UNIT – II REQUIREMENTS ANALYSIS

Requirement Engineering Processes

1. What is Requirement Engineering? Explain Requirement Engineerin Process.

A. Requirement Engineering:
The process to gather the software requirements from client, analyze and documen them is known as requirement engineering. The goal of requirement engineering is to develop and maintain sophisticated and descriptive 'System Requiremen Specification' document.
Requirement Engineering Process It is a four step process, which includes –

- Feasibility Study
- Requirement Gathering
- Software Requirement Specification
- Software Requirement Validation

Feasibility study
When the client approaches the organization for getting the desired product developed, it comes up with rough idea about what all functions the software must perform and which all features are expected from the software. Referencing to this information, the analysts does a detailed study about whether the desired system and its functionality are feasible to develop.
This feasibility study is focused towards goal of the organization. This study analyzes whether the software product can be practically materialized in terms of implementation, contribution of project to organization, cost constraints and as per values and objectives of the organization.

Requirement Gathering
If the feasibility report is positive towards undertaking the project, next phase starts with gathering requirements from the user. Analysts and engineers communicate with the client and end-users to know their ideas on what the software should provide and which features they want the software to include.

Software Requirement Specification
SRS is a document created by system analyst after the requirements are collected from various stakeholders. SRS defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.
SRS should come up with following features:

- User Requirements are expressed in natural language.

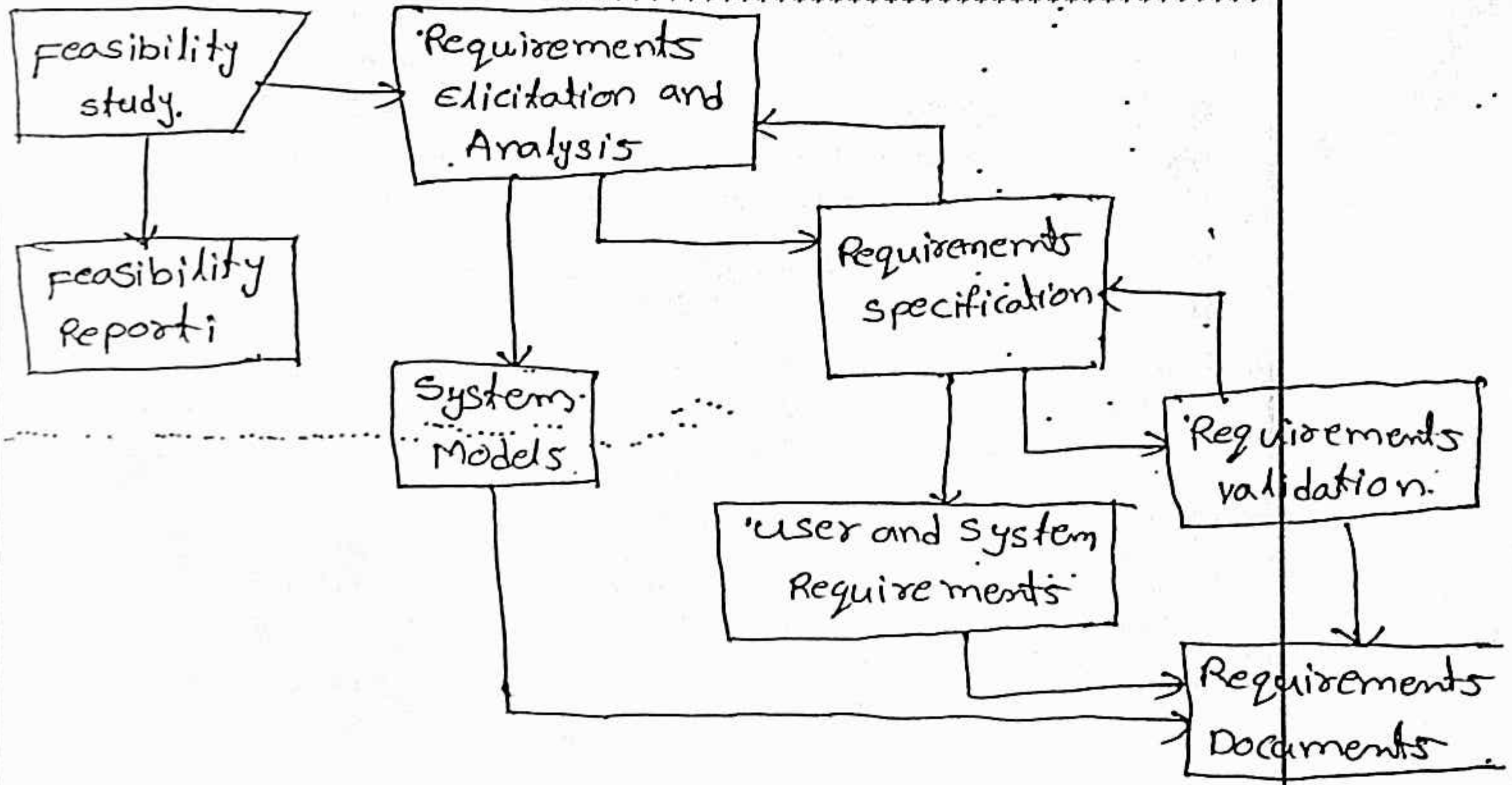
- Technical requirements are expressed in structured language, which is used inside the organization.
- Design description should be written in Pseudo code.
- Format of Forms and GUI screen prints.
- Conditional and mathematical notations for DFDs etc.

Software Requirement Validation

After requirement specifications are developed, the requirements mentioned in this document are validated. User might ask for illegal, impractical solution or experts may interpret the requirements incorrectly. This results in huge increase in cost if not nipped in the bud. Requirements can be checked against following conditions -

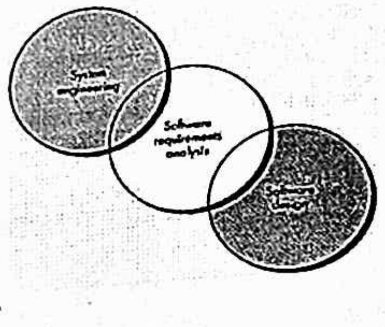
- If they can be practically implemented
- If they are valid and as per functionality and domain of software.
- If there are any ambiguities
- If they are complete
- If they can be demonstrated

****



Software Requirement Analysis

2. What is Requirement Analysis? Explain. (Or) Explain Requiremen Elicitation for software.



A. <u>**Requirement Analysis:**</u> It is a software engineering task that bridges the gap between system level *requirements engineering* and *software design.* It may be divided into five areas of effort: (1) problem recognition (2) evaluation and syntheses (3) modeling (4) specification and (5) review.

Requirements elicitation is the process of gathering the requirements. In this process the technical professional in the organization, like software developers and the system engineers, work together with the users of the system and the customers. This process is useful in finding the problems that has to be solved. The problems include:

1. What the proposed system should provide?
2. What are the expected services form the system?
3. What are the required characteristics of the system?
4. What are the required hardware and software constraints of the system?

Requirements elicitation process includes a chain of processes that interact with each other to produce requirements documentation. The lifecycle of the requirements elicitation process is:

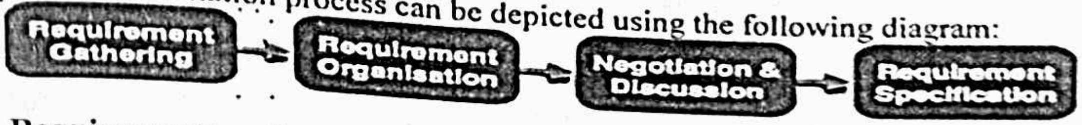
1. *Back ground Knowledge:* The analyst must understand the back ground and the domain knowledge of the application that is being developed. Example: If the system is developed for AT M, the developer should have some basic knowledge of how the ATM works.
2. *Gathering the requirements:* This is the activity of discovering the requirements by involving with the stakeholders and users.
3. *Requirements classification:* This activity includes the organizing of the requirements gathered from different sources.
4. *Requirements conflict:* This activity involves with the stakeholders and requirements engineers. This is used to solve the problems in the requirements that contradict the organization and business rules.
5. *Requirements Prioritization:* Discovering the important requirements by interacting with the stake holders and organize them in to most priority order.
6. *Requirements check:* This activity involves checking the stake holder's expectations on the system with the gathered requirements.

Requirements elicitation process not only helps the organization to gather the requirements, but also it analyses the requirements and the business procedures of the organization. The requirements elicitation and analysis is a difficult activity in the requirements engineering,

****

3. Write about Requirement Elicitation Process?

A. Requirement elicitation process can be depicted using the following diagram:



- **Requirements gathering** - The developers discuss with the client and end users and know their expectations from the software.
- **Organizing Requirements** - The developers prioritize and arrange the requirements in order of importance, urgency and convenience.
- **Negotiation & discussion** - If requirements are ambiguous or there are some conflicts in requirements of various stakeholders, if they are, it is then negotiated and discussed with stakeholders. Requirements may then be prioritized and reasonably compromised. The requirements come from various stakeholders. To remove the ambiguity and conflicts, they are discussed for clarity and correctness. Unrealistic requirements are compromised reasonably.
- **Documentation** - All formal & informal, functional and non-functional requirements are documented and made available for next phase processing.

****

4. What Requirement Elicitation techniques?

A. Requirements Elicitation is the process to find out the requirements for an intended software system by communicating with client, end users, system users and others who have a stake in the software system development. There are various ways to discover requirements:

Interviews:

Interviews are strong medium to collect requirements. Organization may conduct several types of interviews such as:

- Structured (closed) interviews, where every single information to gather is decided in advance, they follow pattern and matter of discussion firmly.
- Non-structured (open) interviews, where information to gather is not decided in advance, more flexible and less biased.
- Oral interviews
- Written interviews
- One-to-one interviews which are held between two persons across the table.
- Group interviews which are held between groups of participants. They help to uncover any missing requirement as numerous people are involved.

Surveys

Organization may conduct surveys among various stakeholders by querying about their expectation and requirements from the upcoming system.

Questionnaires

A document with pre-defined set of objective questions and respective options is handed over to all stakeholders to answer, which are collected and compiled.
A shortcoming of this technique is, if an option for some issue is not mentioned in the questionnaire, the issue might be left unattended.

Task analysis

Team of engineers and developers may analyze the operation for which the new system is required. If the client already has some software to perform certain operation, it is studied and requirements of proposed system are collected.

Domain Analysis

Every software falls into some domain category. The expert people in the domain can be a great help to analyze general and specific requirements.

Brainstorming

An informal debate is held among various stakeholders and all their inputs are recorded for further requirements analysis.

Prototyping

Prototyping is building user interface without adding detail functionality for user to interpret the features of intended software product. It helps giving better idea of requirements. If there is no software installed at client's end for developer's reference and the client is not aware of its own requirements, the developer creates a prototype based on initially mentioned requirements. The prototype is shown to the client and the feedback is noted. The client feedback serves as an input for requirement gathering.

Observation

Team of experts visits the client's organization or workplace. They observe the actual working of the existing installed systems. They observe the workflow at client's end and how execution problems are dealt. The team itself draws some conclusions which aid to form requirements expected from the software.

****

Analysis Concepts and Principles

5. What are analysis principles?

A. Each analysis method has a unique point of view. All analysis methods are related by a set of operational principles:

- ✓ Represent and understand the information domain
- ✓ Define the functions that the software
- ✓ Represent the behavior of the software
- ✓ Use models to depict information, function, and behavior
  - → uncover the details in a layered fashion.
- ✓ Move from essential information toward to details

A set of guidelines for requirement engineering:

- Understand the problem before beginning to create the analysis model
- Develop prototypes to help user to understand how human-machine interactions
- Record the origin of and the reasons for every requirement
- Use multiple views of requirements
- Prioritize requirements
- Work to eliminate ambiguity

The Information Domain: Software is built to process data, to transform data from one form to another. Software also process events. The first operational analysis principle requires to exam the information domain. Information domain contains three different views of the data and control:

1. *Information content and relationship:* Information content --> represent the individual data and control objects. There are different relationships between data and objects.
2. *Information flow*: Represents the manner in which data and control change as each move through a system. Data and control moves between two transformations (functions).
3. *Information structure:* Represent the internal organization of various data and control items.
   - ➡ Data tree structure
   - ➡ Data table (n-dimension)

Modeling: During software requirements analysis, we create models of the system to be built.

→ The models focus on - what the system must do, not how it does it.

→ The models usually have a graphic notation to represent - information, processing, system behavior, and other features

→ **The second and third operational analysis principles require:**

- build models of function and behavior
- Functional models

→ **Software transforms information. Three generic functions:** - input, processing, output

→ **Behavior models:** Most software responds to events from the outside world. A behavior model creates a representation of the states of the software and events that cause software to change state

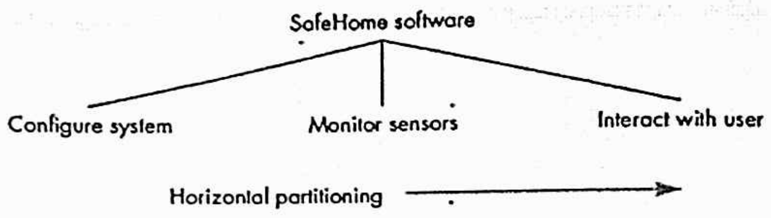
→ **Important roles of models:** The model aids the analyst in understanding the information, function, and behavior of a system.

The model becomes the focal point for review in the aspects of completeness, consistency, and accuracy of the specification.

The model becomes the foundation for design, providing the designer with an essential representation of software.

<u>**Partitioning:**</u> Partitioning decomposes a problem into its constituent parts. Establish a hierarchical representation of information (or function):

- Exposing increasing detail by moving vertically in the hierarchy
- Decomposing the problem by moving horizontally in the hierarchy.

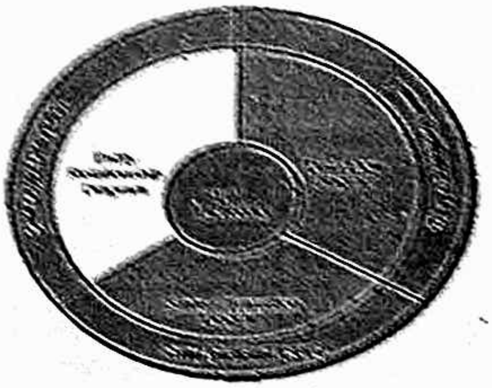


****

Analysis Model

6. What are the elements of the Analysis model?

A. The analysis model must achieve three primary objectives: 1) to describe what the customer requires. 2) to establish a basis for the creation of a software design. 3) to define a set of requirements that can be validated once the software is built. The below is the structure of the analysis model:



Data Dictionary – a repository that contain descriptions of all data objects produced by the software.

Entity Relation Diagram (ERD) – defines relationship between data objects.

Data Flow Diagram (DFD) – serves two purposes: 1) defines the transformation of the data as they move through the system. 2) to depict the functions that transform the data flow.

Process Specification (PSPEC) – a description of each function presented in the DFD is contained in the PSPEC.

State Transition Diagram (STD) – indicates how the system behaves as a consequence of external events.

Control Specification (CSPEC) – additional information about the control aspects of the software is contained in the CSPEC.

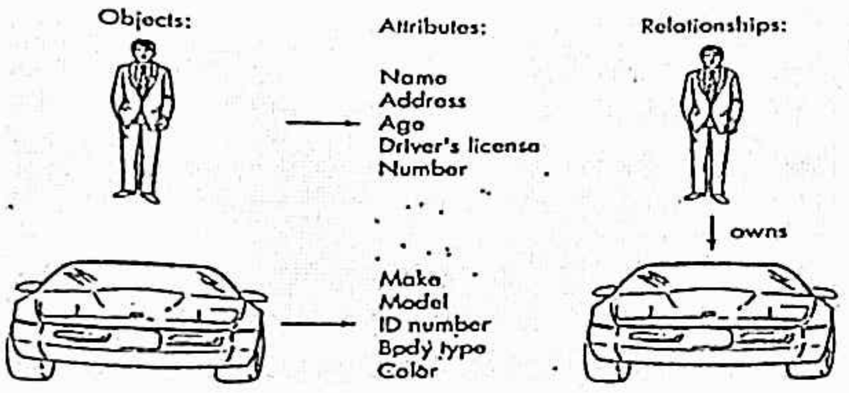
****

7. What is Data Modeling? Explain.

A. Data modeling answers a set of specific questions that are relevant to any data processing application. Data modeling methods make use of ER Diagrams.

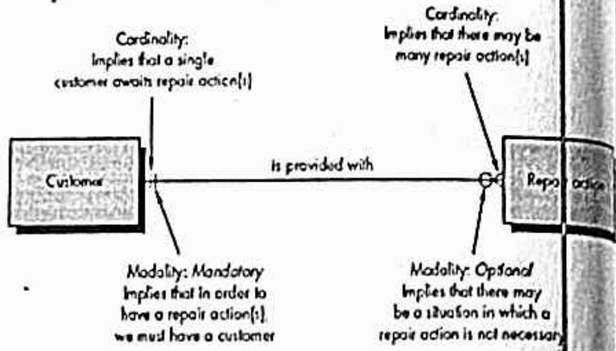
<u>**Data objects, attributes and relationships:**</u> The data model consists three interrelated pieces of information:

Data Objects: It is a representation of any composite information that must be understood by software. A data object can be an external entity, a thing, or anything. Data objects are related to one another.

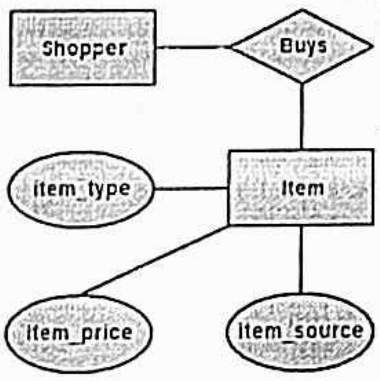


Attributes: Attribute define the properties of a data object. They can be used to:
(1) Name an instance of the data object..
(2) Describe the instance.
(3) Make reference to another instance in another table.
Relationships: It describes the connected components.

<u>Cardinality and Modality:</u> The **cardinality** defines the maximum number of objects that can participate in a relationship. The **modality** of a relationship 0 if there is no explicit need for the relationship to occur.


Cardinality:
Implies that a single customer awaits repair action(s)
Cardinality:
Implies that there may be many repair action(s)
Customer
is provided with
Repa action
Modality: Mandatory
Implies that in order to have a repair action(s), we must have a customer
Modality: Optional
Implies that there may be a situation in which a repair action is not necessary

<u>Entity – Relationship Diagrams:</u> The object / relationship pair is the cornerstone of the data model. These are represented graphically using the *entity / relationship* diagrams.



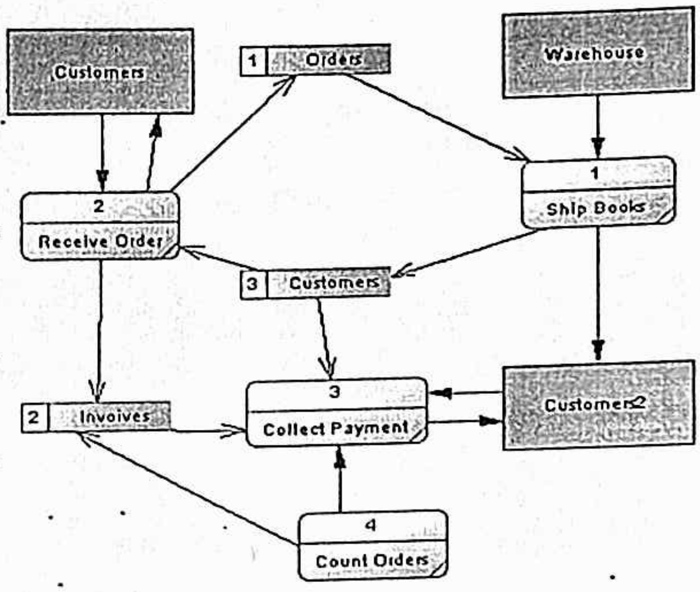
****

8. Explain functional Modeling and information flow.

A. Functional Modeling gives the process perspective of the object-oriented analysis model and an overview of what the system is supposed to do. It defines the function of the internal processes in the system with the aid of Data Flow Diagrams (DFDs). It depicts the functional derivation of the data values without indicating how they are derived when they are computed, or why they need to be computed.

Data Flow Diagrams

Functional Modeling is represented through a hierarchy of DFDs. The DFD is a graphical representation of a system that shows the inputs to the system, the processing upon the inputs, the outputs of the system as well as the internal data stores. DFDs illustrate the series of transformations or computations performed on the objects or the system, and the external controls and objects that affect the transformation.



The four main parts of a DFD (Features of DFD) are:

1) Processes

Processes are the computational activities that transform data values. A whole system can be visualized as a high-level process. A process may be further divided into smaller components. The lowest-level process may be a simple function.

2) Data Flows

Data flow represents the flow of data between two processes. It could be between an actor and a process, or between a data store and a process. A data flow denotes the value of a data item at some point of the computation. This value is not changed by the data flow.

3) Actors

Actors are the active objects that interact with the system by either producing data and inputting them to the system, or consuming data produced by the system. In other words, actors serve as the sources and the sinks of data.

4) Data Stores

Data stores are the passive objects that act as a repository of data. Unlike actors, they cannot perform any operations. They are used to store data and retrieve the stored data. They represent a data structure, a disk file, or a table in a database.

The other parts of a DFD are:

1) Constraints

Constraints specify the conditions or restrictions that need to be satisfied over time. They allow adding new rules or modifying existing ones. Constraints can appear in all the three models of object-oriented analysis.

2) Control Flows

A process may be associated with a certain Boolean value and is evaluated only if the value is true, though it is not a direct input to the process. These Boolean values are called the control flows.

****