Chapter Two

Requirement Engineering Process

Compiled by: | Sinodos G

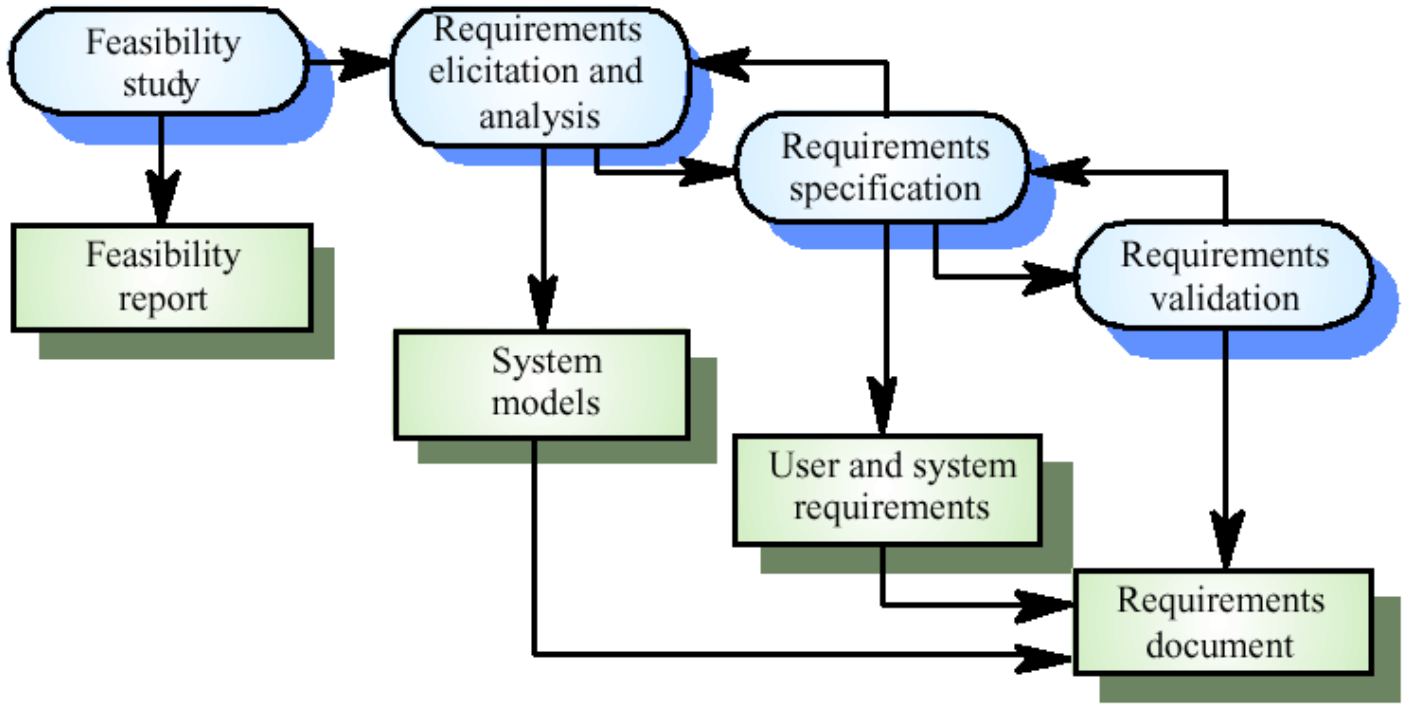
Dire Dawa - 2017 E.C

Requirement Engineering Process

- Refer to the systematic approach used to gather, analyze, document, validate, and manage software or system requirements.
- Processes used to discover, analyse and validate system requirements
  - These processes ensure that the final product meets stakeholder needs and business objectives while minimizing errors and misunderstandings.
- The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements

- The Requirement Engineering Process consists the following five common steps

  - Feasibility study
  - Requirements elicitation & analysis
  - Requirements Specification
  - Requirements validation
  - Requirements management



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 organisational objectives
  - If the system can be engineered using current technology and within budget
  - If the system can be integrated with other systems that are used

Feasibility study implementation

- Based on information assessment (what is required), information collection and report writing
- Questions for people in the organisation
  - 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?

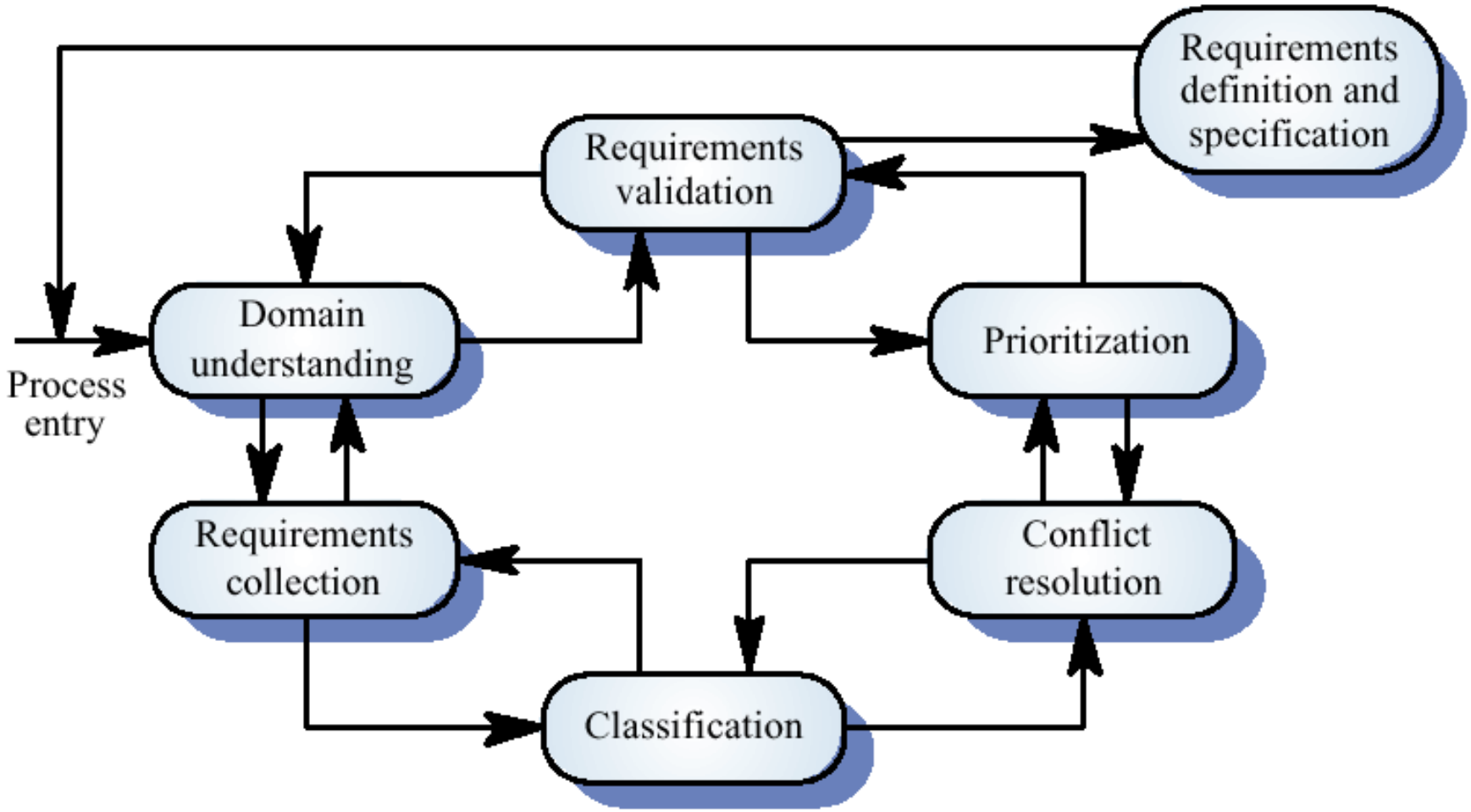
Elicitation and analysis

- Sometimes called requirements elicitation or requirements discovery
- Involves technical staff working with customers to find out about
  - the application domain,
  - the services that the system should provide and
  - the system's operational constraints
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called *stakeholders*

Problems of requirements analysis

- Stakeholders don't know what they really want
- Stakeholders express requirements in their own terms
- Different stakeholders may have conflicting requirements
- Organisational and political factors may influence the system requirements
- The requirements change during the analysis process.
- New stakeholders may emerge and the business environment change

The requirements analysis process



Process activities

- List of process Activities
  - Domain understanding
  - Requirements collection
  - Classification
  - Conflict resolution
  - Prioritisation
  - Requirements checking
  - Requirement definition and specification

Requirement Process System Models

- used in the **requirements engineering process** to help stakeholders and developers visualize, understand, and analyze system requirements.
- These models provide a structured way to represent system functionality, behavior, and constraints.
- Different models produced during the requirements analysis activity
- Requirements analysis involve three structuring activities
  - Partitioning:- Identifies the structural (part-of) relationships between entities
  - Abstraction:- Identifies generalities among entities
  - Projection:- Identifies different ways of looking at a problem

- Advantages of System Models in Requirements Engineering
  - Clarifies requirements before implementation.
  - Improves communication between stakeholders and developers.
  - Reduces errors and ambiguities in requirements.
  - Enhances documentation and future system maintenance.

- **List of Common System Models in Requirements Engineering**

- **Data Flow Model (DFD - Data Flow Diagram)**
  - Represents how data moves through the system.
  - Shows processes, data stores, external entities, and data flow.
  - Helps in understanding input, processing, and output of data.

Example:

- A **DFD for an online banking system** showing user login, transaction processing, and account updates.

- **Use Case Model**
  - Describes system interactions from a user's perspective.
  - Contains actors (users, systems) and use cases (functionalities performed by the system).
  - Helps define functional requirements clearly.

  Example:
    - A use case diagram for an e-commerce website showing actions like "Login," "Add to Cart," "Checkout," and "Make Payment."

- **Entity-Relationship Diagram (ERD)**
  - Represents data relationships in a structured way.
  - Uses entities (objects), attributes (properties), and relationships to model the system's data.

  Example:
  - An ERD for a university system showing entities like "Student," "Course," and "Instructor" with relationships such as "Enrolls In."

- **State Transition Model (State Diagram)**
  - Represents different states of a system and how it transitions between them based on events.
  - Useful for modeling real-time systems and interactive applications.

  Example:
    - A state diagram for a traffic light system, showing transitions between "Red," "Yellow," and "Green" states based on time.

- **Class Diagram (UML Model)**
  - Represents the object-oriented structure of the system.
  - Defines classes, attributes, methods, and relationships between classes.
  - ***Example:***
    - A class diagram for a library management system with "Book," "Member," and "Librarian" classes and their associations.

- **Workflow Model (Activity Diagram)**
  - Represents the sequence of activities in a process.
  - Helps in business process modeling and requirement analysis.

  Example:

  - An activity diagram for an order processing system, showing steps from "Order Placed" to "Payment" to "Shipment."

Viewpoint-Oriented Elicitation

- A structured approach to gathering requirements by considering multiple perspectives (viewpoints) from different stakeholders involved in the system.
- Each viewpoint represents a different concern, interest, or role within the system, ensuring a comprehensive requirement collection process.
- Stakeholders represent different ways of looking at a problem or problem viewpoints
- This multi-perspective analysis is important as there is no single correct way to analyse system requirements

Advantages of Viewpoints in Requirements Elicitation

- Captures diverse perspectives to avoid missing requirements.
- Helps identify conflicts between stakeholder needs.
- Improves communication and understanding of system goals.
- Ensures better traceability and completeness of requirements.

Types of Viewpoints

- **Interactor Viewpoints**
  - Represent users who interact directly with the system.
  - Example: Customers, employees, administrators.
  - Example: In a banking system
    - a "Customer" wants easy online transactions,
    - a "Bank Manager" needs detailed financial reports.

- **Indirect Stakeholder Viewpoints**
  - Represent people or systems affected by the system but do not directly interact with it.
  - Example: Regulators, auditors, external systems.
  - Example Use Case: In an IoT smart home system, government regulations on energy efficiency must be considered.

- **Domain Viewpoints**
  - Represent industry standards, laws, and technical constraints.
  - Example: ISO standards for software.
  - Example: In a hospital management system, the Health Ministry's regulations on patient data security must be incorporated.

- **Development Viewpoints**
  - Represent developers, testers, and maintainers who build and manage the system.
  - Example: System architects, software engineers, cybersecurity teams.
  - **Example:** a **cloud-based SaaS application**, developers focus on scalability and performance.

- **Steps in Viewpoint-Oriented Elicitation**
  - **Identify Viewpoints** – List all stakeholders and categorize them into interactor, domain, indirect, or development viewpoints.
  - **Gather Requirements** – Use interviews, surveys, observation, and brainstorming for each viewpoint.
  - **Analyze and Prioritize** – Identify conflicting needs and prioritize based on business goals.
  - **Validate with Stakeholders** – Review requirements to ensure completeness and correctness.
  - **Document and Manage** – Store requirements in an organized manner for traceability.

A **Process Model**

- a process model defines the structured approach followed during software development.
- activities for designing, implementing, and testing a software system.
- It provides guidelines on how to plan, develop, test, and maintain software efficiently.
- an **abstract representation** of the development process.
- Different process models exist to suit various project requirements, risks, and constraints.

- A model will define the following:
  - The tasks to be performed
  - The input and output of each task
  - The pre and post-conditions for each task
  - The flow and sequence of each task
- Goal of a software process model
  - To provide guidance for controlling and coordinating the tasks to achieve the end product and objectives as effectively as possible.

Types of process model

- Refer to SDLC models (Software Development Life Cycle models).
- There are many kinds of process models for meeting different requirements. List of the most popular process models are:-
  - Waterfall model
  - V-model
  - Incremental model
  - RAD model
  - Agile model
  - Iterative model
  - Prototype model
  - Spiral model

- **Factors in choosing a software process**
- Choosing the right software process model for your project can be difficult.
- If you know your requirements well, it will be easier to select a model that best matches your needs.
  - Project requirements
  - Project size
  - Project complexity
  - Cost of delay
  - Customer involvement
  - Familiarity with technology
  - Project resources

Waterfall Model

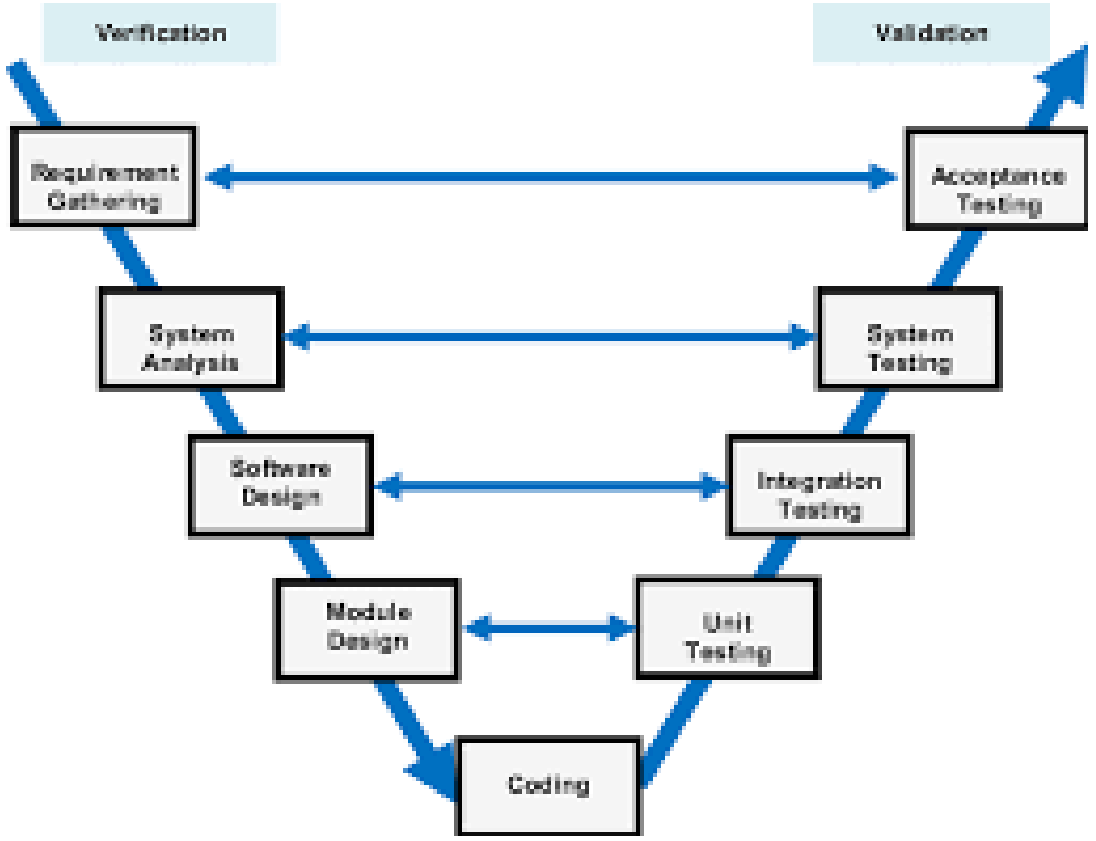
- a **sequential, plan driven-process** where you must plan and schedule all your activities before starting the project.
- Each activity in the waterfall model is represented as a separate phase arranged in linear order.
  - Requirements
  - Design
  - Implementation
  - Testing
  - Deployment
  - Maintenance

- Each of these phases produces one or more documents that need to be approved before the next phase begins.
- However, in practice, these phases are very likely to overlap and may feed information to one another.
- The software process isn't linear, so the documents produced may need to be modified to reflect changes.

- Characteristics:
  - Simple and structured.
  - Each phase must be completed before moving to the next.
  - Suitable for well-defined projects with clear requirements.
- Limitations:
  - Difficult to accommodate changes after the design phase.
  - Late-stage testing may reveal major issues.
- Example: Government or military projects with strict documentation requirements.

V-Model

- Called as Verification and Validation model
- an extension of the waterfall model.
- All the requirements are gathered at the start and cannot be changed.
- For every phase in the development cycle, there is an corresponding testing activity.



- **V-Model (Validation and Verification Model)**
  - **Phases:** Similar to Waterfall but with testing integrated at each stage.
  - **Characteristics:**
    ✓ Ensures early defect detection.
    ✓ Parallel testing and development improve reliability.
  - **Limitations:**
    - Expensive and rigid in handling requirement changes.
  - **Example:** Safety-critical systems (e.g., medical devices, aerospace).

- **Incremental Model**
- **List of Phases**
  - Requirements → Design → Implement First Module → Test → Repeat for More Modules
- **Characteristics:**
  ✓ Software is developed in small parts (increments).
  ✓ Each increment adds functionality based on feedback.
  ✓ More flexible than Waterfall.
- **Limitations:**
  ✗ Requires good planning for integration.
- **Example: L**arge-scale enterprise applications (e.g., ERP systems).

- **Spiral Model (Risk-Driven Model)**
  - **Phases:** Planning → Risk Analysis → Engineering → Evaluation (Repeated in Cycles)
  - **Characteristics:**
    ✓ Emphasizes risk assessment and iterative development.
    ✓ Suitable for high-risk and complex projects.
  - **Limitations:**
    ✗ High cost and complexity.
  - **Example:** Large, evolving projects (e.g., financial software, AI systems).

- **Agile Model (Iterative & Adaptive)**
  - an iterative and flexible software development approach that emphasizes collaboration, customer feedback, and rapid delivery.
  - allows continuous adaptation to changing requirements, making it ideal for dynamic projects.

  - **Phases**
    - Planning → Design → Development → Testing → Review (Repeated in Short Iterations)

- **Characteristics of Agile Model**
  - **Iterative & Incremental Development** – Software is built in small cycles (sprints).
  - **Customer Collaboration** – Frequent stakeholder involvement ensures alignment with needs.
  - **Flexibility to Changes** – Requirements can evolve throughout development.
  - **Continuous Testing & Integration** – Bugs are fixed early to improve quality.
  - **Self-Organizing Teams** – Teams work independently with less managerial control.

- Advantages of Agile Model
  - Faster delivery of working software.
  - Better collaboration between developers & stakeholders.
  - Higher adaptability to changes.
  - Improved software quality with continuous testing.
- **Limitations:**
  - Requires active customer involvement.
  - Less suited for strict regulatory environments.
- **Example:** Web applications, mobile apps, startups

- **DevOps Model (Development + Operations)**
  - **Phases:** Continuous Development → Integration → Deployment → Monitoring
  - **Characteristics:**
    ✓ Focuses on automation and collaboration.
    ✓ Ensures fast and stable releases.
  - **Limitations:**
    ✗ Requires strong infrastructure and expertise.
  - **Example Use Case:** Cloud-based applications, SaaS platforms.

- Choosing the Right Process Model
  - Stable, well-defined requirements? → Waterfall / V-Model
  - Gradual improvements → Incremental Model
  - High-risk project → Spiral Model
  - Rapid changes and customer feedback → Agile
  - Continuous deployment and monitoring → DevOps

- **Actors in the Requirements Engineering Process**
  - different actors (stakeholders) play crucial roles in gathering, defining, analyzing, validating, and managing requirements.
- **Customers (Clients/Business Owners)**
  - Provide business goals and high-level system expectations.
  - Approve final requirements before development starts.
  - Example: A **retail company** requesting an e-commerce platform.
- **End Users (System Users)**
  - Use the system after deployment.
  - Provide functional and usability feedback.
  - Example: Bank customers using a **mobile banking app**.

- System Analysts
  - Bridge the gap between stakeholders and developers.
  - Define, document, and refine system requirements.
  - Example: An IT consultant analyzing business needs for a healthcare system.
- Software Developers (Engineers)
  - Implement system requirements into code.
  - Provide technical insights on feasibility and constraints.
  - Example: A backend developer ensuring database structure
- Testers (Quality Assurance – QA Engineers)
  - Validate that requirements are correctly implemented.
  - Identify inconsistencies or missing functionalities.
  - Example: A QA engineer testing an online booking system's

- Project Managers
  - Ensure requirements align with project scope, budget, and timeline.
  - Manage communication between different stakeholders.
- Domain Experts
  - Provide industry-specific knowledge and regulations.
  - Ensure compliance with legal and technical standards.
- Regulatory Bodies & Auditors
  - Ensure compliance with laws, safety, and security standards.
- Competitors & Market Analysts
  - Influence system features based on market trends and competition.

- Process Support in Requirements Engineering
- refers to the methods, tools, and frameworks that help streamline and manage the **elicitation, analysis, specification, validation, and management** of system requirements.
- Process support ensures consistency, efficiency, and accuracy in requirement-related activities.

Types of Process Support

- **Methodological Support**
  - Uses structured techniques and best practices to guide requirements engineering.
  - **Examples:**
    ✓ **Agile Methods** – User stories, backlog grooming, sprint planning.
    ✓ **Use Case Modeling** – Capturing functional requirements.
    ✓ **Prototyping** – Creating visual representations before development.
- **Tool Support**
  - Uses software tools to automate, track, and document requirement-related activities.
  - **Examples:**
    ✓ **JIRA, Trello** – Agile project management & requirement tracking.
    ✓ **IBM DOORS, ReQtest** – Requirement documentation & traceability.
    ✓ **Enterprise Architect, Lucidchart** – UML diagrams & system modeling.

- **Standardization Support**
  - Follows industry standards and best practices to ensure quality and compliance.
  - **Examples:**
    ✓ **IEEE 830** – Software Requirements Specification (SRS) guidelines.
    ✓ **ISO 29148** – Standard for requirement engineering process.
    ✓ **BABOK (Business Analysis Body of Knowledge)** – Best practices for business analysis.

- **Process Automation & AI Support**
  - Uses automation and AI to enhance requirements management and analysis.
  - **Examples:**
    ✓ **Natural Language Processing (NLP)** – Analyzing stakeholder requirements.
    ✓ **AI-based Requirement Extraction** – Automatically identifying key requirements from documents.
    ✓ **Chatbots & Virtual Assistants** – Assisting with requirement elicitation.
- **Collaboration & Communication Support**
  - Enables better interaction between stakeholders, developers, and business analysts.
  - **Examples:**
    ✓ **Slack, Microsoft Teams, Zoom** – Remote collaboration.
    ✓ **Google Docs, Confluence** – Shared documentation and real-time editing.

- **Advantages of process Support**
  - Reduces **errors and ambiguities** in requirements.
  - Improves **efficiency and productivity** in managing requirements.
  - Ensures **better traceability** and compliance with standards.
  - Enhances **team collaboration and communication**.

Process Improvement in Requirements Engineering

- refers to the continuous refinement and enhancement of requirement-related activities to increase efficiency, accuracy, and overall project success.
- It involves identifying weaknesses in the current process and implementing changes to improve quality, reduce costs, and minimize risks.
- Some Key Aspects of Process Improvement
  - Identifying Process Weaknesses
  - Applying Standardized Models for Improvement
  - Automation & Tool Integration
  - Stakeholder Collaboration & Communication Enhancement
  - Continuous Training & Skill Development
  - Regular Review and Metrics-Based Evaluation

Question