INTRODUCTION TO SOFTWARE ENGINEERING

Lesson 03 – Requirements Engineering

OUTLINE

1. Requirements elicitation
2. Writing and managing requirements
3. Use cases and user stories

1. REQUIREMENTS ELICITATION

According to IEEE standard 729, a requirement is defined as follows:

- A condition or capability needed by a user to solve a problem or achieve an objective
- A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed documents
- A documented representation of a condition or capability, as in 1 and 2.

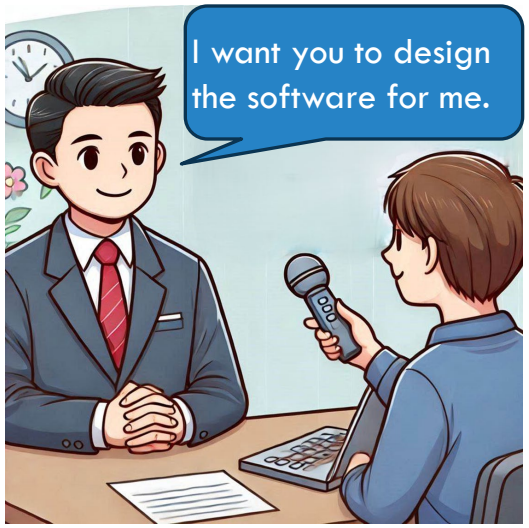
Actively gather detailed information from stakeholders about what the software needs to do. Methods for gathering requirements are:

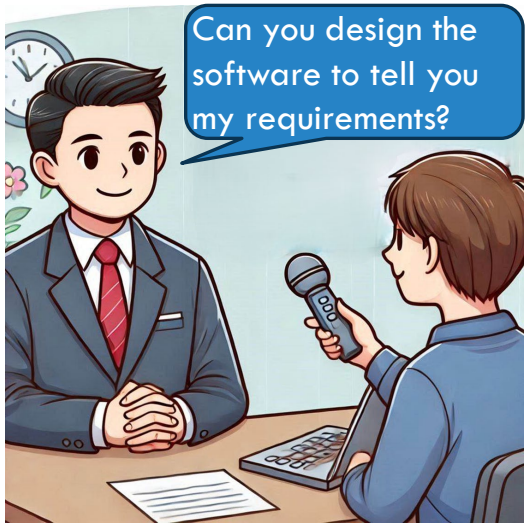
- Interviews
- Questionnaires
- User observation

1.1. IMPORTANCE OF REQUIREMENTS ELICITATION

- **Compliance with Business Objectives:** Comprehending the business context of a solution that adds value for the company.
- **User Satisfaction:** Higher user pleasure and acceptance of the finished product are the results of Requirements elicitation.
- **Time and Money Savings:** Having precise and well-defined specifications aids in preventing miscommunication and rework during the development phase.
- **Compliance and Regulation Requirements:** Requirements elicitation is crucial for projects in regulated industries to guarantee that the software conforms with applicable laws and norms. In industries like healthcare, finance, and aerospace, this is crucial.
- **Traceability and Documentation:** Throughout the software development process, traceability is based on well-documented requirements.

1.2. INTERVIEW QUESTIONS





1.2. INTERVIEW QUESTIONS

How requirements questions

- How will your stakeholders use this feature?
- Is this feature a process and, if so, what are the steps?
- What are the success criteria?

Where requirements questions

- Where does the process start?
- Where would the user be located physically when using this feature? Are they at home? In the office? Offsite?
- Where would the results be visible?

When requirements questions

- When will this feature be used?
- When will we be ready to start?
- When does this need to be completed by?

1.2. INTERVIEW QUESTIONS

Who requirements questions

- Who will use this feature?
- Who will deliver the inputs for the feature?
- Who will receive the outputs of the feature?
- Who will learn about the results of someone using this feature?
- Who can I ask to learn more about this?

What requirements questions

- What does this feature need to do? What is the end result of doing this?
- What are the pieces of this feature?
- What needs to happen next?
- What must happen before?
- What if….? Think of all the alternative scenarios and ask questions about what should happen if those scenarios are true.
- What needs to be tracked?
- What device will the stakeholder be using when they use this feature?

1.3. USER OBSERVATION

Observing users interacting with a product can be a great way to understand the product usability and user experience.

Conducting observations is relatively easy as it doesn't require a huge amount of training, and it can be relatively fast.

Two ways of observations:

- **Controlled observation** (instructs, give examples, and then observes how users use it)
- **Naturalistic observation** (observes user's daily use of a product)

2. WRITING AND MANAGING REQUIREMENTS

There are 3 types of requirements:

- Functional requirements
- Non-functional requirements
- Domain requirements

2.1. FUNCTIONAL REQUIREMENTS

Functional Requirements are the requirements that the end user specifically demands as basic facilities that the system should offer.

It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality that defines what function a system is likely to perform.

- The requirements stated by the user, one can see directly in the final product, unlike the non-functional requirements. For example, in a hospital management system, a doctor should be able to retrieve the information of his patients.
- Each high-level functional requirement may involve several interactions or dialogues between the system and the outside world.
- To accurately describe the functional requirements, all scenarios must be enumerated.

2.2. NON-FUNCTIONAL REQUIREMENTS

These are basically the quality constraints that the system must satisfy according to the project contract.

Nonfunctional requirements, not related to the system functionality, rather define how the system should perform The priority or extent to which these factors are implemented varies from one project to other. It includes:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

2.3. DOMAIN REQUIREMENTS

Domain requirements are the requirements that are characteristic of a particular category or domain of projects.

Domain requirements can be functional or nonfunctional.

Domain requirements engineering is a continuous process of proactively defining the requirements for all foreseeable applications to be developed in the software product line.

3. USE CASES AND USER STORIES

Use cases and user stories are both tools used in software development to plan and explain a product's requirements and features:

Use case

A description of how a user interacts with a system to achieve a specific goal. Use cases can help identify requirements, guide development, and ensure stakeholders understand what needs to be built.

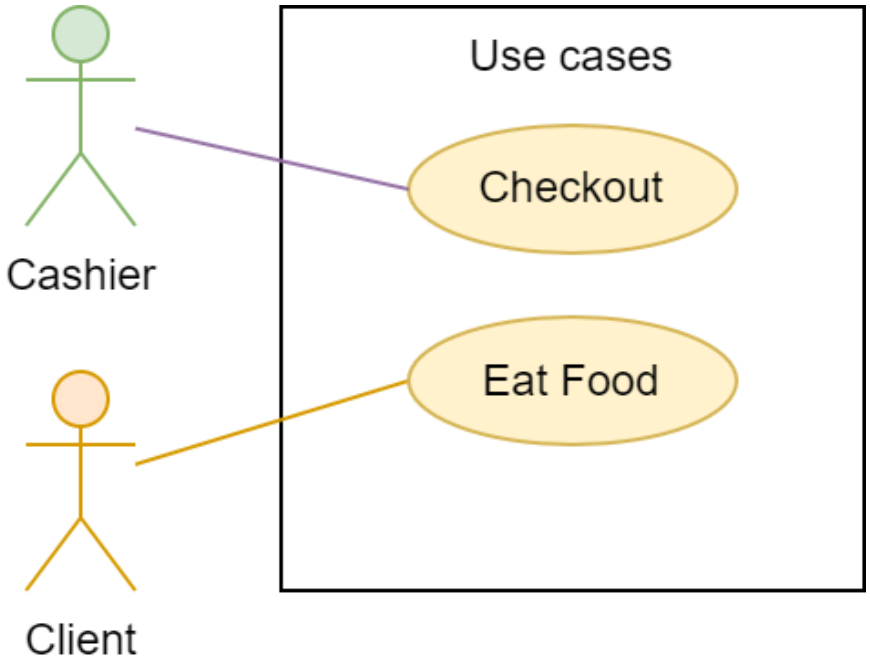
User stories

Focus on the user's needs & goals, and are written as simple narratives that capture what the user wants to do. They are often used in short-term, Agile projects.

3.1. USE CASES

Cashier can Checkout or get payment from client.

Client can Eat Food they have ordered.



3.2. USER STORIES

User stories are typically written using the following format: "As a [type of user], I want [an action] so that [a benefit]".

- [type of user]: a role or a group of users. Ex: client, cashier, designer, ...
- [an action]: a thing or feature that the system should have. Ex: a color palette tool, ...
- [a benefit]: reason of having the action. Ex: apply consistence colors, ...

Examples:

- "As a designer named John Doe, I want a color palette tool in my design software so that I can easily apply consistent colors to my website designs."
- "As an admin, I want to change my account's password so that I can add extra protection to my account."