UNIT II

REQUIREMENT ELICITATION, ANALYSIS AND SPECIFICATION

1. Requirement elicitation is the practice of collecting the requirements of a system from users, customers and other stakeholders. The practice is also sometimes referred to as "requirement gathering."

2. Types of Requirements -
   (i) **Functional Requirements** -
       - describe all the required functionality or system services -
       - dependant upon the type of software, expected users & type of system.
       - Requirement imprecision
       - Requirement completeness and consistency
   (ii) **Non-Functional Requirements** -
       - describe system properties and constraints -
       - system properties such as reliability, response time, storage requirements -
       - Constraints such as input/output data, capability, system separations etc.

   Types of non-functional requirements:
   - Usability Requirements
   - Reliability Requirements
   - Portability Requirements
   - Efficiency Requirements
   - Performance Requirements
   - Size Requirements

   Requirement arise through:
   (1) User needs
   (2) Because of Inadequate Constraints
   (3) Organization policies
   (4) Need for interoperability with other software or hardware systems
   (5) Because of external factors such as safety regulations.
(iii) User Requirements:

It should describe functional and non-functional requirements in such a way that they are understandable by system users who do not have detailed technical knowledge (e.g., natural language, tables, diagrams)

2 Requirement Sources and Elicitation Techniques

- **Requisite Interview:** Most commonly used technique is to conduct interviews

  - **Participants:** Stakeholder - like end users, system maintenance engineers, etc.

    - Problems:
      - Unrealistic expectations
      - Differences in the requirements (between stakeholders)
      - Economic and human environment
      - Political changes

- **Interviewing:** Communicates the stakeholders by asking them various questions about the system. Two types:

  - **Closed interview:** Answers predefined set of questions
  - **Open interview:** No predefined agenda.

- **Characteristics of effective interviews:**
  - Conduct in free environment with open-minded approach
  - Interviewer starts by asking questions & requirements should be gathered by

- **Use Case Modelling:** Used to action user actions with their actions. It identifies individual interactions with the system.

- **Extensively used for requirements elicitation**

- **Facility Application Specification Technique (FAST):**

  It is a method in which joint team of customer and developers work together to identify the problem, propose elements of solution, negotiate different approaches, and prepare a specification for preliminary set of solution requirements.

  Guideline for FAST approach: make list of objects, services and constraints
(5) **Quality Function Development (QFD)** - It is a quality management technique which translates the customer needs and wants into technical requirements. There are types of requirements in QFD are:

- **(1) Planned Requirements** → as per goals & objectives of the system.
- **(2) Enacted Requirements** → software package.
- **(3) Existing Requirements** → satisfied by software beyond customer's expectations.

(4) **Analysis and Modelling for function and object oriented design** - Analysis and modelling is a technical representation of the system.

- **Analysis model objectives:**
  - To define what the customer requires.
  - To establish a basis for the creation of the software design.
  - To derive a set of valid requirements after which the software construction

- **Analysis modelling approaches:**
  - **Structured approach** - Analysis is made on data and processes in which data is transformed as separate entities.
  - **Object oriented approach** - Analysis is made on the classes and interaction among them in order to meet the customer requirements.

**Structured Approach**
- Data modelling
  - E-R diagram
- Functional modelling
  - DFD (Data Flow Diagram)
  - Data Dictionary
- Behavioural modelling
  - State chart diagram

**Object Oriented Approach**
- Unified Modelling Language (UML)
  - Use Case Diagram
  - Class Diagram
  - Activity Diagram
  - Sequence Diagram
  - Deployment Diagram
  - Component Diagram
  - State Diagram
Data Modelling - data objects are examined independently of process

1. **Data Object** - a set of attributes (data items) that act as an aspect, characteristic, quality or descriptor of the object
   - **Attribute** - defines the properties of a data object. Three types:
     - **Naming attribute** - used to name an instance of a data object.
     - **Descriptive attribute** - used to describe characteristics or features of a data object.
     - **Referential attribute** - used in making reference to another instance in another table.

2. **Relationship** - connection between the data objects
   - **Cardinality** - specifies how the number of occurrences of one object is related to the number of occurrences of another object.
     - One-to-one (1:1), one-to-many (1:N), many-to-many (M:N).
   - **Modality** - indicates whether or not a particular data object must participate in the relationship.
Entity-Relationship Diagram (ERD)

- **Entity** →
- **Weak entity** →
- **Relationship** →
- **Key attribute** →
- **Multivalued attribute** →
- **Derived attribute** →

**Functional modelling**

- Used to represent the flow of information in any computer-based system. Three generic functionalities are input, process, and output.

**Data Flow diagram (DFD)**

- **Process** →
- **Data Flow** →
- **External Entity** →
- **Flow of data** →

- Represents the system at any level of abstraction.
- DFD can be partitioned into levels that represent increase in information flow and detailed functionality.

- Level 0 DFD = 'fundamental system model' or 'context model'
- Level 1 DFD = sub functions of overall system.

**Control Flow Diagram**

- Three control flows
- **Input flow** = window
- **Output flow** = (CSPEC)

- Time representation of specification are Control specification and flow specification (PSPEC)

**Rules of defining DFD**

- **1. No function can have only output or only input**
- **2. Verb phrase should be properly used**
- **3. No chain flow between data flows and external entity**
- **4. Data flow should go through a process**
Data Dictionary - It can be defined as an organized collection of all the data elements of the system with precise and rigorous definitions so that user and system analyst will have a common understanding of inputs, outputs, components of states and intermediate calculation.

Notation - Composition (\( \otimes \)), Sequence (\( + \)), Selection (\( \oplus \)), Repetition (\( ? \)), Optional data (\( \odot \)), ... comments.

Automated (CASE) tools can be used to maintain the data dictionary.

Advantages -
1. Support name management and avoid duplication.
2. Store of organizational knowledge linking analysis, design and implementation.

Behavioral Models -

Describe overall behaviour of a system represented by the state transition diagram.

State transition diagram - Collection of states and events. The event causes the system to change its state.

Also represent what actions are to be taken on occurrence of particular event.

System and Software Requirement Specification -

Software Requirement Specification (SRS)

Software requirements provide a basis for creating the Software Requirement Specification (SRS).

SRS is useful in estimation cost, planning, team activities, perform tasks and tracking the team's progress throughout the development activity.
1. Introduction

1.1 Purpose of this document

1.2 Scope of this document

1.3 Overview

2. General description

3. Functional Requirements (Description, Criticality, Technical Issues, Cost and Schedule, Risks, dependencies with other requirements)

4. Interface Requirements

4.1 User Interfaces

4.1.1 GUI (Graphical User Interface)

4.1.2 CLI (Command line Interfaces)

4.1.3 API (Application Programming Interface)

4.2 Hardware Interface

4.3 Communication Interfaces

4.4 Software Interfaces

5. Performance Requirements

6. Design Constraints

7. Other Non-Functional Attributes

7.1 Security

7.2 Binary Compatibility

7.3 Reliability

7.4 Maintainability

7.5 Portability

7.6 Extensibility

7.7 Reusability

7.8 Application Compatibility

7.9 Resource Utilization

7.10 Serviceability
8. Operational Scenarios
9. Preliminary Schedule
10. Preliminary Budget
11. Appendices
   11.1 Definitions, Acronyms, Abbreviations
   11.2 References

Characteristics of good SRS:

1. Consistent
2. Complete
3. Unambiguous (unique interpretation)
4. Consistent (no conflicts)

Three types of conflicts:

- Logical and Temporal conflicts
- Characteristic conflicts of real world object

Two different discussions about the same real world object

5. Stability
6. Verifiable

7. Traceable: Two types -
   - Forward Traceability: Each requirement is referred in the SRS document by its unique name or reference number.
   - Backward Traceability: If the reference to the requirement is mentioned in earlier document, then it is backward traceability

6. Requirement Validation -

It is a technique in which the description in SRS document is verified for system implementation. Checks made on SRS document are:

1. Completeness and consistency
2. Conformance to standards
3. Requirement conflicts
4. Ambiguity in requirements
Requirement validation techniques:

(1) Requirement Review:
   (1) Read and analyse requirements
   (2) Identify the problems
   (3) Discuss the problems
   (4) Agree upon the actions to solve these problems

Software Prototyping:

Use of system prototype is to help customers and developers to understand the system requirements.

Software prototyping activities:

- Envision software prototype
- Enforce errors and incomplete requirements

Requirement Elicitation → Requirement Validation

Traceability:

It is concerned with relationship between requirements, their sources and the system design. Using traceability the requirement finding becomes easy. Various types of traceability are:

(1) Source traceability - stakeholders
(2) Requirements traceability - dependent requirements
(3) Design traceability - design

Traceability information is typically represented by a data structure called Traceability matrix.

<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8</td>
<td>D</td>
<td>R</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>4</td>
<td>D</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>