Requirements Engineering

CSC224

Week 3 Notes

Week 3: Requirements Engineering

- **Topics:**
  - Functional and non-functional requirements.
  - Requirements elicitation techniques.
  - Requirements documentation and validation.
  - Gathering and analyzing user requirements
  - Writing software requirements specifications
  - Use case modeling
- **Learning Objectives:**
  - Apply techniques to gather and analyze software requirements.
  - Document requirements effectively using industry standards.

Requirements Engineering

- The requirements for a system are the descriptions of the services that a system should provide and the constraints on its operation. These requirements reflect the needs of customers for a system that serves a certain purpose such as controlling a device, placing an order, or finding information.
- The process of finding out, analyzing, documenting and checking these services and constraints is called requirements engineering (RE).

1. What is Requirements Engineering?

- **1. What is Requirements Engineering?**
- **Definition:**
  Requirements engineering is the process of defining, documenting, and maintaining the requirements for a software system. It ensures that the system meets user and stakeholder needs.
- **Key Goals:**
- Understand what the stakeholders need from the system.
- Establish clear, unambiguous, and complete documentation of requirements.
- Provide a basis for software design, implementation, and validation.

2. Types of Requirements

- **A. Functional Requirements**
- Define the specific functions a system must perform.
- Typically describe inputs, outputs, and processing.
- **Examples:**
- A banking system must allow users to transfer funds between accounts.
- An e-commerce platform must enable users to add items to a shopping cart.

User vs System Requirements

- **User requirements** are statements, in a natural language plus diagrams, of what services the system is expected to provide to system users and the constraints under which it must operate. The user requirements may vary from broad statements of the system features required to detailed, precise descriptions of the system functionality.
- **System requirements** are more detailed descriptions of the software system's functions, services, and operational constraints. The system requirements document (sometimes called a functional specification) should define exactly what is to be implemented. It may be part of the contract between the system buyer and the software developers.

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.

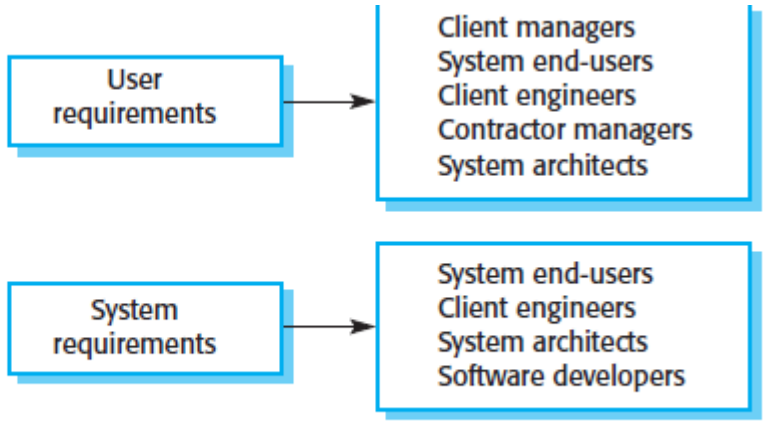

Figure 4.2 Readers of different types of requirements specification

2. Types of Requirements

- **B. Non-Functional Requirements**
- Define constraints on how the system performs its functions.
- Categories include:
  - **Performance:** Response time, processing speed.
  - **Usability:** Ease of use, accessibility.
  - **Reliability:** System uptime, fault tolerance.
  - **Security:** Access control, data encryption.
  - **Scalability:** Ability to handle increased loads.
- **Examples:**
- The system must respond to user queries within 2 seconds (performance).
- Only authorized users can access customer data (security).

2. Types of Requirements

- **C. Domain Requirements**
- Specific to the application domain or industry.
- Derived from domain knowledge, standards, or regulations.
- **Examples:**
- In healthcare systems, comply with HIPAA regulations for patient data confidentiality.

3. The Requirements Engineering Process

- **A. Requirements Elicitation**
- The process of gathering information about what the stakeholders need.
- **Techniques:**
- **Interviews:**
  - One-on-one sessions with stakeholders to gather detailed insights.
  - Types: Structured (predefined questions) and unstructured (free-form).
- **Workshops and Focus Groups:**
  - Collaborative sessions to gather requirements from multiple stakeholders.
  - Useful for resolving conflicting viewpoints.
- **Observation:**
  - Watching users interact with existing systems to identify pain points.

3. The Requirements Engineering Process

- **Prototyping:**
  - Developing early models of the system to clarify requirements.
- **Questionnaires and Surveys:**
  - Useful for collecting input from a large number of stakeholders.
- **Document Analysis:**
  - Reviewing existing documentation, such as policy manuals and user guides.

B. Requirements Analysis

- Prioritize and refine the gathered requirements.
- Resolve conflicts and eliminate ambiguities.
- **Key Techniques:**
- **Modeling Requirements:**
  Use tools like UML to create diagrams (use case diagrams, activity diagrams).
- **Requirements Categorization:**
  Classify requirements into functional, non-functional, and domain requirements.
- **Conflict Resolution:**
  Engage stakeholders to address contradictions in their needs.

C. Requirements Specification

- **C. Requirements Specification**
- Documenting the requirements clearly and systematically.
- Typically results in a **Software Requirements Specification (SRS)** document.

- **Structure of an SRS Document:**

1. Introduction (Purpose, Scope, Definitions).
2. Overall Description (Product Perspective, User Characteristics).
3. Functional Requirements.
4. Non-Functional Requirements.
5. Constraints and Assumptions.

D. Requirements Validation

- **D. Requirements Validation**
- Ensures the documented requirements are correct, complete, and feasible.
- **Validation Techniques:**
- **Reviews and Inspections:**
  Conduct formal reviews with stakeholders and technical teams.
- **Prototyping:**
  Build prototypes to confirm understanding with users.
- **Test Cases:**
  Define acceptance criteria and test cases based on requirements.
- **Checklists:**
  Use standard checklists to verify completeness and consistency.

4. Challenges in Requirements Engineering

- **Ambiguity:**
  Poorly defined requirements can lead to misinterpretations.
- **Changing Requirements:**
  Evolving stakeholder needs or market conditions may necessitate updates.
- **Stakeholder Conflicts:**
  Different stakeholders may have competing or conflicting priorities.
- **Incomplete Knowledge:**
  Stakeholders may not always articulate their needs effectively.
- **Communication Barriers:**
  Miscommunication between technical teams and non-technical stakeholders.

5. Key Characteristics of Good Requirements

- To ensure the success of a software project, requirements should be:
- **Clear and Unambiguous:** Avoid vague terms (e.g., "fast response").
- **Complete:** Cover all aspects of the system.
- **Consistent:** Avoid conflicts between requirements.
- **Feasible:** Ensure the requirements are technically achievable.
- **Testable:** Define criteria for validating each requirement.

6. Tools for Requirements Engineering

- **Popular Tools:**
- **Jira:** Managing requirements, tasks, and workflows.
- **Lucidchart/Draw.io:** Creating visual models like UML diagrams.
- **Microsoft Word/Google Docs:** Documenting the SRS.
- **Trello/Asana:** Tracking progress and stakeholder discussions.

7. Example: ATM System Requirements

- **Functional Requirements:**
- Users can withdraw cash from their accounts.
- Users can check their account balance.
- The system should generate a receipt for each transaction.
- **Non-Functional Requirements:**
- The system should process transactions within 2 seconds.
- All data must be encrypted during transmission.
- **Domain Requirements:**
- The system must comply with local banking regulations.
- Support multiple currencies for international transactions.

8. Key Takeaways

- Requirements engineering bridges the gap between stakeholders' needs and software design.
- Functional and non-functional requirements provide a comprehensive view of what the system should do and how it should behave.
- Effective elicitation, specification, and validation are critical to successful software projects.

Discussion Questions

- Why is it important to distinguish between functional and non-functional requirements?
- How can prototyping help resolve ambiguous requirements?
- Discuss strategies to manage changing requirements during a project.

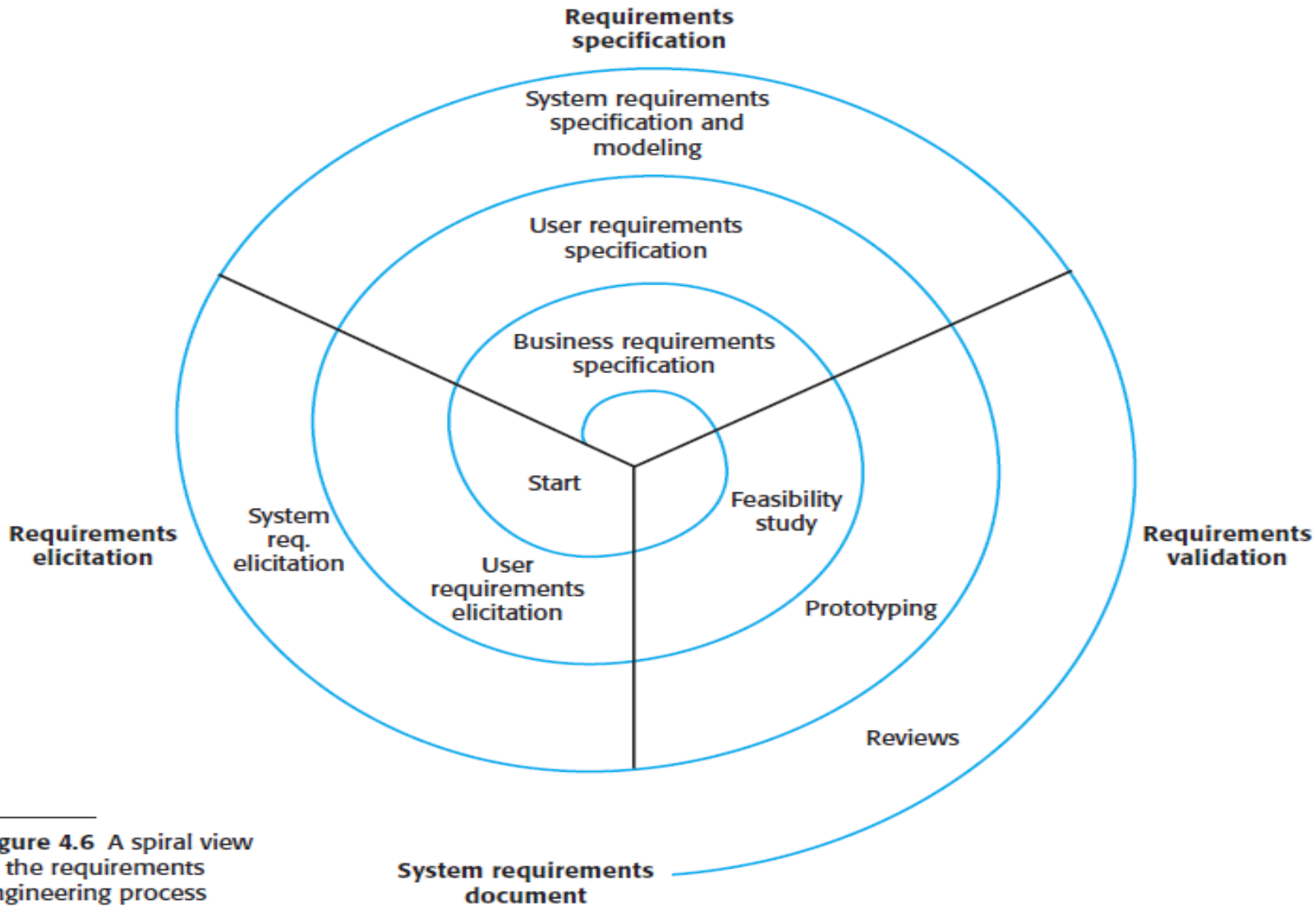

Figure 4.6 A spiral view of the requirements engineering process

4.3 Requirements elicitation

The aims of the requirements elicitation process are to understand the work that stakeholders do and how they might use a new system to help support that work. During requirements elicitation, software engineers work with stakeholders to find

Reference

- Sommerville, I. (2015). *Software engineering* (10th ed.). Pearson.