Chapter 4

Requirements Engineering

Requirements Engineering

Requirements Engineering is the process of figuring out what a system needs to do and how it should operate. It involves gathering, analyzing, and documenting these requirements.

Example: Imagine you're building a house. Before you start hammering nails, you need to know what kind of house you want to build, how many rooms it should have, and what features it should include. This is essentially what requirements engineering is for software systems

What is the Requirement?

A requirement is a statement that describes what the system must do or how it must behave. It can be anything from a general statement of purpose to a specific technical specification.

Types of Requirements

- **User Requirements:** Statements in natural language. Plus diagrams of the services the system provides and its operational constraints. Written for customers.

- **System Requirements:** More detailed and technical. Define how the system will be built and implemented.

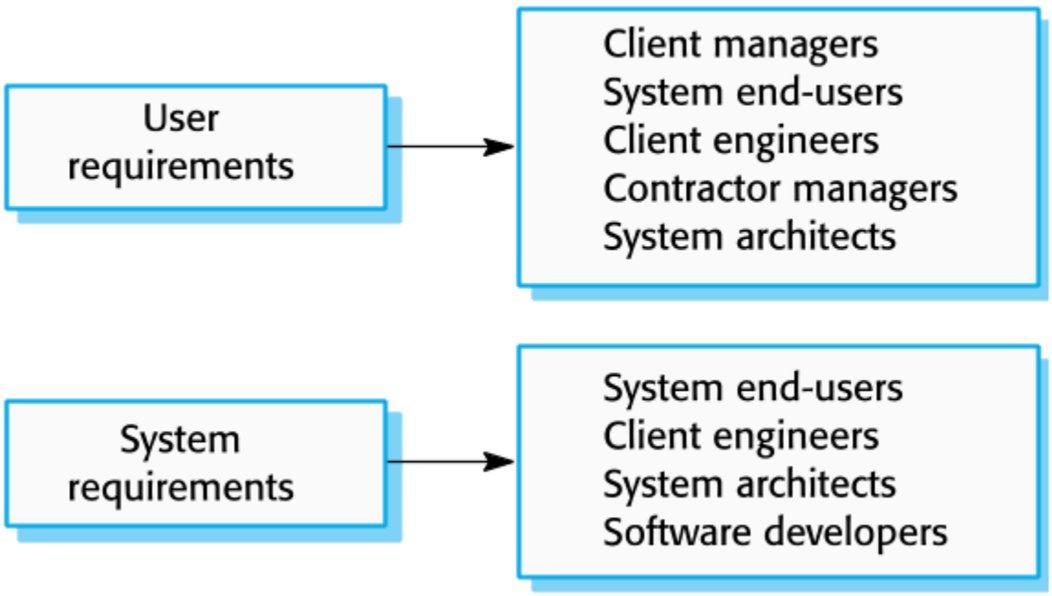
User requirements definition

1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20mg, etc) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

Readers of different types of requirements specification



Agile Methods and Requirements

Agile software development methods emphasize flexibility and adaptability. They often use incremental requirements engineering, where requirements are gathered and refined throughout the development process.

Functional Vs Non Functional Requirements

Functional Requirements

- **Purpose:** Describe the specific tasks or services that a system must perform.
- **Characteristics**
  - ★ Focus on what the system should do, not how it should do it.
  - ★ Can be high-level or detailed.
- **Examples:**
  - ★ The system shall allow users to create and edit documents.
  - ★ The system shall generate reports based on user-defined criteria.
  - ★ The system shall integrate with existing systems.

Non-Functional Requirements

- **Purpose:** Specify constraints on the system's behavior or operation.
- **Characteristics**
  - ★ Do not describe specific tasks or services.
  - ★ Specify how the system should perform its tasks or services.

- **Examples:**
  - ★ **Performance:** The system shall process at least 100 transactions per second.
  - ★ **Reliability:** The system shall be available 99.9% of the time.
  - ★ **Security:** The system shall protect user data from unauthorized access.
  - ★ **Usability**: The system shall be easy to use for users with varying levels of technical expertise.

Requirements engineering processes

Requirements engineering processes vary depending on the application domain, the people involved, and the organization developing the requirements. However, there are a number of generic activities common to all processes:

- **Requirements elicitation:** Gathering requirements from stakeholders.
- **Requirements analysis:** Understanding and refining the requirements.
- **Requirements validation:** Checking that the requirements are correct and complete.
- **Requirements management:** Managing the requirements throughout the development process.

In practice, requirements engineering is an iterative activity in which these processes are interleaved.

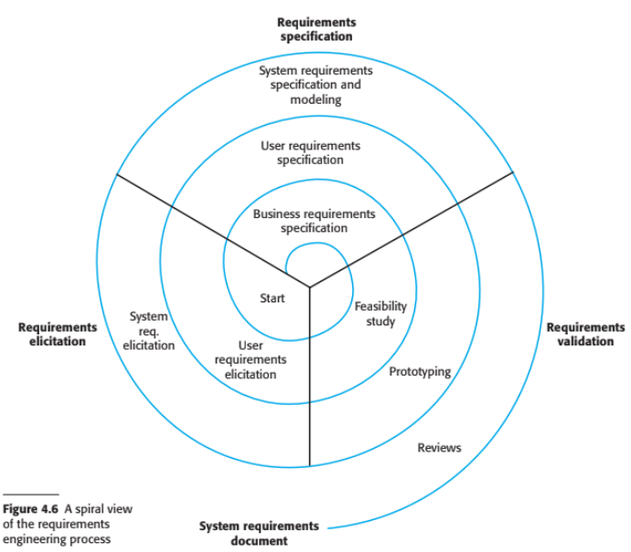


Figure 4.6 A spiral view of the requirements engineering process

Requirements elicitation is the process of gathering information about the required and existing systems and distilling the user and system requirements from this information.

Stakeholders: Requirements elicitation involves interacting with system stakeholders, which can include:

- ❖ End-users
- ❖ Managers
- ❖ Engineers involved in maintenance
- ❖ Domain experts
- ❖ Trade unions

Process Activities

The requirements elicitation and analysis process typically includes the following activities:

- **Requirements discovery:** Gathering information about the application domain, system services, and operational constraints.
- **Requirements classification and organization:** Grouping related requirements and organizing them into coherent clusters.
- **Requirements prioritization and negotiation:** Prioritizing requirements and resolving conflicts.
- **Requirements specification:** Documenting requirements and inputting them into the next round of the spiral.

Problems of Requirements Elicitation

1. Stakeholders may not know what they really want.
2. Stakeholders may express requirements in their own terms, which may be difficult for requirements engineers to understand.
3. Different stakeholders may have conflicting requirements.
4. Organizational and political factors may influence the system requirements.
5. The requirements may change during the analysis process due to new stakeholders emerging or changes in the business environment.

Techniques for Requirements Elicitation

Interviewing

1. Formal or informal interviews with stakeholders are commonly used to gather information.

2. Types of interviews include closed interviews (based on predefined questions) and open interviews (where various issues are explored).
3. Effective interviewing involves being open-minded, avoiding preconceived ideas, and prompting interviewees to discuss the system.

Ethnography

Ethnography involves a social scientist observing and analyzing how people actually work.

Benefits of ethnography include:

1. People do not have to explain or articulate their work.
2. Social and organizational factors of importance may be observed.
3. Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.
4. Ethnography is effective for understanding existing processes but cannot identify new features that should be added to a system.

Stories and Scenarios

1. Stories and scenarios are real-life examples of how a system can be used.
2. Stories are high-level descriptions of system use, while scenarios are more structured and detailed.
3. Stories and scenarios can be used to facilitate discussions with stakeholders and to develop more specific system requirements.

Requirements Specification

Requirements specification is the process of writing down the user and system requirements in a requirements document.

Types of Requirements Specifications

- **User requirements:** Understandable by end-users and customers who do not have a technical background.
- **System requirements:** More detailed requirements that may include more technical information.

Notation for Writing Requirements Specifications

- **Natural language:** Expressive, intuitive, and universal.
- **Structured natural language:** Uses a standard form or template to provide more structure.
- **Design description languages:** Abstract languages that define an operational model of the system.
- **Graphical notations:** Graphical models supplemented by text annotations, such as UML use case and sequence diagrams.
- **Mathematical specifications:** Based on mathematical concepts, but may be difficult for customers to understand.

Guidelines for Writing Requirements

- Use a standard format for all requirements.
- Use language consistently (e.g., "shall" for mandatory requirements).
- Highlight key parts of the requirement.
- Avoid computer jargon.
- Include an explanation of why a requirement is necessary.

Problems with Natural Language Requirements

- Lack of clarity and precision.
- Confusion between functional and non-functional requirements.
- Amalgamation of several requirements.

Structured Specifications

- Limit the freedom of the requirements writer.
- Use a standard way of writing requirements.
- May be too rigid for some types of requirements.

Form-Based Specifications:

- Define the function or entity, inputs, outputs, information needed, actions to be taken, pre- and post-conditions, and side effects.

Tabular Specifications:

- Supplement natural language.
- Useful for defining alternative courses of action.

Use Cases

- Scenarios that identify actors and interactions in the system.
- Describe all possible interactions with the system.

- Can be supplemented with UML sequence diagrams to show event processing.

Software Requirements Document (SRD)

- Official statement of system requirements.
- Includes both user requirements and system requirements.
- Should focus on what the system should do, not how it should do it.

Users of an SRD:

- **Developers**
- **Testers**
- **Customers**
- **Project managers**

Variability in SRDs

- Depends on the type of system and development approach.
- Incremental development typically results in less detailed SRDs.

Structure of an SRD

1. Preface
2. Introduction
3. Glossary
4. User requirements definition
5. System architecture
6. System requirements specification
7. System models
8. System evolution
9. Appendices
10. Index

Requirements Validation

Requirements validation ensures that the specified requirements accurately reflect the intended functionality and behavior of the system as desired by the stakeholders.

Importance:

- High cost of errors: Fixing requirements errors can be extremely expensive, especially after deployment.

- Complexity of systems: Modern systems often involve complex interactions and dependencies, making it challenging to verify their correctness and completeness.

Requirements Checking

- **Validity:** Assessing whether the requirements align with the customer's needs and objectives.
- **Consistency:** Identifying and resolving conflicts between different requirements.
- **Completeness:** Ensuring that all necessary requirements are included.
- Realism: Evaluating whether the requirements can be implemented within the given constraints (e.g., budget, technology).
- **Verifiability:** Determining whether the requirements can be objectively tested and evaluated.

Requirements Validation Techniques

- **Requirements reviews**

    ★ Formal reviews: Involving stakeholders to systematically analyze and discuss requirements documents.
    ★ Informal reviews: Facilitating ongoing communication and feedback between developers, customers, and users to identify and address issues early on.

- **Prototyping**

    ★ Executable models: Creating simplified, working models of the system to demonstrate its functionality and gather feedback.

- **Test-case generation**

    ★ Testability: Developing test cases to assess whether the requirements can be effectively verified through testing.

Review Checks

- **Verifiability:** Is the requirement realistically testable?
- **Comprehensibility:** Is the requirement properly understood?
- **Traceability:** Is the origin of the requirement clearly stated?
- **Adaptability:** Can the requirement be changed without affecting other requirements or the system's overall functionality? other parts of the system?