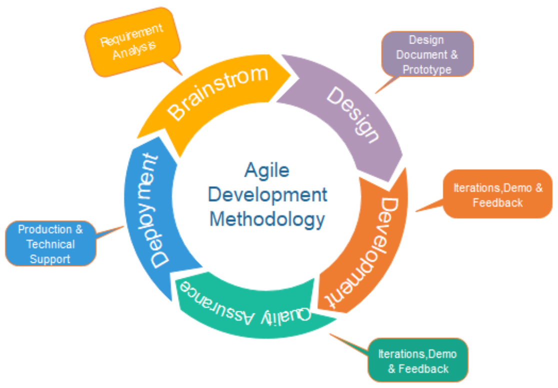

Requirement Analysis
Brainstrom
Design Document & Prototype
Design
Iterations,Demo & Feedback
Development
Iterations,Demo & Feedback
Quality Assurance
Deployment
Production & Technical Support
Agile Development Methodology

Fig. Agile Model

Phases of Agile Model:

Following are the phases in the Agile model are as follows:

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

Agile Testing Methods:

- Scrum
- Crystal
- Dynamic Software Development Method(DSDM)
- Feature Driven Development(FDD)
- Lean Software Development
- eXtreme Programming(XP)

Q5. (i) Difference between Spiral Model and Waterfall Model
(ii) Describe the Prototype model

Ans. (i) Difference between Spiral Model and Waterfall Model

There are some differences between these two models waterfall and spiral models which are given below:

Aspect	Waterfall Model	Spiral Model
Complexity	The Waterfall model is simple and easy.	The spiral model is a lot more complex.
Development Method	The waterfall model works in a sequential method.	While the spiral model works in the evolutionary method.
Risk Management	In the waterfall model errors or risks are identified and rectified after the completion of stages.	In the spiral model errors or risks are identified and rectified earlier.
Adoption	The waterfall model is adopted by customers.	While the spiral model is adopted by developers.
Project Size Suitability	The waterfall model is applicable for small projects.	While the Spiral model is used for large projects.
Planning	In waterfall model requirements and early stage planning is necessary.	While in spiral model requirements and early stage planning is necessary if required.
Flexibility to Change	Flexibility to change in waterfall model is Difficult.	Flexibility to change in spiral model is not Difficult.
Risk Level	There is high amount risk in waterfall model.	There is low amount risk in spiral model.
Cost	Waterfall model is comparatively inexpensive.	While cost of spiral model is very expensive.

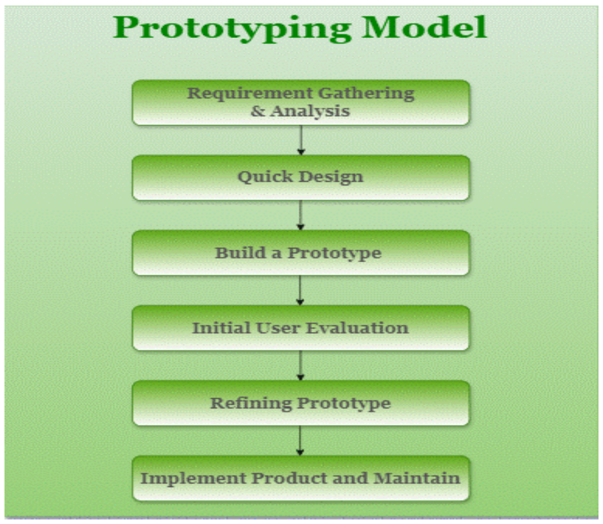
Aspect	Waterfall Model	Spiral Model
Maintenance	It requires least maintenance.	It requires typical maintenance.
Framework Type	It is based on linear framework type.	It is based on linear and iterative framework type.
Testing	Testing is done after the coding phase in the development life cycle.	Testing is done after the engineering phase in the development cycle.
Reusability	Reusability is extremely unlikely.	To a certain extent, reusability is possible.
Customer Control	Customer control over the administrator is very limited.	Customers have control over the administrator as compared to waterfall model.

(ii) Prototype model

The Prototyping Model is one of the most popularly used Software Development Life Cycle Models (SDLC models). This model is used when the customers do not know the exact project requirements beforehand. In this model, a prototype of the end product is first developed, tested, and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.

Steps of Prototyping Model

Step 1: Requirement Gathering and Analysis
Step 2: Quick Design
Step 3: Build a Prototype
Step 4: Initial User Evaluation
Step 5: Refining Prototype
Step 6: Implement Product and Maintain



Types of Prototyping Models

There are four types of Prototyping Models, which are described below.

- Rapid Throwaway Prototyping
- Evolutionary Prototyping
- Incremental Prototyping
- Extreme Prototyping

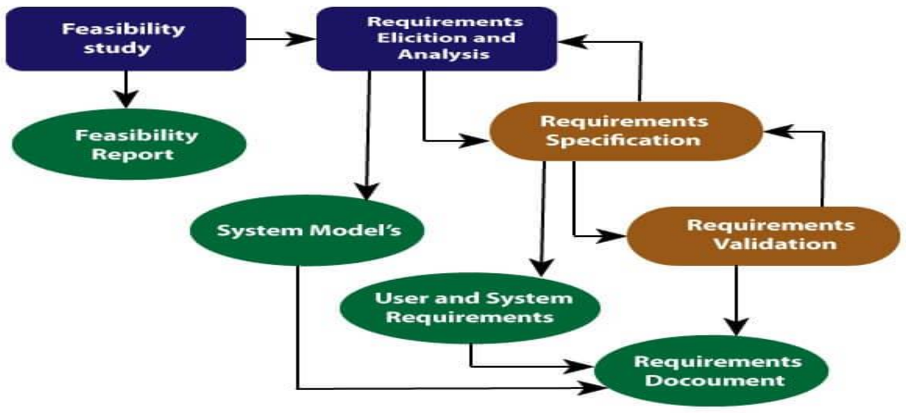
Q6. What is a requirements engineering process? Describe its various stages.

Ans. Requirements Engineering is the process of identifying, eliciting, analyzing, specifying, validating, and managing the needs and expectations of stakeholders for a software system.

Requirements Engineering Process

A systematic and strict approach to the definition, creation, and verification of requirements for a software system is known as requirements engineering. To guarantee the effective creation of a software product, the requirements engineering process entails several tasks that help in understanding, recording, and managing the demands of stakeholders.

1. **Feasibility Study**
2. **Requirements elicitation**
3. **Requirements specification**
4. **Requirements for verification and validation**
5. **Requirements management**



Requirement Engineering Process

1. Feasibility Study

A feasibility study determines whether a software project will be viable. A feasibility study aims to identify and assess a proposed project's strengths, weaknesses, opportunities, and threats to determine its potential for success.

It is of following types -

1. **Technical Feasibility**:
2. Operational Feasibility:
3. Economic Feasibility:
4. Legal Feasibility:
5. Schedule Feasibility:

2. Requirements Elicitation

. Requirements elicitation is the process of **gathering information** about the needs and expectations of stakeholders for a software system. It is related to the various ways used to gain knowledge about the project domain and requirements. The goal of this step is to understand the problem that the software system is intended to solve and the needs and expectations of the stakeholders who will use the system.

Several techniques can be used to elicit requirements, including:

- **Interviews**:
- **Surveys**:.
- **Focus Groups**:
- **Observation**:
- **Prototyping**:

3. Requirements Specification

Requirements specification is the process of documenting the requirements identified in the analysis step in a clear, consistent, and unambiguous manner. The goal of this step is to create a clear and comprehensive document that describes the requirements for the software system. This document should be understandable by both the development team and the stakeholders. The models used at this stage include **ER diagrams, data flow diagrams(DFDs), function decomposition diagrams(FDDs), data dictionaries, etc.**

4. Requirements Verification and Validation

Verification: It refers to the set of tasks that ensures that the software correctly implements a specific function.
Validation: It refers to a different set of tasks that ensures that the software that has been built is traceable to customer requirements. If requirements are not validated, errors in the requirement definitions would propagate to the successive stages resulting in a lot of modification and rework.

5. Requirements Management

Requirements management is the process of managing the requirements throughout the software development life cycle, including tracking and controlling changes, and ensuring that the requirements are still valid and relevant. The goal of requirements management is to ensure that the software system being developed meets the needs and expectations of the stakeholders and that it is developed on time, within budget, and to the required quality.

Several key activities are involved in requirements management, including:

1. Tracking and controlling changes
2. Version control:
3. Traceability: **Communication:**
4. Monitoring and reporting:

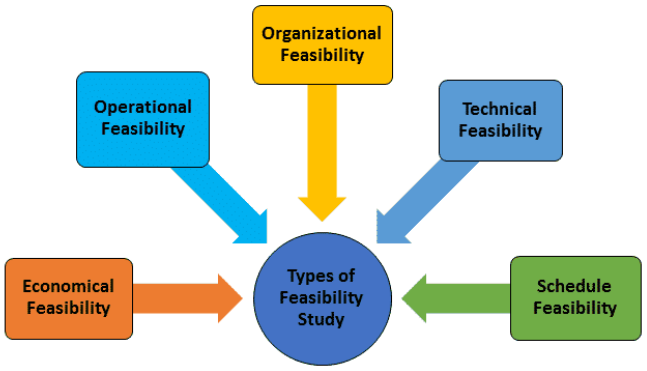
.

Q7. What is Feasibility study? Explain the types of feasibility study.

Ans. **Feasibility Study**

A feasibility study determines whether a software project will be viable. A feasibility study aims to identify and assess a proposed project's strengths, weaknesses, opportunities, and threats to determine its potential for success.

It is of following types –



1. **Technical Feasibility**: In Technical Feasibility current resources both hardware software along required technology are analyzed/assessed to develop the project. This technical feasibility study reports whether there are correct required resources and technologies that will be used for project development. Along with this, the feasibility study also analyzes the technical skills and capabilities of the technical team, whether existing technology can be used or not, whether maintenance and up-gradation are easy or not for the chosen technology, etc.

2. **Operational Feasibility:** In Operational Feasibility degree of providing service to requirements is analyzed along with how easy the product will be to operate and maintain after deployment. Along with this other operational scopes are determining the usability of the product, Determining suggested solution by the software development team is acceptable or not, etc.

3. **Economic Feasibility:** In the Economic Feasibility study cost and benefit of the project are analyzed. This means under this feasibility study a detailed analysis is carried out will be cost of the project for development which includes all required costs for final development hardware and software resources required, design and development costs operational costs, and so on. After that, it is analyzed whether the project will be beneficial in terms of finance for the organization or not.

4. **Legal Feasibility:** In legal feasibility, the project is ensured to comply with all relevant laws, regulations, and standards. It identifies any legal constraints that could impact the project and reviews existing contracts and agreements to assess their effect on the project's execution. Additionally, legal feasibility considers issues related to intellectual property, such as patents and copyrights, to safeguard the project's innovation and originality.

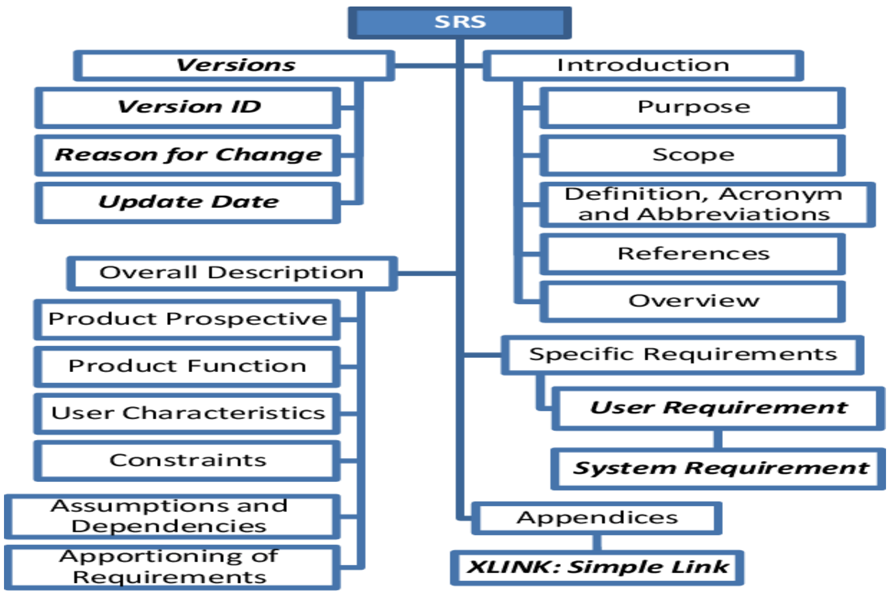
5. **Schedule Feasibility:** In schedule feasibility, the project timeline is evaluated to determine if it is realistic and achievable. Significant milestones are identified, and deadlines are established to track progress effectively. Resource availability is assessed to ensure that the necessary resources are accessible to meet the project schedule. Furthermore, any time constraints that might affect project delivery are considered to ensure timely completion.

Q8. What is a SRS document and its characteristics? What is IEEE standard for SRS?

Ans. A software requirements specification (SRS) is a document that describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfill the needs of all stakeholders (business, users).

Structure of SRS

- **Purpose of this Document –** At first, main aim of why this document is necessary and what's purpose of document is explained and described.
- **Scope of this document –** In this, overall working and main objective of document and what value it will provide to customer is described and explained. It also includes a description of development cost and time required.
- **Overview –** In this, description of product is explained. It's simply summary or overall review of product.

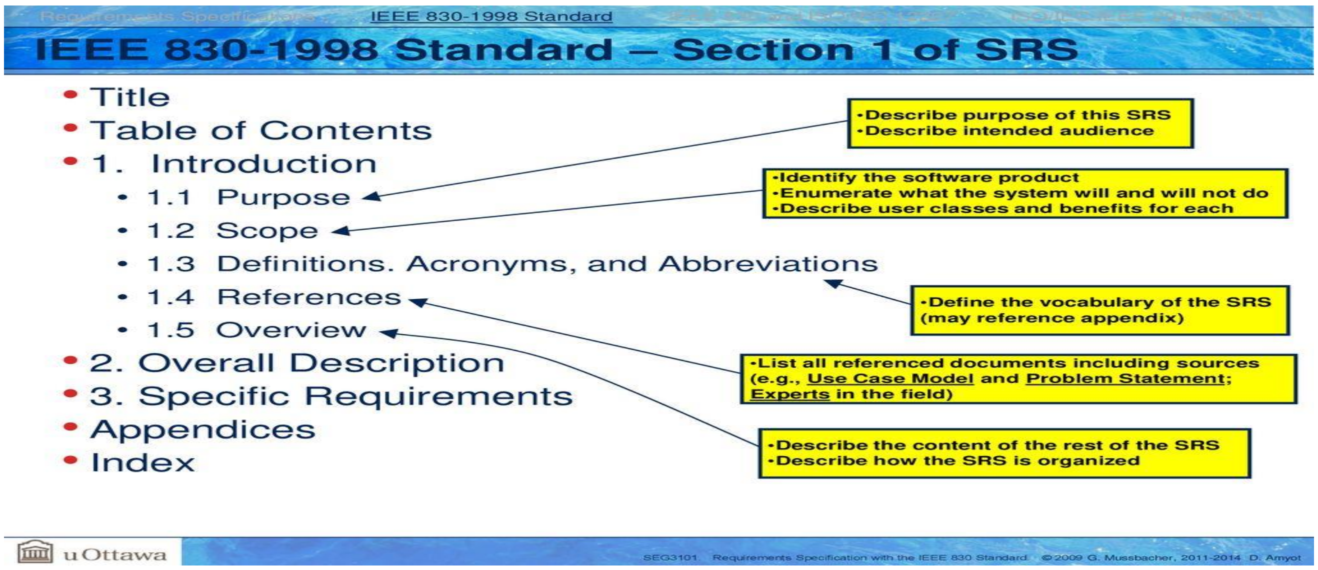


A good SRS document should have the following characteristics:

- Completeness.
- Clarity.
- Correctness.
- Consistency.
- Verifiability.
- Ranking.
- Modifiability.
- Traceability.

IEEE standard for SRS document

IEEE(The Institute of Electrical and Electronics Engineers) defines software requirements specification as, 'a document that clearly and. precisely describes each of the essential requirements (functions, performance, design. constraints and quality attributes) of the software and the external interfaces.



Requirements Specifications IEEE 830-1998 Standard

IEEE 830-1998 Standard – Section 1 of SRS

- Title
- Table of Contents
- 1. Introduction
  - 1.1 Purpose
  - 1.2 Scope
  - 1.3 Definitions. Acronyms, and Abbreviations
  - 1.4 References
  - 1.5 Overview
- 2. Overall Description
- 3. Specific Requirements
- Appendices
- Index

-Describe purpose of this SRS
-Describe intended audience

-Identify the software product
-Enumerate what the system will and will not do
-Describe user classes and benefits for each

-Define the vocabulary of the SRS
(may reference appendix)

-List all referenced documents including sources (e.g., Use Case Model and Problem Statement; Experts in the field)

-Describe the content of the rest of the SRS
-Describe how the SRS is organized

uOttawa SEG3101 Requirements Specification with the IEEE 830 Standard ©2009 G. Mussbacher, 2011-2014 D. Amyot

Q9. (i) What is a data flow diagram(DFD)? Describe type of DFD and its componets.
(ii) What is the ER diagram?

Ans. Data Flow Diagram (DFD) represents the flow of data within information systems. Data Flow Diagrams (DFD) provide a graphical representation of the data flow of a system that can be understood by both technical and non-technical users. The models enable software engineers, customers, and users to work together effectively during the analysis and specification of requirements.

Types of Data Flow Diagram (DFD)

There are two types of Data Flow Diagram (DFD)

1. Logical Data Flow Diagram
2. Physical Data Flow Diagram

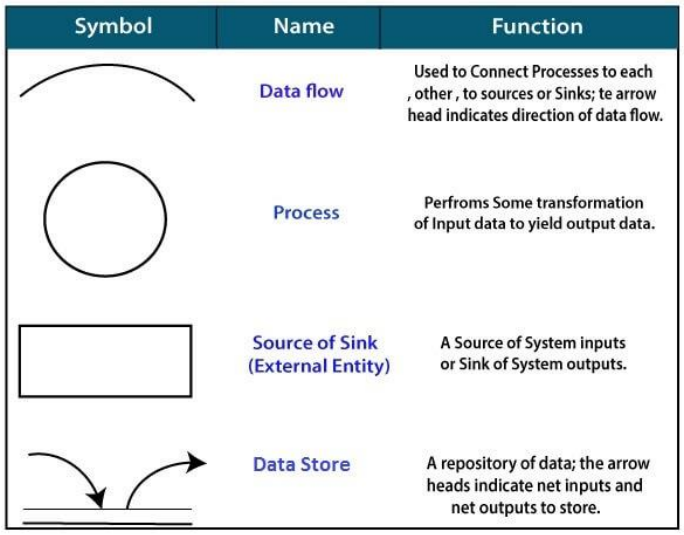
Logical Data Flow Diagram (DFD)

Logical data flow diagram mainly focuses on the system process. It illustrates how data flows in the system. Logical Data Flow Diagram (DFD) mainly focuses on high level processes and data flow without diving deep into technical implementation details.

Physical Data Flow Diagram

Physical data flow diagram shows how the data flow is actually implemented in the system. In the Physical Data Flow Diagram (DFD), we include additional details such as data storage, data transmission, and specific technology or system components. Physical DFD is more specific and close to implementation.

Components of Data Flow Diagrams (DFD)



Symbol	Name	Function
⌒	Data flow	Used to Connect Processes to each , other , to sources or Sinks; te arrow head indicates direction of data flow.
○	Process	Perfroms Some transformation of Input data to yield output data.
□	Source of Sink (External Entity)	A Source of System inputs or Sink of System outputs.
═	Data Store	A repository of data; the arrow heads indicate net inputs and net outputs to store.

Symbols for Data Flow Diagrams

Characteristics of Data Flow Diagram (DFD)

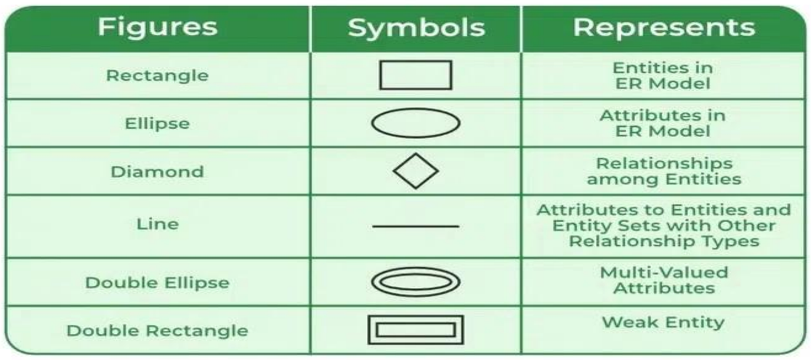
- Graphical Representation**.**
- Problem Analysis
- Abstraction
- Hierarchy
- Data Flow
- Ease of Understanding
- Modularity

Ans (ii) The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically.
The Entity Relationship Diagram explains the relationship among the entities present in the database. ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects. In short, the ER Diagram is the structural format of the database.

Symbols Used in ER Model

ER Model is used to model the logical view of the system from a data perspective which consists of these symbols:

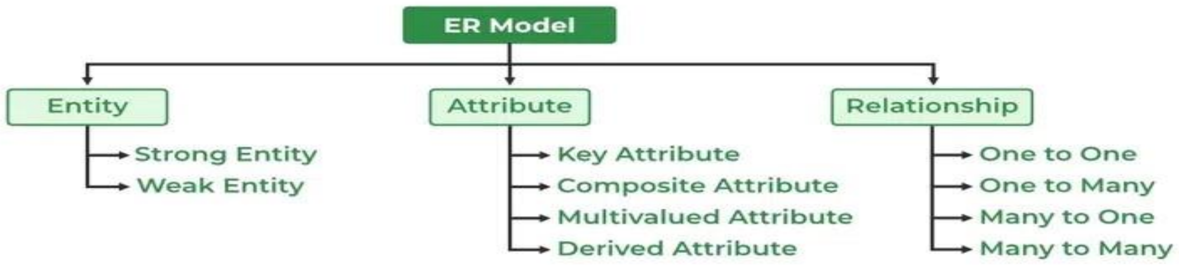
- **Rectangles:** Rectangles represent Entities in the ER Model.
- **Ellipses:** Ellipses represent Attributes in the ER Model.
- **Diamond:** Diamonds represent Relationships among Entities.
- **Lines:** Lines represent attributes to entities and entity sets with other relationship types.
- **Double Ellipse:** Double Ellipses represent Multi-Valued Attributes.
- **Double Rectangle:** Double Rectangle represents a Weak Entity.



Figures	Symbols	Represents
Rectangle	▭	Entities in ER Model
Ellipse	⬭	Attributes in ER Model
Diamond	◇	Relationships among Entities
Line	—	Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse	◎	Multi-Valued Attributes
Double Rectangle	▣	Weak Entity

Components of ER Diagram

ER Model consists of Entities, Attributes, and Relationships among Entities in a Database System.



Q10. Campare ISO 9000 model and SEI-CMM model.

Ans. ISO 9000 model

ISO9000 is an international standard of quality management and quality assurance. It certifies the companies that they are documenting the quality system elements which are needed to run an efficient and quality system.

SEI-CMM model

SEI (Software Engineering Institute) - Capability Maturity Model (CMM) is specifically for software organizations to certify them at which level, they are following and maintaining the quality standards.

Following are the important differences between ISO9000 and SEI-CMM.

Sr. No.	Key	ISO9000	SEI-CMM.
1	Definition	ISO9000 is an international standard of quality management and quality assurance. It certifies the companies that they are documenting the quality system elements which are needed to run a efficient and quality system.	SEI-CMM is specifically for software organizations to certify them at which level, they are following and maintaining the quality standards.
2	Focus	Focus of ISO9000 is on customer supplier relationship, and to reduce the customer's risk.	Focus of SEI-CMM is to improve the processes to deliver a quality software product to the customer.
3	Target Industry	ISO9000 is used by manufacturing industries.	SEI-CMM is used by software industry.
4	Recognition	ISO9000 is univesally accepted accross lots of countries.	SEI-CMM is mostly used in USA.
5	Guidelines	ISO9000 guides about concepts, priciples and safeguards to be in place in a workplace.	SEI-CMM specifies what is to be followed at what level of maturity.
6	Levels	ISO9000 has one acceptance level.	SEI-CMM has five acceptance levels.
7	Validity	ISO9000 certificate is valid for three years.	SEI-CMM certificate is valid for three years as well.
8	Level	ISO9000 has no levels.	SEI-CMM has five levels, Initial, Repeatable, Defined, Managed and Optimized.
9	Focus	ISO9000 focuses on following a set of standards so that firms delivery are successful everytime.	SEI-CMM focuses on improving the processes.