Requirements Engineering

```
Module Code: ELEE1149

Module Name: Software Engineering

Credits: 15

Module Leader: Seb Blair BEng(H) PGCAP MIET MIHEEM FHEA
```

Download as a PDF

What is Requirements Engineering?

- **Definition**: A systematic process of defining, documenting, and maintaining software requirements.
- **Purpose**: Ensure that software meets stakeholder needs and expectations.
- **Outcome**: A clear, agreed-upon set of requirements.

Importance of Requirements Engineering

- Avoid misunderstandings between stakeholders and developers.
- Reduce the risk of project failure.
- Ensure timely delivery of software within budget.
- Provide a basis for testing and validation.

Key Stages of Requirements Engineering

1. **Elicitation**
    - Identifying the needs of stakeholders.
2. **Analysis**
    - Refining and prioritizing requirements.
3. **Specification**
    - Documenting requirements in a structured format.
4. **Validation**
    - Ensuring requirements meet stakeholder expectations.
5. **Management**
    - Tracking and updating requirements throughout the project lifecycle.

Requirements Types

- **Functional Requirements**
  - Define system behavior and features.
  - Example: "The system shall allow users to log in using a username and password."
- **Non-functional Requirements**
  - Specify system qualities and constraints.
  - Example: "The system shall respond to user actions within 2 seconds."

Elicitation Techniques

- **Interviews**
- **Workshops**
- **Surveys and Questionnaires**
- **Observation**
- **Document Analysis**
- **Prototyping**

Tools for Requirements Engineering

- **Modeling Tools**
  - UML diagrams
  - Use case diagrams
- **Requirement Management Tools**
  - Jira
  - Kanban
  - IBM DOORS
  - Trello
- **Prototyping Tools**
  - Figma
  - Adobe XD

Challenges in Requirements Engineering

- Ambiguous requirements.
- Changing stakeholder needs.
- Communication barriers.
- Technical feasibility issues.

Best Practices

1. Engage stakeholders early and often.
2. Use clear and consistent language.
3. Validate requirements frequently.
4. Document assumptions and constraints.
5. Prioritize requirements based on business value.
6. Use visual models to complement text-based requirements.

Types of Requirements

User Requirements

- High-level descriptions aimed at customers.
- Example: "The system shall allow users to reset their password."

System Requirements

- Detailed technical specifications for developers.
- Example: "The system shall send an email with a password reset link within 2 seconds."

System Stakeholders

- Any person or organization who is affected by the system in some way and so who has a legitimate interest
- Stakeholder types
  - End users
  - System managers
  - System owners
  - External stakeholders

Agile methods and requirements

- Many agile methods argue that producing detailed system requirements is a waste of time as requirements change so quickly. - Therefore, the requirements document is always out of date.
- Agile methods usually use incremental requirements engineering and may express requirements as ‘user stories’.
- This is practical for business systems but problematic for systems that require pre-delivery analysis (e.g. critical systems) or systems developed by several teams.

Managing Requirements Changes

- Business and technical environments evolve.
- Address conflicting user needs through negotiation.
- Keep the requirements document up-to-date.

Importance of Requirements Engineering

- Reduces project risks and ensures system success.
- Aligns stakeholder expectations with delivered functionality.
- Provides a basis for contracts and development plans.

Functional vs Non-Functional Requirements

Functional Requirements

- Define specific behaviors or functions of the system.
- Example: "The system shall allow users to log in with their email and password."

Non-Functional Requirements

- Define system constraints like performance or usability.
- Example: "The system shall respond to user actions within 200ms."

Functional and Non-functional Requirements: A Case Study - Online Bookstore

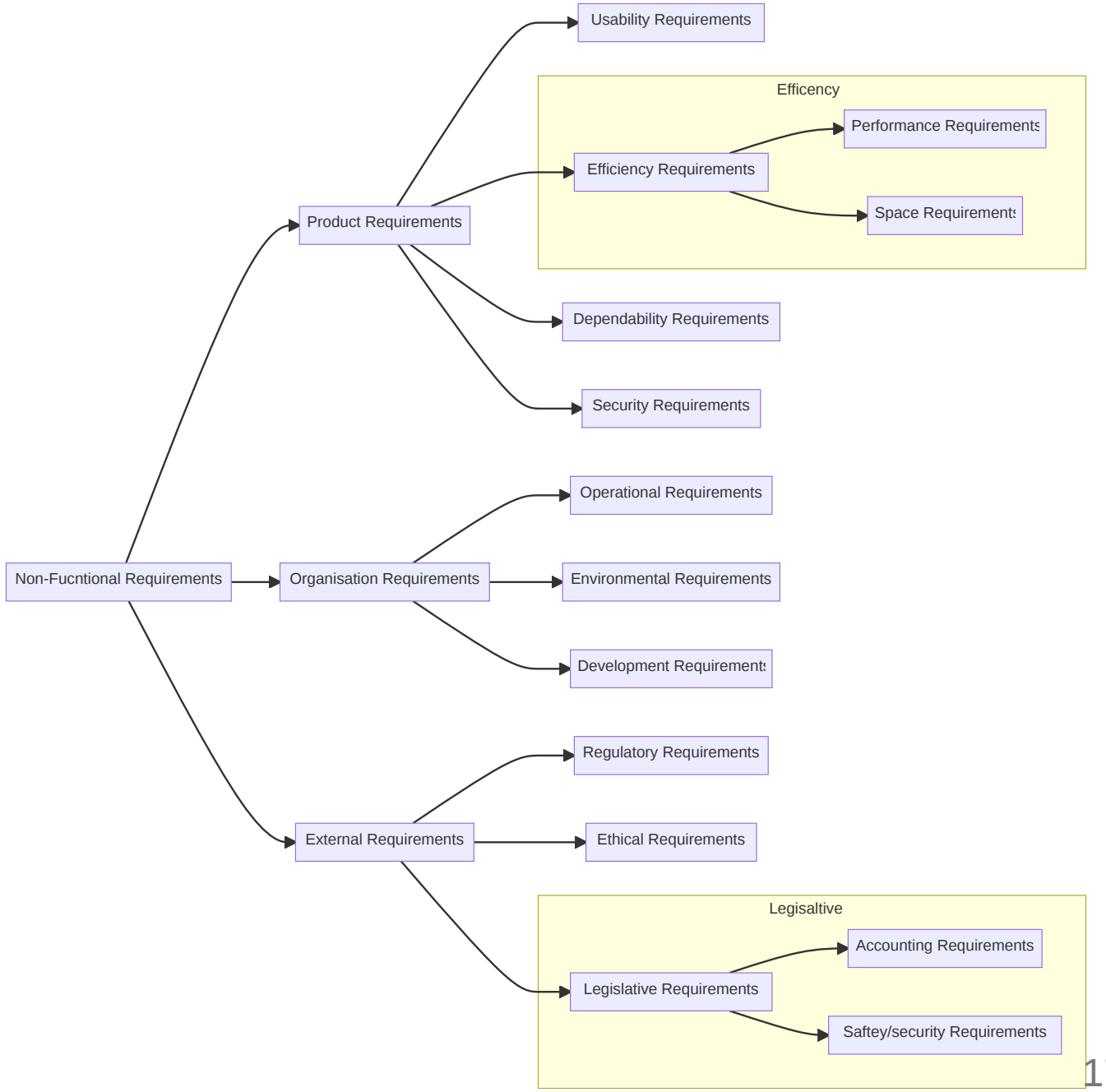
Functional Requirements:

- Users shall be able to create an account with their email and password.
- The system shall allow users to browse books by categories and search by title or author.
- Users shall be able to add books to a shopping cart and complete the purchase using a credit card.
- Admins shall be able to add, update, or remove books from the inventory.

Non-functional Requirements:

- The website shall load the homepage within 3 seconds under normal network conditions.
- All user passwords shall be stored securely using encryption.
- The system shall handle up to 10,000 simultaneous users without performance degradation.
- The application shall comply with GDPR for data privacy and protection.

Types of Non-functional Requirements



Non-functional requirements implementation

- Non-functional requirements may affect the overall architecture of a system rather than the individual components.
    - For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- A single non-functional requirement, such as a security requirement, may generate several related functional requirements that define system services that are required.
    - It may also generate requirements that restrict existing requirements.

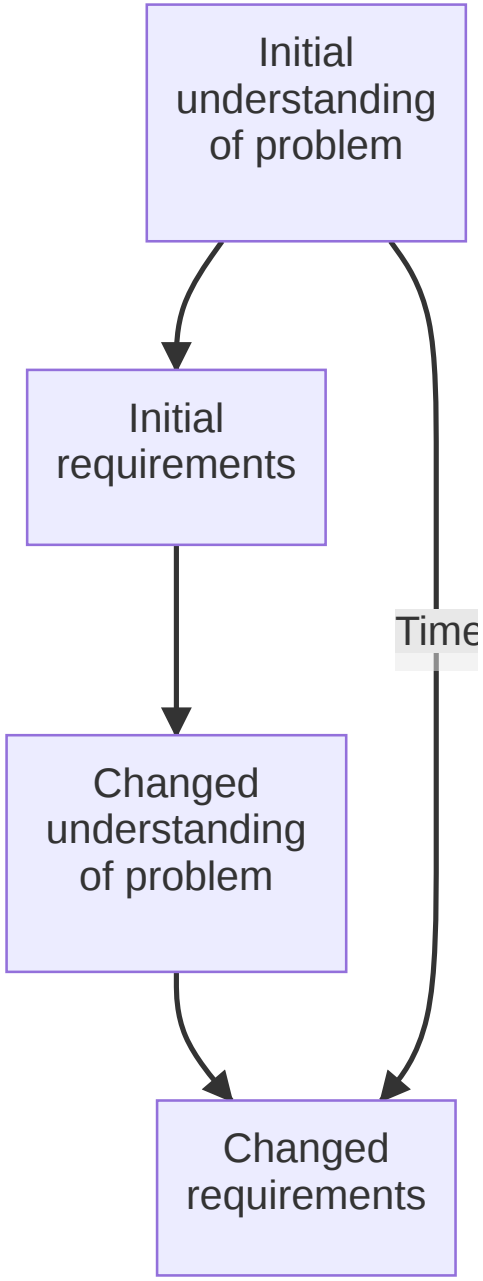
Non-functional Classifications

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc
- Organisational requirements
  - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Goals and Requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- Goal
  - A general intention of the user such as ease of use. They are usually quite general and vague (not easily measurable)
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested.
- Goals are helpful to developers as they convey the intentions of the system users.

Requirements' Evolution



Coursework

- Start working on the requirements of the system.