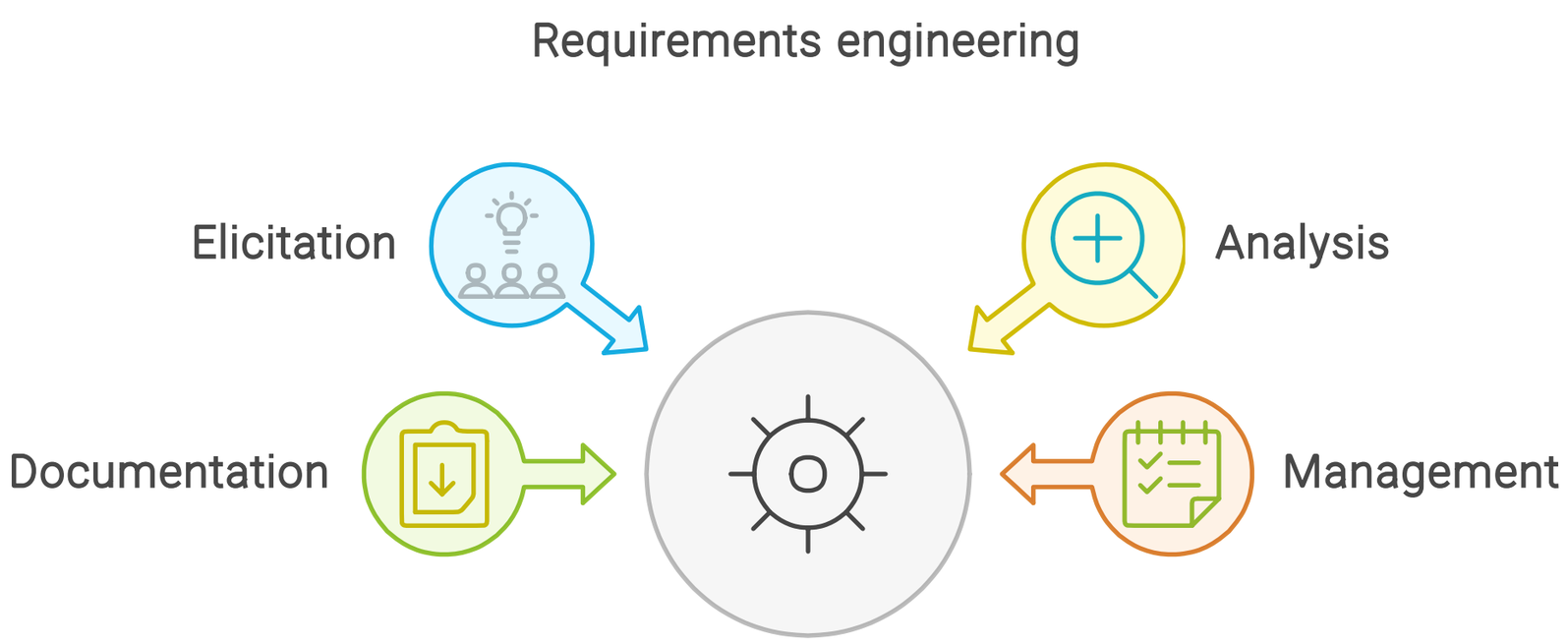
Software Requirements

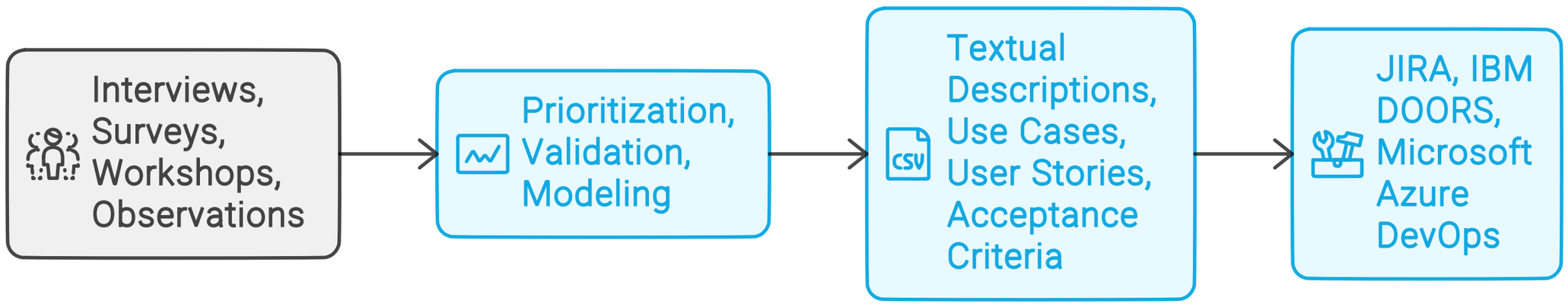
Topic: Software Requirements Engineering
Introduction:
Software Requirements Engineering is the process of eliciting, analyzing, documenting, and managing requirements for software development projects.
It lays the foundation for successful software development by defining what needs to be built and ensuring alignment with stakeholder needs and expectations.
Key Activities:

- Elicitation: Gathering requirements from stakeholders, including end-users, customers, and domain experts.
- Analysis: Analyzing and prioritizing requirements to understand their impact on the software system and its stakeholders.
- Documentation: Documenting requirements in a clear, concise, and unambiguous manner to serve as a basis for design, development, and testing.
- Management: Managing requirements throughout the software development lifecycle, including traceability, change control, and versioning.



Importance:
Helps to establish a common understanding among stakeholders.
Minimizes the risk of project failure by ensuring that the software meets stakeholder needs and expectations.
Provides a basis for estimating project scope, schedule, and cost.

Topic: Elicitation, Analysis, and Documentation of Requirements
Elicitation:
Involves techniques such as interviews, surveys, workshops, and observations to gather requirements from stakeholders.
Focuses on understanding stakeholder needs, goals, and constraints.
Analysis:
Involves analyzing and prioritizing requirements to identify conflicts, ambiguities, and inconsistencies.
Techniques such as requirements prioritization, requirements validation, and requirements modeling are used.
Documentation:
Involves documenting requirements in various formats, such as textual descriptions, use cases, user stories, and acceptance criteria.
The documentation should be clear, concise, and unambiguous to facilitate communication and understanding among stakeholders.
Tools:
Requirements management tools, such as JIRA, IBM DOORS, or Microsoft Azure DevOps, are often used to capture, organize, and track requirements throughout the project lifecycle.



Topic: Use Case Modeling
Definition:
Use case modeling is a technique for capturing functional requirements from the perspective of system users.
It describes how users interact with the system to achieve specific goals or tasks.
Components:
Actors: Users or external systems that interact with the system.
Use Cases: Descriptions of system functionality from the user's perspective.
Relationships: Associations between actors and use cases, such as "extends" and "includes" relationships.
Benefits:
Provides a visual representation of system functionality that is easy to understand and communicate.
Helps to identify system boundaries, interfaces, and interactions.
Serves as a basis for system design, development, and testing.
Topic: User Stories and Acceptance Criteria
User Stories:
User stories are short, simple descriptions of a feature or functionality from the perspective of an end-user.
They follow a simple template: "As a [user role], I want [feature] so that [benefit]."
User stories are often written on index cards or sticky notes and are used to facilitate conversations between stakeholders and development teams.
Acceptance Criteria:
Acceptance criteria define the conditions that must be met for a user story to be considered complete and ready for acceptance by stakeholders.
They are typically written in a "Given-When-Then" format and specify the expected behavior or outcomes of the user story.
Acceptance criteria serve as a basis for testing and validation during the development process.

Use Case Modeling