

Introduction and Requirement engineering

Chapter 1 and 2

Introduction

Software is a computer program that follows defined procedural steps or algorithms to perform specific tasks.

Software engineering is the disciplined application of scientific principles to design, develop, document, operate, and maintain software systems.

Software products are complete software systems delivered with documentation explaining installation and use. They can be:

- **Generic products:** designed for the general market (e.g., Word processors, databases).
- **Customized products:** developed for a specific client or purpose (e.g., Business process systems, control systems).

A **process** is a structured set of activities, actions, and tasks used to create a work product.

Software Application Domains

- 1. System software:** Supports other programs (e.g., compilers, operating systems, network software).
- 2. Application software:** Designed for end-user tasks.
- 3. Engineering/scientific software:** Used for technical and scientific computations.
- 4. Embedded software:** Built into devices to control their functions (e.g., car fuel control, braking).
- 5. Product-line software:** Developed for a family of related products.
- 6. Web/mobile applications:** Software designed for online or mobile use.
- 7. Artificial intelligence software:** Uses heuristics and learning techniques for complex problem-solving (e.g., robotics, decision systems, pattern recognition, game playing).

Software process

A **software process** is a set of related activities that results in the creation of a software product.

All software processes share four **fundamental activities**:

1.Specification (Requirements Engineering): Defining what the software should do and the constraints on its operation.

2.Development (Design and Implementation): Designing the structure and writing the code for the software.

3.Validation: Ensuring the developed software meets customer requirements.

4.Evolution: Modifying the software to adapt to new customer or market needs.

These activities are organized differently depending on the **development model** used:

- In the **Waterfall model**, they occur in a strict sequence.
- In **Incremental development**, they are performed in overlapping or iterative cycles.

The execution of these activities also varies according to the **type of software**, the **team members**, and the **organizational structure** involved.

Software Requirements and Requirements Engineering

System requirements describe what a system should do, the services it must provide, and the constraints under which it operates. These requirements express the **customers' needs** for a system designed to achieve specific goals, such as controlling devices, processing orders, or retrieving information.

Requirements engineering (software specification) is the process of **understanding, defining, and documenting** the required services and operational constraints of the system.

It includes four main activities:

- 1. Feasibility study:** Assessing whether the proposed system is practical and achievable.
- 2. Requirements elicitation and analysis:** Gathering and examining user needs and expectations.
- 3. Requirements specification:** Formally documenting the system's functions and constraints.
- 4. Requirements validation:** Ensuring the documented requirements accurately represent user needs.

Types of Software Requirements

Software requirements describe what a system should do and the conditions under which it operates. They are categorized as follows:

1. User Requirements

- High-level statements describing the **services** the system should provide and the **constraints** it must follow.
- Focus only on the **external behavior** of the system.
- Written in **natural language**, using simple tables or diagrams (not code or formal notation).
- Used mainly by **clients, end-users, and system architects**.

2. System Requirements

- More detailed descriptions of the **functions, services, and operational constraints** of the software.
- **Do not** specify how the system will be designed or implemented.
- Used by **developers, engineers, and system architects** for system construction.

Types of Software Requirements

Classification by Nature

A. Functional Requirements

- Define **what services or functions** the system should perform, how it reacts to inputs, and how it behaves in specific situations.
- Depend on the **software type, user expectations, and organizational standards**.

B. Non-Functional Requirements

- Define **constraints** on system operation, such as **performance, timing, standards, or development limitations**.
- Usually apply to the **entire system**, not individual components.
- More challenging to link directly to components because they often affect the **overall architecture** (e.g., minimizing communication to improve performance).

Relationship Between Non-Functional and Functional Requirements

A single **non-functional requirement** (for example, **security**) can lead to:

- **New functional requirements** — additional system services needed to meet that non-functional goal (e.g., user authentication).
- **Restrictions on existing requirements** — limiting how current system functions operate to ensure compliance with the non-functional constraint.

Requirements Engineering Activities

* Feasibility Study

A **feasibility study** estimates whether user needs can be met using existing software and hardware technologies. It evaluates if the proposed system is **technically possible, cost-effective, and achievable within budget and time constraints.**

The study should be **quick and inexpensive**, and its outcome determines whether to proceed with detailed system analysis.

Requirements Elicitation and Analysis

Requirements Elicitation and Analysis This process identifies **user and system requirements** by defining the application domain, required services, performance expectations, and system constraints.

Stakeholders include everyone who influences requirements—such as **end-users, engineers, managers, domain experts, and union representatives**.

The process consists of four main activities:

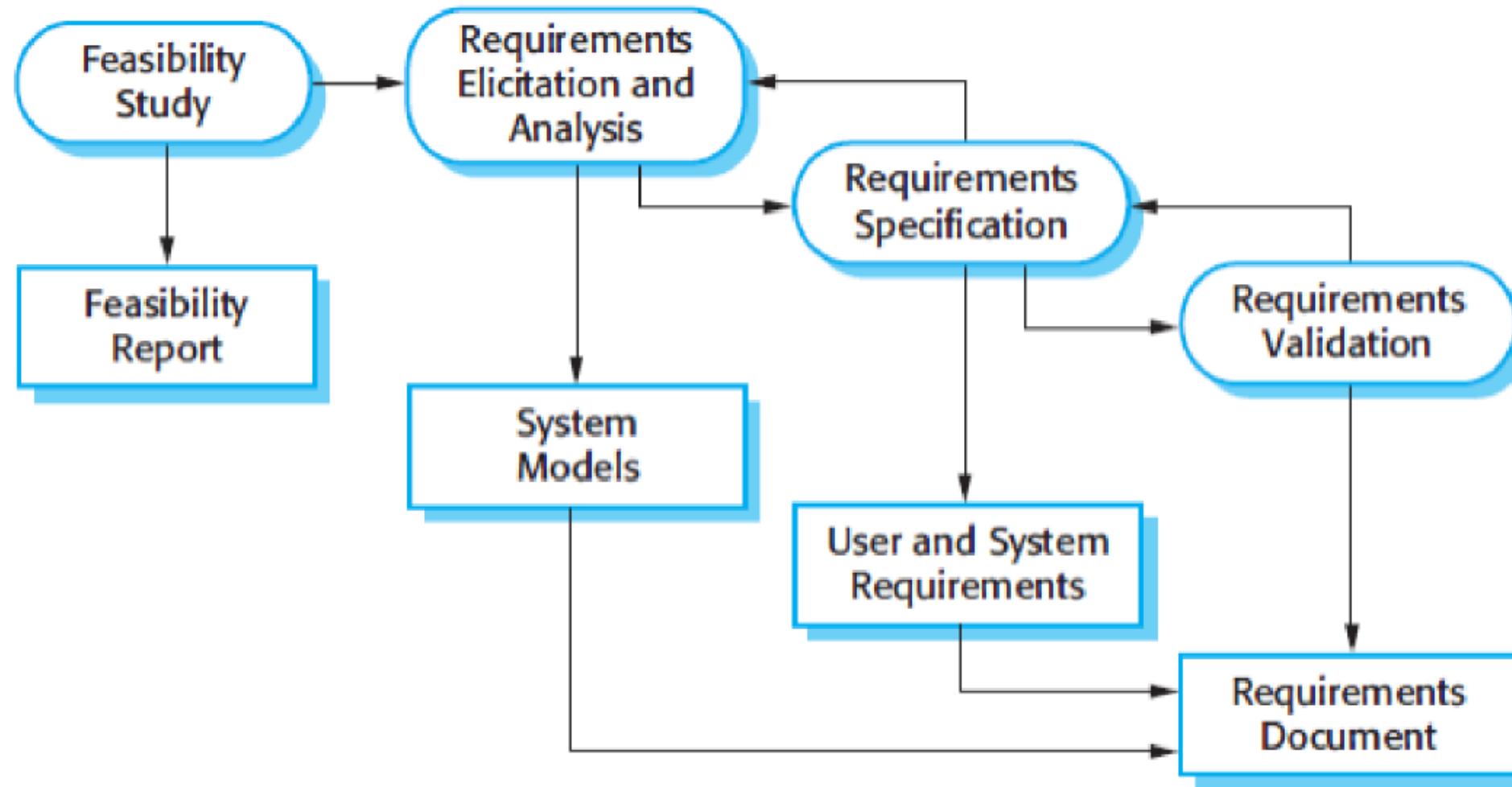
1. Requirements Discovery (Elicitation):

Collecting information about the desired system and existing systems through **documents, stakeholder interaction, scenarios, and prototypes** to clarify needs.

2. Requirements Classification and Organization:

Grouping and structuring related requirements into **coherent clusters**—often aligned with the **system's architecture** and sub-systems.

Requirement engineering process (exam)



Requirements Elicitation and Analysis

1. Requirements Prioritization and Negotiation:

Handling **conflicts among stakeholders** by prioritizing and reconciling differences through **discussion and compromise**.

2. Requirements Specification:

Documenting and formalizing the discovered and agreed-upon requirements for use in subsequent phases.

Principles guiding analysis methods:

1. Understand and represent the **information domain**.
2. Define the **functions** the software performs.
3. Represent the **behavior** in response to external events.
4. Partition models into **layers or hierarchies** to show increasing detail.
5. Move progressively from **essential information** to **implementation details**.

Requirements Specification

This activity involves translating analyzed information into a **Software Requirements Specification (SRS)** — a formal document describing both user and system requirements.

The requirements must be **clear, unambiguous, complete, consistent, and easy to understand**.

The **SRS document** serves as an official agreement and may combine or separate user and system requirements depending on complexity.

Used by:

- 1. System customers** – to confirm the system meets their needs.
- 2. Managers** – to plan bids and manage project development.
- 3. System engineers** – to understand what system to build.
- 4. Test engineers** – to design validation tests.
- 5. Maintenance engineers** – to understand the system's structure and relationships.

Requirements Validation

Requirements validation is the process of reviewing gathered, analyzed, and documented requirements to ensure their **accuracy, completeness, and feasibility**. It is essential because **errors in the requirements document** can cause major rework and high costs if discovered during or after system development.

During validation, several types of **checks** are performed:

- 1. Validity check:** Ensures the system functions truly reflect user needs; additional or revised functions may be identified.
- 2. Consistency check:** Confirms that no requirements conflict or contradict each other.
- 3. Completeness check:** Verifies that all intended functions and constraints are included.
- 4. Realism check:** Ensures requirements are **practical and achievable** within current technology, budget, and schedule limits.
- 5. Verifiability check:** Confirms each requirement can be **tested and verified**, preventing future disputes between the customer and developer.

Requirements Modeling

Requirements modeling involves creating visual representations (models) that help software engineers understand and define what the system should do. These models describe the **data, functions, and behavior** of the system.

Types of Models:

1. Data Models:

Represent the **information domain** of the problem — how data are stored and related.

Example: Entity Relationship Diagram (ERD).

2. Functional Models:

Show how the software **processes data** through **input, processing, and output** operations.

Examples: Data Flow Diagram (DFD) and Control Flow Diagram (CFD).

3. Behavioral Models:

Describe how the software **responds to external events** and **changes its state** (e.g., waiting, computing, or printing).

Example: State Transition Diagram (STD).

Requirements Modeling

Objectives of Requirements Modeling:

1. To clearly describe **what the customer requires**.
2. To provide a **foundation for software design** (architecture, interface, components).
3. To define a **set of requirements** that can be **validated** after the software is built.

Requirements Management

Requirements management is the process of **tracking, controlling, and updating** system requirements as they evolve.

In large software systems, requirements **change frequently** because all problems cannot be fully defined at the beginning. After the **initial requirements** are set, stakeholders often introduce **new or modified requirements**.

The engineering team must:

- Add and integrate new requirements with existing ones.
- Update relationships among requirements to maintain consistency.
- Establish a **formal change-control process** to evaluate, approve, and document modifications and link them to the corresponding system requirements.

Thank you