Chapter One

Software Requirement Engineering

Compiled by: | *Sinodos G*

Dire Dawa - 2017 E.C

Requirements

- A statement of a system service or constraint
- The foundation of software development.
- A clear definition of what the system should do.
- Includes functional and non-functional aspects.
- They may be:
  - *user-level facility description,*
  - *detailed specification of expected system behavior,*
  - *general system property,*
  - *specific system constraints*
  - *Information on how to carry out some computation,*

- *Example*
  - *Imagine you're building an online Clearance System for DDU.*
    - □ *What features should it have?*
    - □ *Who will use it?*

  - A requirement can be **bad vs. good.**
    - □ *The system should be fast.* (bad)
    - □ *The system should load the dashboard within 2 seconds for 97% of users.*

Requirements Engineering

- The process of gathering, analyzing, documenting, and managing software requirements
- Covers all of the activities involved in discovering, documenting, and maintaining a set of requirements for a computer-based system.
- It is a common approach in system development.
- *The processes involved in developing system requirements*

- ***Example:***
  - *Think of RE like designing a house*
- How much does requirements engineering cost?
  - About 15% of system development costs

- ***Objectives***
  - Introduce the notion of system requirements and the requirements engineering process.
  - explain how requirements engineering fits into a broader system engineering process
  - explain the importance of the requirements document

Importance of RE

- **Prevents misunderstandings**
  - developers and clients.
  - Users & organizations
  - Serves as a **blueprint** for developers, testers, and stakeholders.
- **Reduces development costs**
  - catching errors early.
- **Ensures software quality**
  - defining clear objectives.
- **Improves user satisfaction**
  - meeting their needs.

- ***Example:***
  - Imagine you develop an e-commerce app, but the client wanted a payment system that supports multiple currencies, and you didn't include it.

Requirements are wrong?

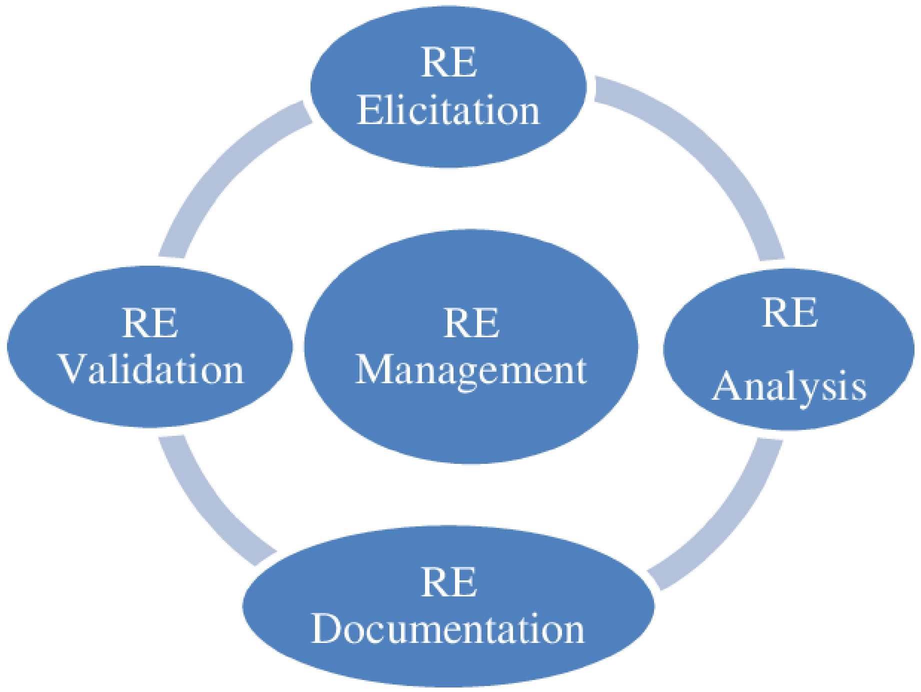
- *Delays & Higher Costs*
  - The system may be delivered late and exceed the budget.
- *User Dissatisfaction*
  - Customers may reject the system if it doesn’t meet their needs.
- *Unreliable Performance*
  - Frequent errors and crashes may disrupt operations.
- *High Maintenance Costs*
  - Long-term upkeep and modifications may be expensive.

The requirements engineering is difficult. *Why?*

- **Evolving Needs**
  - Business requirements constantly change due to a dynamic environment.
- **Diverse Stakeholders**
  - Multiple stakeholders with different goals and priorities are involved in the requirements engineering process.
- **Unclear Expectations**
  - Stakeholders may not fully understand their system needs.
- **Hidden Influences**
  - Political and organizational factors often shape requirements behind the scenes.

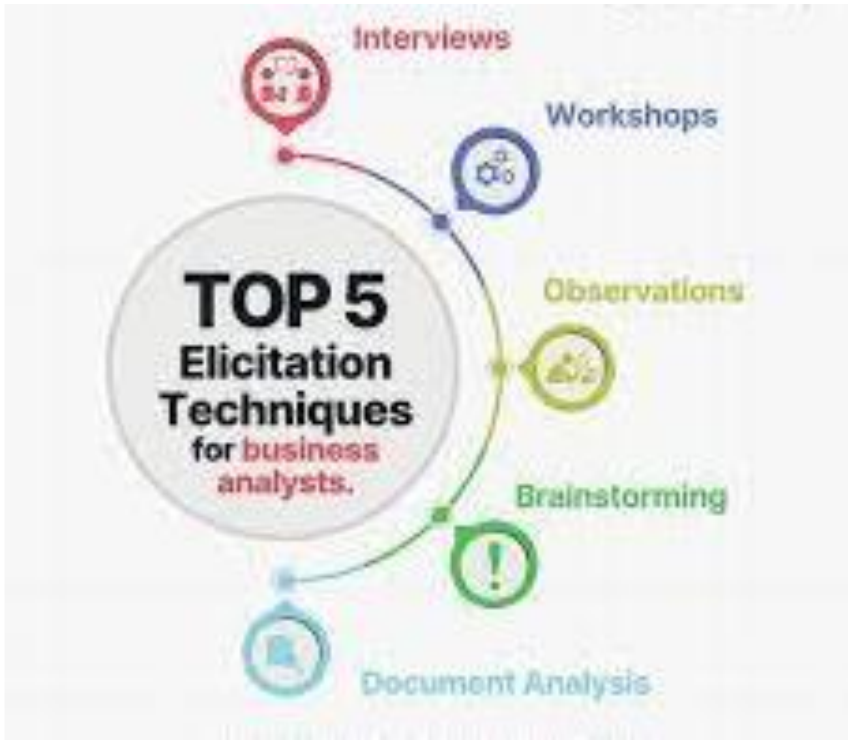
Phases of Requirements Engineering

- Tre are five types of Software Requirement Engineering Phases
  - Elicitation
  - Analysis
  - Documentation
  - Validation
  - Management



Requirements Elicitation

- Represents for requirements ***gathering***
- Identifying stakeholders
- Conducting interviews, surveys, and workshops
- Observing user workflows
- Reviewing existing documentation
- Identify stakeholder needs through interviews, surveys, observation, and brainstorming.
  - ***Example:*** What features should a customer want in an e-commerce app?



Requirements Analysis

- Checking feasibility
- Resolving conflicts among stakeholders
- Prioritizing requirements
- Defining constraints
- Resolve conflicts, assess feasibility, and prioritize key requirements.
  - Example: Balancing strong security with a simple login process.

- *Requirements Specification*
  - Represents for Requirement Documentation
  - Writing Software Requirement Specification (SRS)
  - Using structured formats like IEEE standards
  - Including functional, non-functional, and system requirements
  - Clearly write requirements in an SRS (Software Requirements Specification) document.
    - ***Example:*** The system shall support secure two-factor authentication.

- ***Requirements Validation***
  - Ensuring completeness and correctness
  - Conducting reviews and walkthroughs
  - Prototyping and simulations
  - Ensure accuracy through reviews, prototyping, and stakeholder feedback.
    - Example: Checking if all necessary features are included before development starts.

- ***Requirements Management***
  - Handling changes in requirements
  - Version control and traceability
  - Managing dependencies
  - Track, update, and control changes as business needs evolve.
    - Example: If laws change, update security policies accordingly.

Types of requirements

- **Very General Requirements**
  - Define the broad purpose and goals of the system without specifying details.
  - Example: *The system should support multiple Language*
- **Functional Requirements**
  - Describe specific functionalities that the system **must provide**.
  - Example: *The system shall allow users to select up to four Language*
- **Implementation Requirements**
  - Specify **technical constraints** and technologies to be used in development.
  - Example: *The system must be developed using PHP with the CodeIgniter 4 framework and a MySQL database.*

- **Performance Requirements**
  - Define the **minimum acceptable system performance** in terms of speed, capacity, or response time.
  - Example: *The system should process and display Bingo results within 2 seconds for 95% of users.*
- **Usability Requirements**
  - Specify how **easily and efficiently users can interact** with the system.
  - Example: *New users should be able to complete the registration process within 1 minute.*
- **Domain Requirements**
  - Constraints related to the industry
  - *A banking system must follow financial regulations.*

Common Requirements problems

- **Mismatch with Customer Needs**
  - The system requirements **do not accurately represent** what the customer actually needs.
  - *Example: A banking app is built without multi-currency support, but the customer needed it.*

- **Inconsistent or Incomplete Requirements**
  - Conflicting or missing details lead to **ambiguity and errors** during development.
  - *Example: One requirement states that users can reset passwords via email, but another says only administrators can do it.*

- **High Cost of Changes**
  - Modifying requirements **after approval** is expensive and time-consuming.
  - *Example: Late changes to database design may require rewriting large parts of the system.*
- **Miscommunication Among Stakeholders**
  - Customers, system analysts, and software engineers **misinterpret** requirements.
  - *Example: The customer wants a "simple" login, but developers assume they want a complex multi-factor authentication system.*

Feasibility studies

- A feasibility study decides whether or not the proposed system is worthwhile.
- A short focused study that checks
  - If the system contributes to organizational objectives;
  - If the system can be implemented using current technology, within given cost and schedule constraints;
  - If the system can be integrated with other systems that are already in place.

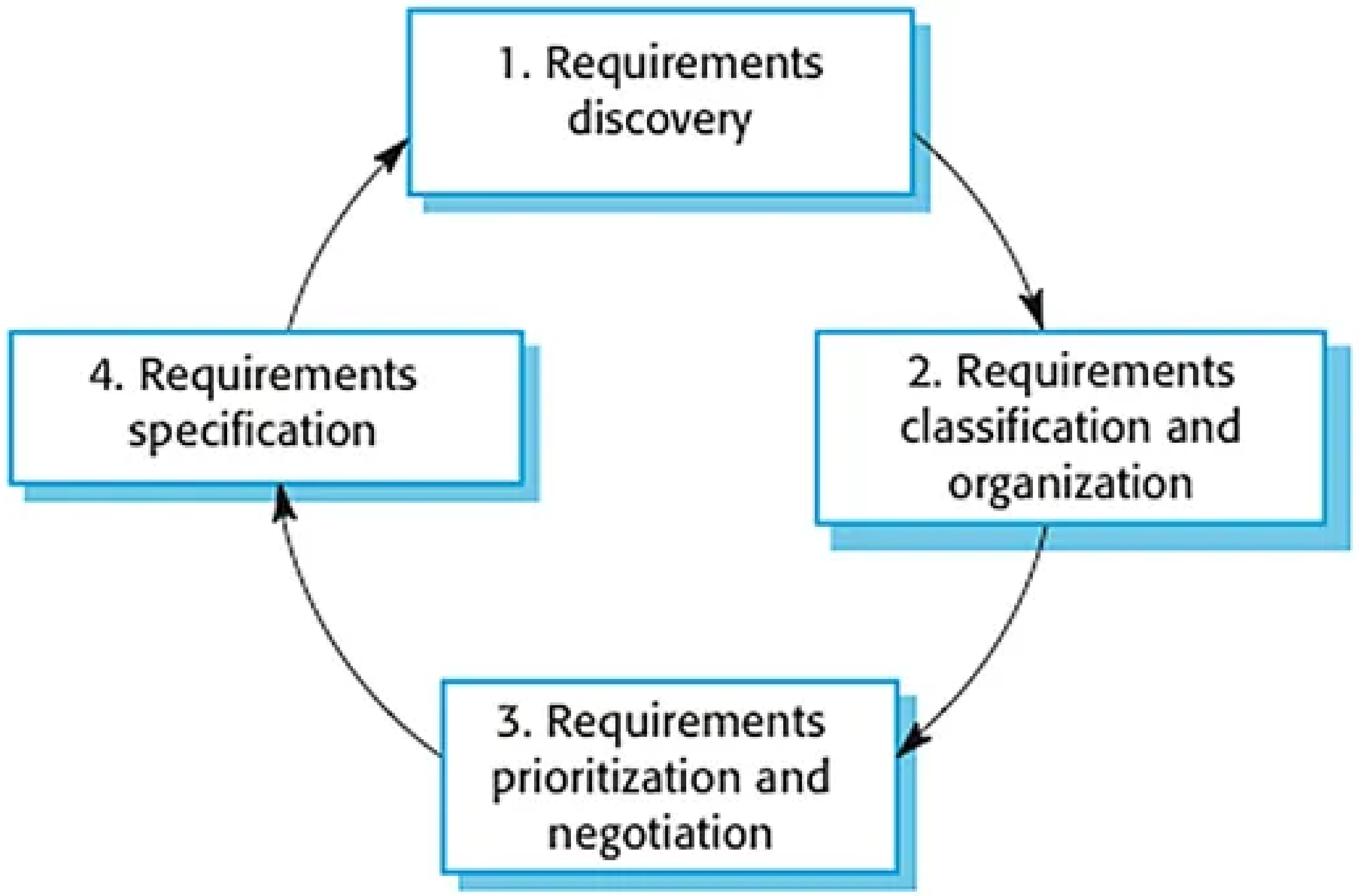
Feasibility study implementation

- Feasibility study involves information assessment (what is required), information collection and report writing.
- Questions for people in the organization for information assessment and collection:
  - What if the system wasn’t implemented?
  - What are current process problems?
  - How will the proposed system help?
  - What will be the integration problems?
  - Is new technology needed? What skills?
  - What facilities must be supported by the proposed system?
  - Feasibility study report should make a recommendation about the development to continue or not.

E&A Process activities

- Requirements discovery
  - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- Requirements classification and organization
  - Group related requirements and organizes them into coherent clusters.
- Prioritization and negotiation
  - Prioritizing requirements and finding and resolving requirements conflicts.
- Requirements documentation
  - Requirements are documented and input into the next round of the spiral.

The requirements elicitation & analysis Process


1. Requirements discovery
2. Requirements classification and organization
3. Requirements prioritization and negotiation
4. Requirements specification

- **Requirements Discovery (Elicitation)**
  - Identify and gather requirements from stakeholders through various techniques.
  - **Techniques Used:**
  - **Interviews** – Direct discussions with users and stakeholders.
  - **Surveys & Questionnaires** – Collect structured feedback.
  - **Workshops & Brainstorming** – Group discussions to generate ideas.
  - **Observation** – Watching users interact with existing systems.
  - **Prototyping** – Creating models to visualize requirements.
    - *Example: A healthcare app team interviews doctors to understand what features they need.*

- **Requirements Classification & Organization**
- Categorize requirements into meaningful groups.
  - **Types of Requirements:**
  - **Functional** – What the system must do.
  - **Non-Functional** – Performance, security, usability.
  - **Domain-Specific** – Industry-specific constraints.
  - *Example: In a banking system, security requirements would be classified under non-functional requirements.*

- **Requirements Prioritization & Negotiation**
  - Determine which requirements are most important and feasible.
  - **Prioritization Methods:**
  - **MoSCoW Method** (Must-have, Should-have, Could-have, Won't-have).
  - **Cost-Benefit Analysis** – Evaluating value vs. cost.
  - *Example: In an e-commerce app, the "Add to Cart" feature is a Must-have, while AI-based recommendations could be optional.*

- **Requirements Specification (Documentation)**
  - Clearly document requirements in an **SRS (Software Requirements Specification)** document.
  - **Common Formats:**
    - User stories
    - Use case diagrams
    - Structured text-based documentation
    - *Example: "The system shall allow users to reset their password via email."*

- **Requirements Validation (Ensuring Accuracy)**
  - Verify that requirements are complete, consistent, and meet stakeholder needs.
- **Validation Methods:**
  - **Prototyping** – Creating mockups.
  - **Reviews & Inspections** – Checking for errors.
  - **Stakeholder Confirmation** – Getting formal approval.
  - 🔹 *Example: A prototype of a food delivery app is shown to restaurant owners for feedback before finalizing requirements.*

Requirements change

- The priority of requirements from different viewpoints changes during the development process.
- System customers may specify requirements from a business perspective that conflict with end-user requirements.
- The business and technical environment of the system changes during its development.

Question