# **Requirements Modelling**

Requirements Modeling is a structured approach used in software development and system engineering to analyze, document, and represent the functional and non-functional requirements of a system. It involves creating visual and textual representations of how a system should behave, what functionalities it must support, and how users will interact with it.

Requirements Modeling helps bridge the gap between business needs and technical implementation by providing clear, structured documentation that stakeholders—including business analysts, developers, designers, and testers—can use to ensure system alignment with business objectives.

## **Aspects of Requirements Modelling**

Requirements Modeling typically includes different types of models to represent requirements in a structured manner:

1. **Use Case Models**

Use Case Models are a central part of requirements modeling, particularly in system design. They focus on how users (actors) interact with a system to achieve specific goals. The primary purpose of a Use Case Model is to capture the functional requirements of the system by identifying the different ways users will use the system and the actions they will take.

Each use case in the diagram represents a specific system behavior or interaction that provides value to the user or external systems. The model emphasizes user-system interactions and is particularly useful for understanding user needs.

For visual representation, Use Case Diagrams are typically used. These diagrams depict actors (both primary and secondary) and the use cases they interact with. An actor can be a user or an external system, and the use case represents the functional requirements that the actor needs to achieve.

Some common diagram types under this category are:

* Use Case Diagrams
* Interaction Diagrams (Example - Sequence Diagrams)
* Activity Diagrams (which can show the flow of activities between use cases).

1. **Data Models**

Data Models are used to represent how data is structured, stored, and processed within a system. The primary goal of data modeling is to provide an organized framework for how data entities relate to one another and how they are managed. Data models describe the types of data the system will handle, the relationships between these data types, and the rules that govern how data should be handled throughout its lifecycle.

In a data model, we define entities (such as users, orders, or products), their attributes (like user name, product price), and the relationships between these entities (such as one-to-many or many-to-many relationships).

Common diagram types in this category include:

* Entity-Relationship (ER) Diagrams: These diagrams focus on entities and their relationships and are useful for designing databases.
* Class Diagrams (in UML): Represent how different classes (or objects) within the system are related, with attributes and methods defined for each class.
* Data Flow Diagrams (DFD): Used to show the flow of data between processes and external systems, depicting inputs, outputs, and data stores.

1. **Process Models**

Process Models represent workflows and business processes. They depict how the system or business operations occur step by step and how different components within the system interact to achieve the desired outcomes. The primary purpose of process modeling is to describe the business logic, process flow, and interactions within and between different systems or processes.

In this type of modeling, each process is represented by a specific action or event, and arrows or transitions illustrate how one process leads to the next. Process modeling helps identify the sequence of operations, decision points, and any dependencies that exist in a given workflow.

Common diagram types in this category include:

* Business Process Model and Notation (BPMN): A standardized method for depicting the flow of business processes, with symbols representing different types of actions (tasks, gateways, events, etc.).
* Flowcharts: Simplified representations of business processes or operations, with a focus on decision-making paths.
* Activity Diagrams: A type of UML diagram that models workflows, similar to flowcharts, but more formal and structured.

1. **State Models**

State Models are used to describe the different states an object or system can be in and the transitions between those states. State modeling focuses on how a system or object changes over time in response to internal or external events, actions, or conditions.

The primary purpose of state modeling is to track the life cycle of an object or a system and to visualize how the system responds to various triggers. A state model will usually represent a system as a set of states (e.g., active, inactive, completed) and the transitions between those states, based on specific events or conditions.

Common diagram types in this category include:

* State Diagrams (in UML): Depict the different states of an object and the events that cause transitions between states.
* State Machines: Formal models that describe the possible states and transitions in a system.
* State Tables: A tabular format that shows the states of a system and the actions that trigger transitions.

## **Importance of Requirements Modelling**

* **Ensure Clarity an Avoid Miscommunication**

Stakeholders often have different interpretations of what a system should do. Requirements Modeling provides a visual and structured format, ensuring that all stakeholders have a common understanding of the system’s goals.

* **Helps in Requirement Validation and Verification**

Before development begins, models help verify that all critical requirements are captured and validated against business objectives. This reduces the risk of misunderstandings, incomplete requirements, or incorrect functionality.

* **Improves System Design and Development Efficiency**

A well-structured requirements model serves as a blueprint for developers, helping them understand system functionalities clearly. It reduces ambiguity, rework, and misinterpretation in the development process, leading to better efficiency.

* **Supports Better Decision-Making**

By analyzing models, businesses can evaluate different design options, identify risks, and make informed decisions before system implementation.

* **Enhances Traceability and Change Management**

Requirements models allow teams to track changes and understand how modifications in one part of the system affect others. This is crucial in large projects where evolving requirements can impact development.

* **Reduces Development Costs and Time**

Clear and well-modelled requirements help avoid costly changes during later stages of development. Addressing misunderstandings or missing requirements early significantly reduces time delays and budget overruns.