SNS COLLEGE OF ENGINEERING

Kurumbapalayam (po), Coimbatore – 641 107

Accredited by NAAC-UGC with 'A' Grade

Approved by AICTE & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

23ITT203- OBJECT ORIENTED SOFTWARE ENGINEERING

UNIT 2

Requirements Gathering and Analysis in Object-Oriented Software Engineering (OOSE)

In **Object-Oriented Software Engineering**, **Requirements Gathering and Analysis** is the first and crucial phase in the software development lifecycle. This phase focuses on collecting, understanding, and documenting the needs and constraints of the system, ensuring that the software will meet user expectations and business goals.

1. What is Requirements Gathering and Analysis?

Requirements Gathering refers to the process of collecting the needs, expectations, and constraints from stakeholders who will interact with the software. **Requirements Analysis** is the process of understanding and refining these gathered requirements, categorizing them, and structuring them in a way that ensures a clear understanding for the development team.

2. Objectives of Requirements Gathering and Analysis

The main goal of this phase is to understand the system's functionality, the user needs, and the constraints before moving to design and implementation. This phase typically includes:

- Understanding **functional requirements**: What the system should do.
- Understanding **non-functional requirements**: How well the system should perform (e.g., speed, security, scalability).
- Identifying and prioritizing requirements: What are the most important features? What features are must-haves vs. nice-to-haves?
- Clarifying any ambiguities in requirements to ensure that the development team and stakeholders have a shared understanding.

3. Types of Requirements

1. **Functional Requirements**:
   These specify what the system should do, i.e., the tasks it must perform. These often include:
   - Specific actions or operations the system must support.

- Behavior or responses to certain conditions or inputs.

Example:
In a **Library Management System**, a functional requirement might be:

- *"The system should allow a user to borrow a book after verifying the user's account and the book's availability."*

2. **Non-Functional Requirements**:
   These specify **how** the system should perform, addressing system qualities or constraints like:
   - **Performance**: Speed, response times, throughput.
   - **Scalability**: How well the system can handle increased loads.
   - **Security**: Data protection, access control.
   - **Usability**: User experience, ease of use.
   - **Reliability**: Uptime, error handling.

   Example:
   In the **Library Management System**, a non-functional requirement might be:

   - *"The system must support up to 500 concurrent users without performance degradation."*

3. **Interface Requirements**:
   These describe the interactions between the software system and external systems or devices (e.g., hardware, third-party services, or APIs).

   Example:

   - *"The Library Management System must integrate with a payment gateway for handling overdue fines."*

4. Process of Requirements Gathering

Requirements Gathering involves several activities to ensure all relevant information is captured. Below are the key steps involved in this process:

1. **Identify Stakeholders**:
   Stakeholders are individuals or groups who have an interest in the system. These might include:
   - End users (people who will use the system regularly)
   - System administrators
   - Project managers
   - Business analysts
   - Developers
   - Customers or clients
   - Regulatory bodies (if applicable)

Example:
In a **Library Management System**, stakeholders may include the library staff, library users, system administrators, and external vendors (if there's a payment system integrated).

2. **Conduct Stakeholder Interviews**:
   One of the most common ways to gather requirements is through one-on-one interviews with stakeholders to learn about their needs, problems, and desired features.

   Example:
   Interviewing a librarian about the most common tasks they perform can provide insights into essential features such as:

   - Searching for books.
   - Issuing/returning books.
   - Generating reports on overdue books.
3. **Surveys and Questionnaires**:
   If there are many stakeholders or users, surveys and questionnaires can help collect feedback from a large group efficiently.

   Example:
   In a **Library System**, a survey might ask users about their experience searching for books online or about features they'd like, such as:

   - Online book reservations.
   - Email notifications for overdue books.
4. **Observe Users and Existing Systems**:
   Observing how users interact with an existing system (if any) or how they perform tasks manually provides valuable insights into how the new system can improve their workflow.

   Example:
   Observing library staff manually checking books in and out could highlight areas for automation and streamline processes in the new **Library Management System**.

5. **Document Analysis**:
   If there are existing documents (e.g., policies, reports, or previous system documentation), reviewing these can help identify requirements or improve the new system's design.

   Example:
   Analyzing past reports about book usage or overdue books can reveal important features, such as:

   - Fine calculation.
   - Popular book tracking.
   - Book reservation limits.

6. **Use Cases and User Stories**:
   A **Use Case** is a detailed description of how users will interact with the system to achieve specific goals. Use cases often define step-by-step interactions.

   Example:

   - **Use Case**: Borrow Book
     Actors: User, Library System
     Scenario:
     1. The user logs into the system.
     2. The user searches for a book by title or author.
     3. The system displays book details.
     4. If the book is available, the user can borrow it.
     5. The system updates the book status as "borrowed."

5. Requirements Analysis

Once requirements are gathered, they must be analyzed to ensure they are well-defined, feasible, and aligned with the overall goals of the system. Analysis helps clarify ambiguities, resolve conflicts, and prioritize features.

1. **Modeling Requirements**: Object-oriented techniques like **Use Case Modeling** and **Class Modeling** are often used in this phase to visualize system functionality and structure.

   Example:

   - **Use Case Diagram** for the Library Management System might show actors (User, Librarian) and use cases (Search Book, Borrow Book, Return Book).
   - **Class Diagram** might define classes like Book, User, Library, with attributes and methods.
2. **Validate and Refine Requirements**: The gathered requirements need to be validated to ensure they are complete, unambiguous, and realistic. This is usually done by reviewing the requirements with stakeholders.

   Example:
   A stakeholder might point out that the system should allow books to be reserved by users even if they are currently unavailable, which was initially missed in the requirements gathering.

3. **Identify Dependencies and Constraints**: It's important to identify any interdependencies between requirements, as well as external constraints (technical, financial, or legal).

   Example:

   - **Dependency**: The **Borrow Book** feature depends on the **Search Book** feature being functional.

    - **Constraint**: The system needs to be compatible with both Windows and Linux platforms, limiting the choice of development tools.
4. **Prioritize Requirements**: Once requirements are validated, prioritize them based on importance, dependencies, and available resources. This helps the team focus on critical features first.

   Example:

    - **High Priority**: Allow users to borrow and return books.
    - **Low Priority**: Add social sharing features like sharing book details on social media.

6. Challenges in Requirements Gathering and Analysis

- **Ambiguous Requirements**: Users may not know exactly what they want or may give unclear or contradictory requirements.
- **Changing Requirements**: As the development progresses, stakeholders may change their minds or introduce new requirements.
- **Incomplete Information**: Stakeholders may fail to provide all the details needed for a complete specification.
- **Conflicting Stakeholder Needs**: Different stakeholders may have conflicting needs or priorities for the system.

Summary: Requirements Gathering and Analysis

1. **Gathering**: Collect information from stakeholders through interviews, surveys, observations, and document analysis.
2. **Analysis**: Refine the requirements, identify dependencies, resolve ambiguities, and prioritize features.
3. **Modeling**: Use **Use Cases** and **Class Diagrams** to capture and organize the system's functionality.
4. **Validation**: Ensure the requirements are correct, complete, and feasible through reviews and discussions with stakeholders.
5. **Prioritize**: Rank requirements based on importance and available resources.

Example:
In a **Library Management System**, gathering might include conducting interviews with librarians about their current processes, surveying users on their needs (e.g., online reservations), and analyzing system constraints (e.g., the system needs to handle 500 concurrent users). After analysis, you may create use case diagrams and class diagrams to structure the system and identify key features, such as **Search Book**, **Borrow Book**, **Return Book**, and **Generate Overdue Report**.

By gathering and analyzing requirements thoroughly, the development team ensures that the system meets the needs of stakeholders and operates efficiently.