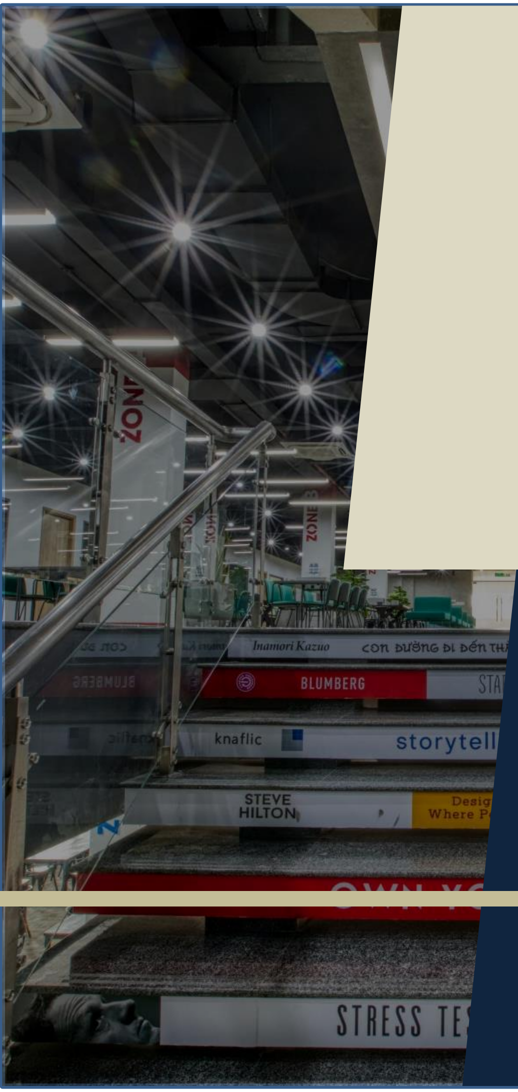


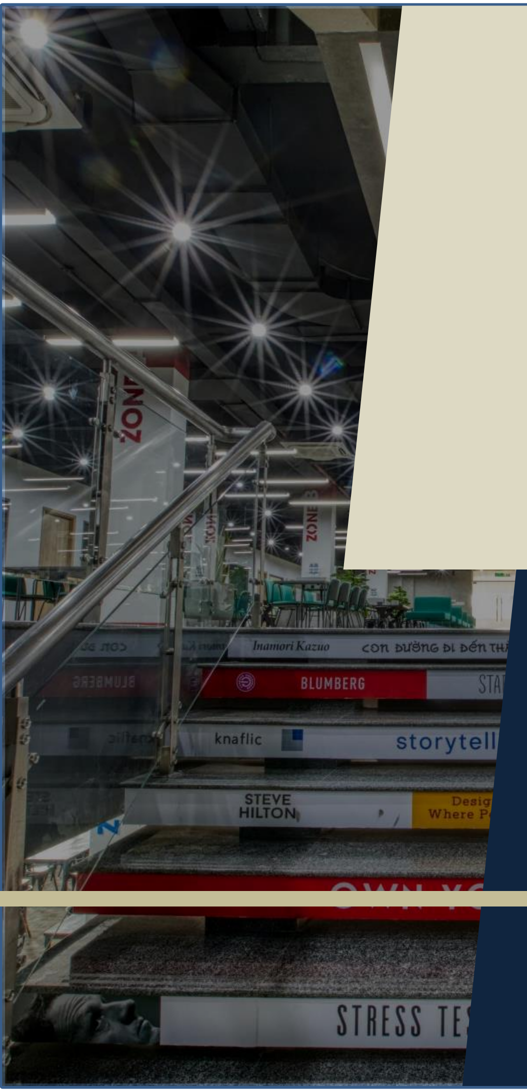
EIU
TRƯỜNG ĐẠI HỌC
QUỐC TẾ
MIỀN ĐÔNG
EASTERN
INTERNATIONAL
UNIVERSITY

Chapter 4. Requirements Engineering

4.1. Functional and Non-Functional Requirements

4.2. Requirements Engineering Processes

4.3. Requirements elicitation and analysis



EIU
TRƯỜNG ĐẠI HỌC
QUỐC TẾ
MIỀN ĐÔNG
EASTERN
INTERNATIONAL
UNIVERSITY

Chapter 4. Requirements Engineering

4.4. Requirements specification

4.5. Requirements validation

4.6. Requirements change

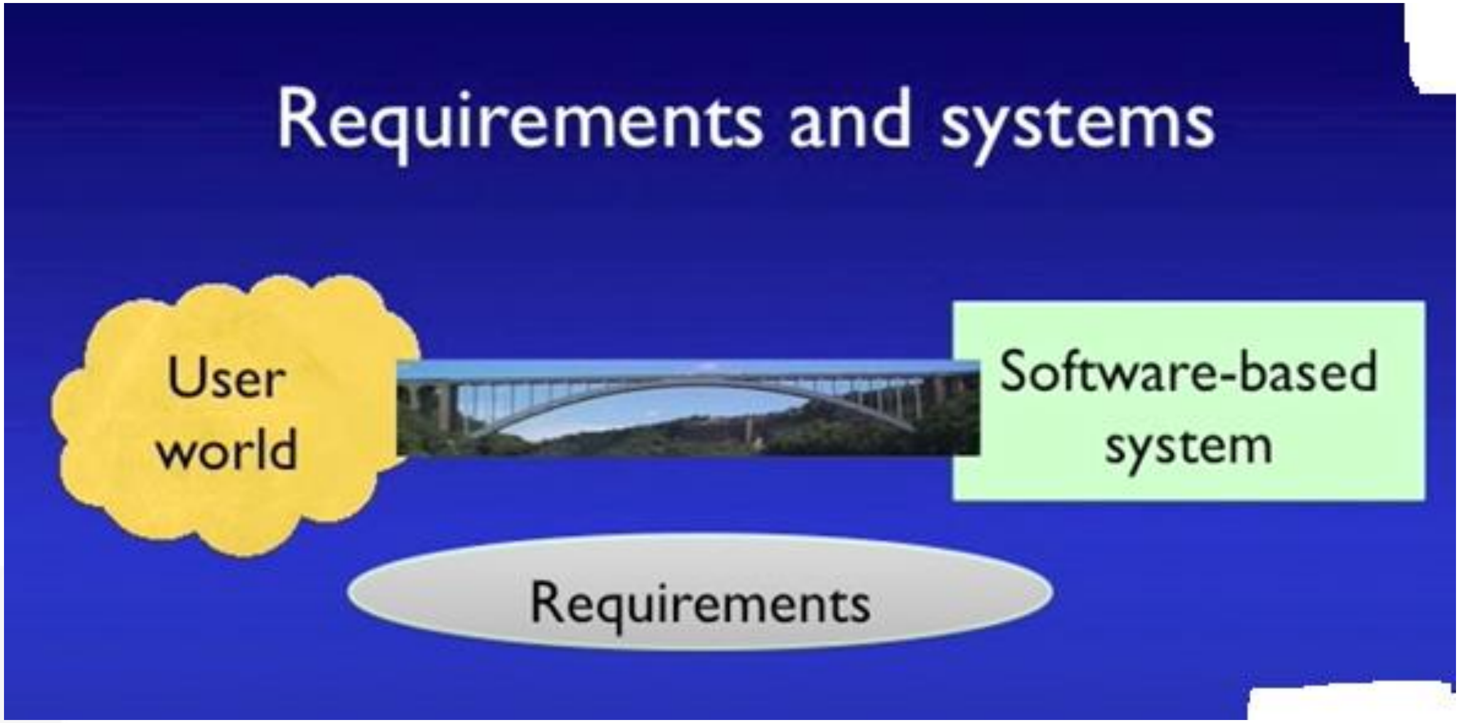
Why we do need Requirements?

- Ask somebody learning how to program what the most difficult part of building software is, they'll tell you coding, or testing if they have a little bit more experience
- But, ask a senior software engineer and they'll tell you it's getting the problem right in the first place: Requirements
- Most problems of software failure come from Requirements Specification

Reasons behind problems in Requirements

- ➢ Software are intangible, those can not be touched
- ➢ Stakeholders don't have clear ideas about what they need
  - ❑ It is very difficult to imagine how future systems might work
  - ❑ They can describe their idea in a vague and ambiguous way
- ➢ Business operates in a rapidly changing environment
  - ❑ Requirements appropriate in one month, may be inappropriate in next month
- ➢ Multiple stakeholders with different goals and priorities
  - ❑ They do not agree all time
  - ❑ Consequently, the requirements are compromised between two groups of stakeholders

Requirements and System



- Requirements are the bridge between the real world and the software system

Software system requirements

- Requirements are defined early on the software process, after an analysis of the users’ problems and the environment in which the system will be used

- This chapter present a ‘traditional’ view of requirements rather than requirements in agile processes. For most large systems, it is still the case that there is a clearly identifiable requirements engineering phase before the implementation of the system begins

Software system requirements

Developers

Users

- **User requirements:**
- written in natural language
- written for the users of the system,
- **System requirement:**
- written in a more technical language.
- written for the developers of the system.

Software system requirements

Users

Developers

User requirement:	System requirement:
The system must allow users to search for and purchase products.	• The system must provide a product search page where users can enter their search criteria and view a list of matching products. • The system must also provide a product detail page where users can view more information about a specific product and add it to their shopping cart. • Once the user has added all of the desired products to their shopping cart, the system must

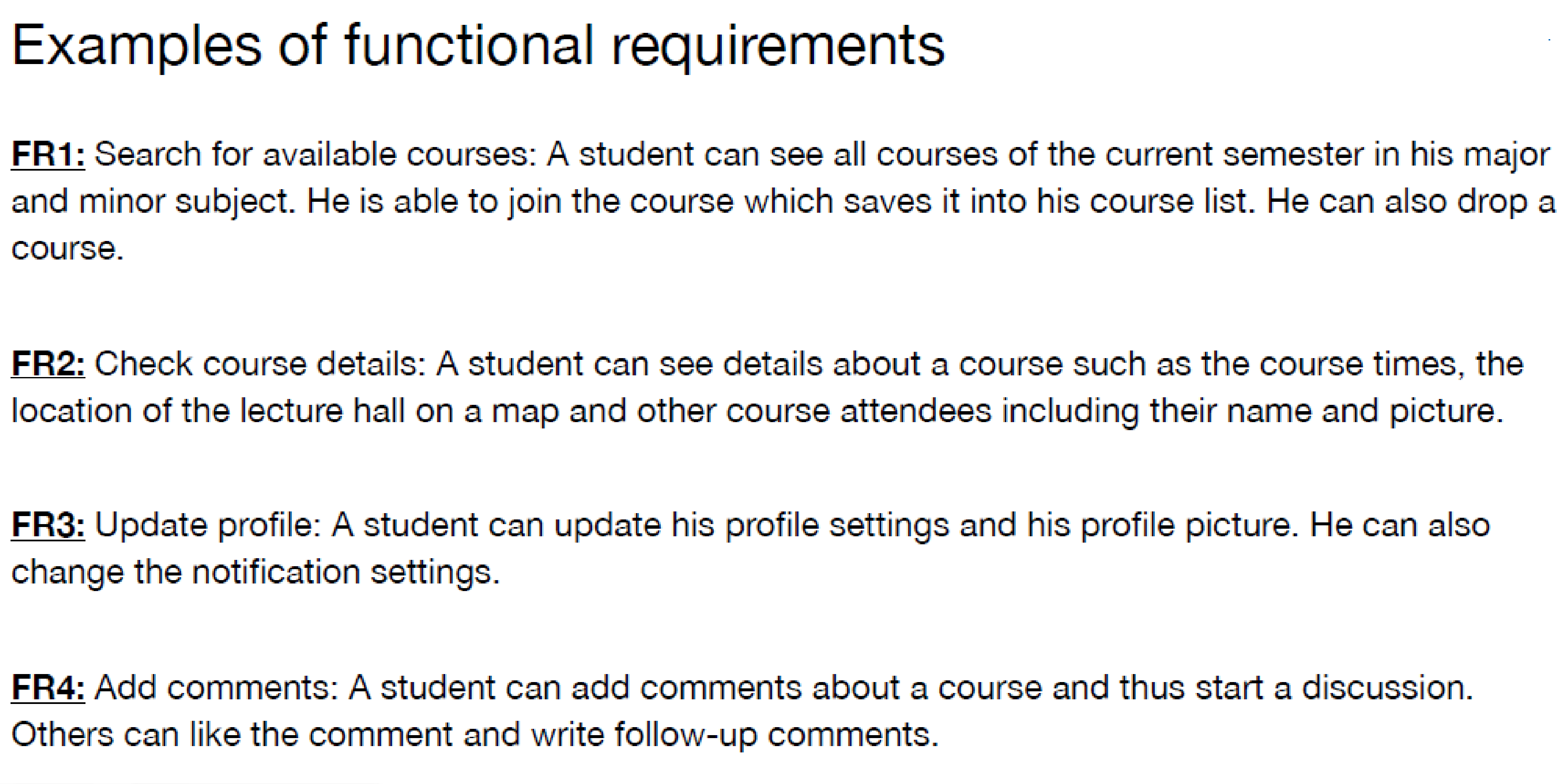
Functional and Non-Functional Requirements

Software system requirements are often classified as **functional** requirements or **nonfunctional** requirements

- **Functional requirements**: These are statements of services the system should provide, how the system should react to particular inputs
- **Non-functional requirements**: These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, and constraints imposed by standards.

In reality, the distinction between different types of requirement is not as clear-cut as these simple definitions suggest.

Example of Functional Requirements



Examples of functional requirements

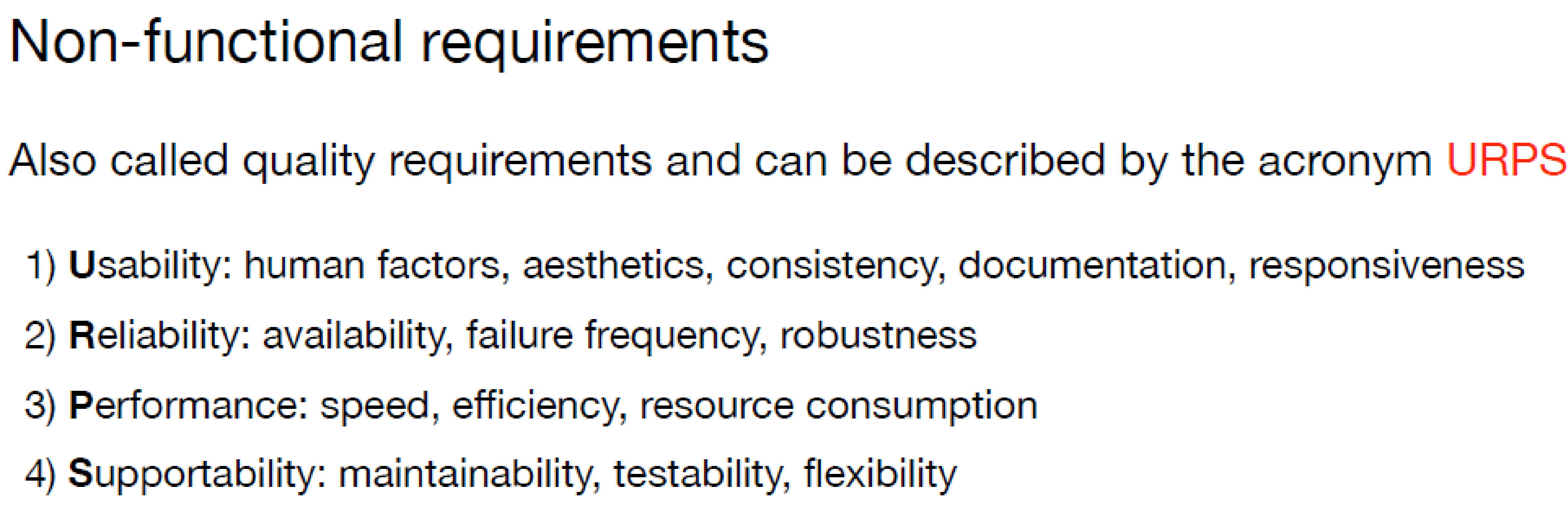
FR1: Search for available courses: A student can see all courses of the current semester in his major and minor subject. He is able to join the course which saves it into his course list. He can also drop a course.

FR2: Check course details: A student can see details about a course such as the course times, the location of the lecture hall on a map and other course attendees including their name and picture.

FR3: Update profile: A student can update his profile settings and his profile picture. He can also change the notification settings.

FR4: Add comments: A student can add comments about a course and thus start a discussion. Others can like the comment and write follow-up comments.

Non-Functional Requirements

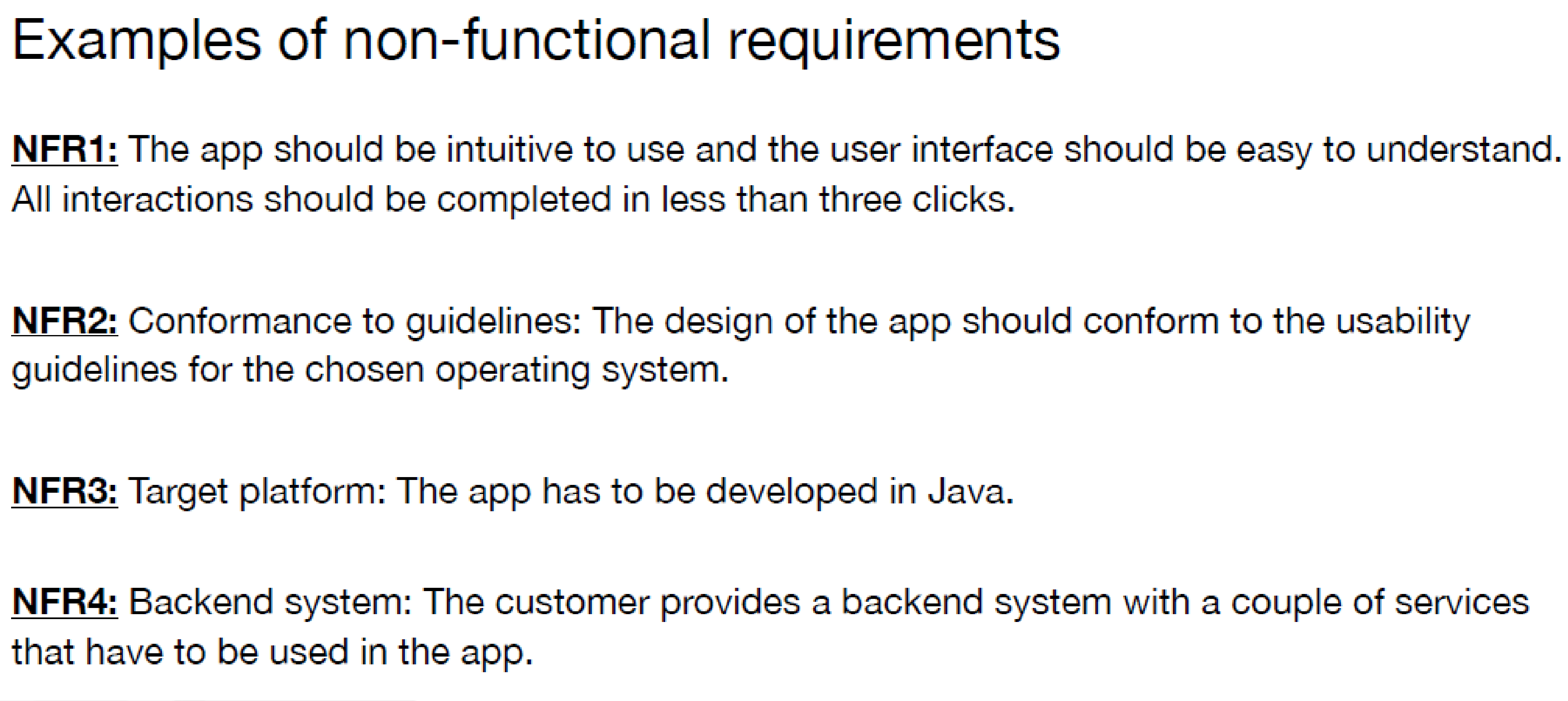


Non-functional requirements

Also called quality requirements and can be described by the acronym URPS

1) **U**sability: human factors, aesthetics, consistency, documentation, responsiveness
2) **R**eliability: availability, failure frequency, robustness
3) **P**erformance: speed, efficiency, resource consumption
4) **S**upportability: maintainability, testability, flexibility

Non-Functional Requirements



Examples of non-functional requirements

NFR1: The app should be intuitive to use and the user interface should be easy to understand. All interactions should be completed in less than three clicks.

NFR2: Conformance to guidelines: The design of the app should conform to the usability guidelines for the chosen operating system.

NFR3: Target platform: The app has to be developed in Java.

NFR4: Backend system: The customer provides a backend system with a couple of services that have to be used in the app.

Questions?

What are non-functional requirements?

a) Requirements that address how the system should operate.

b) Requirements that do not work.

c) Requirements that specify what the system should do.

Questions?

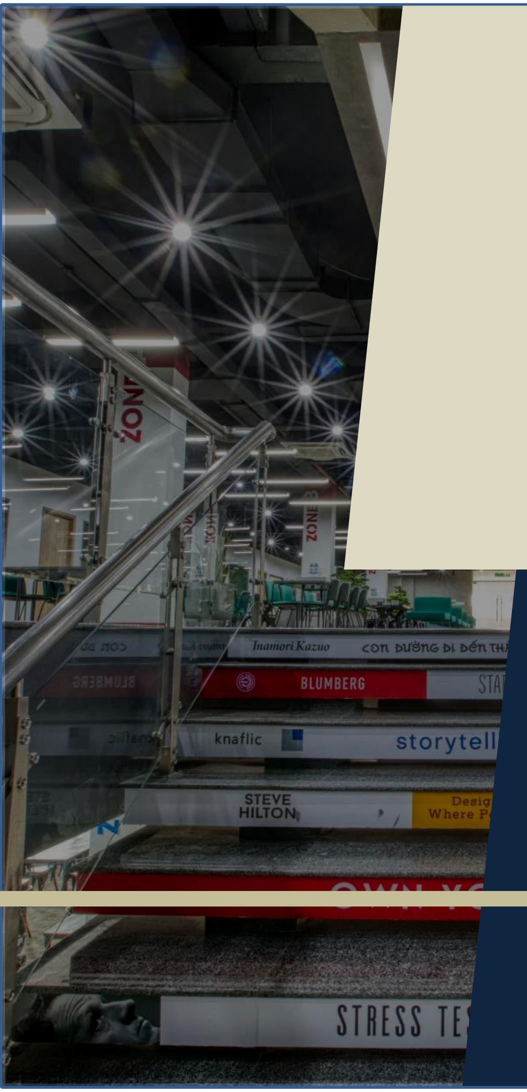
Which of the following are non-functional requirements:

a) Some product requirements, like using a specific encryption protocol, are non-functional requirements.

b) Organization requirements imposed by the company, like a specific coding style, are non-functional requirements.

c) External requirements imposed by external organization, like using a specific development style, are non-functional requirements.

d) All of the above.

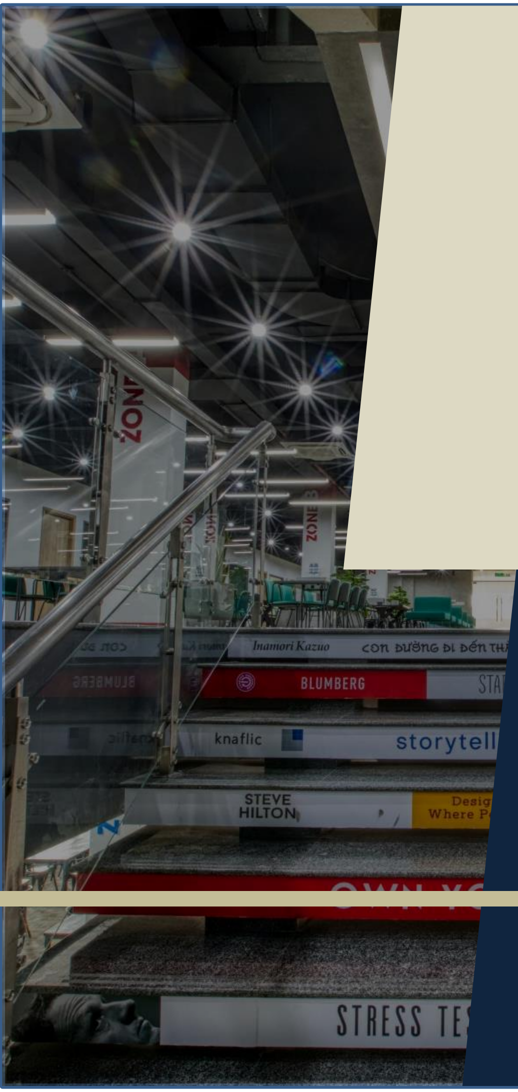


EIU

TRƯỜNG ĐẠI HỌC
QUỐC TẾ
MIỀN ĐÔNG

EASTERN
INTERNATIONAL
UNIVERSITY

Software Engineering
Course's Code: CSE 305

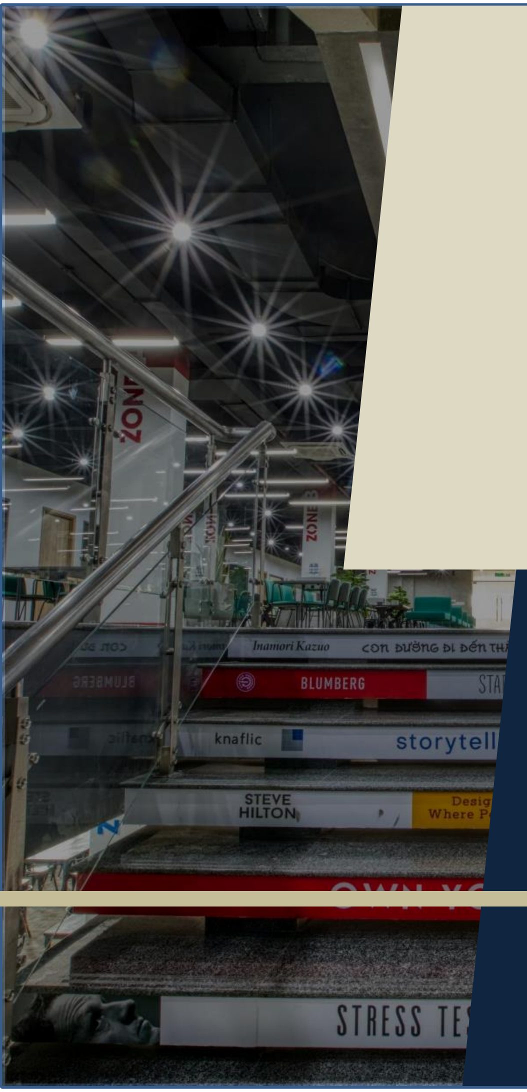


Chapter 4. Requirements Engineering

4.1. Functional and Non-Functional Requirements

4.2. Requirements Engineering Processes

4.3. Requirements elicitation and analysis



EIU
TRƯỜNG ĐẠI HỌC
QUỐC TẾ
MIỀN ĐÔNG
EASTERN
INTERNATIONAL
UNIVERSITY

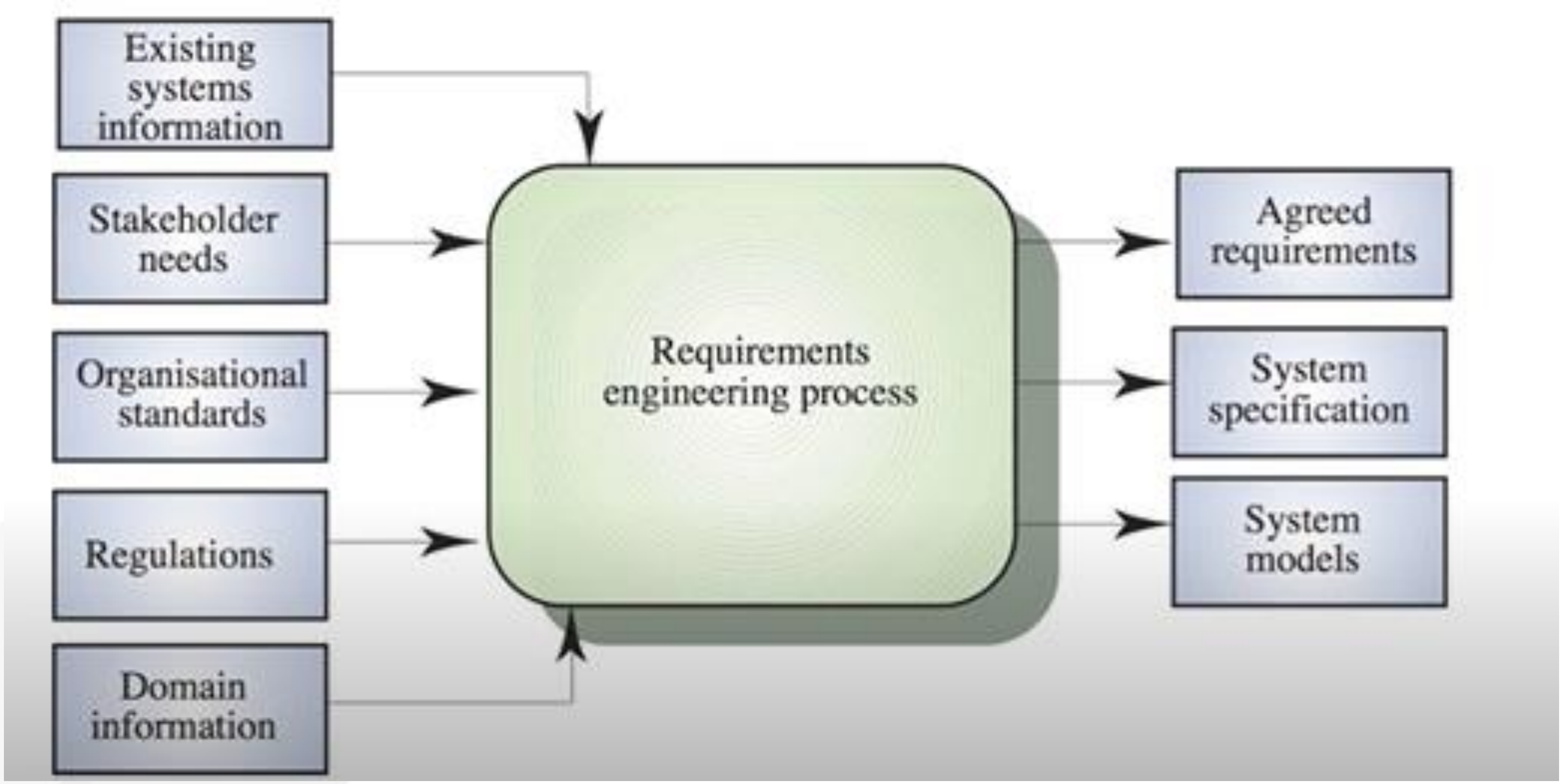
Chapter 4. Requirements Engineering

4.4. Requirements specification

4.5. Requirements validation

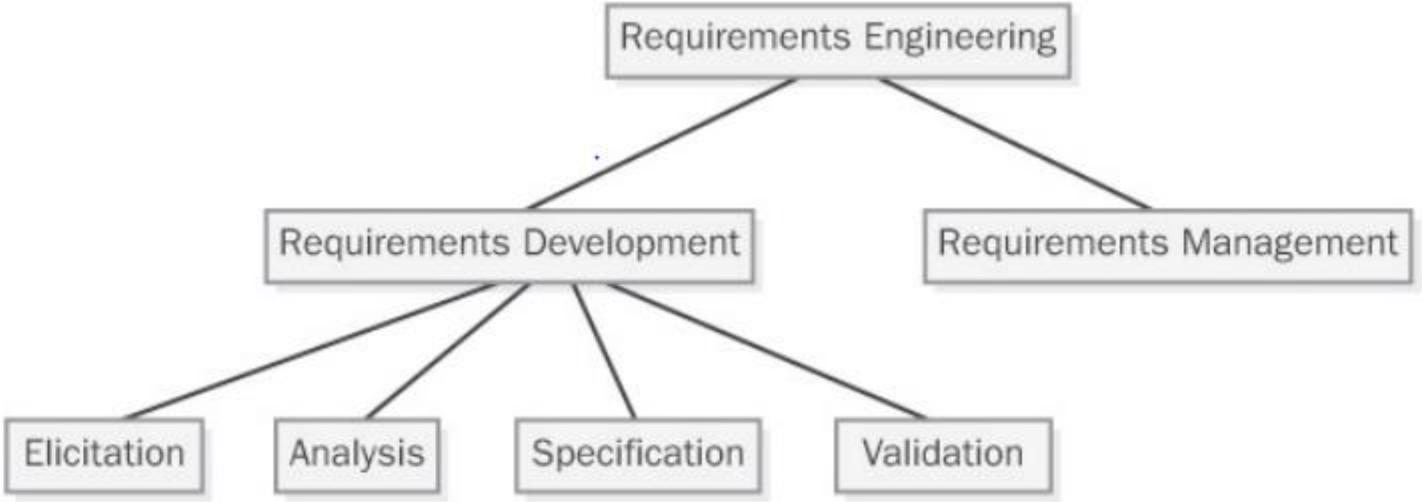
4.6. Requirements change

Requirements Engineering Process



Requirements Engineering Process

Requirements Engineering



Requirements Engineering

Requirements Development

Requirements Management

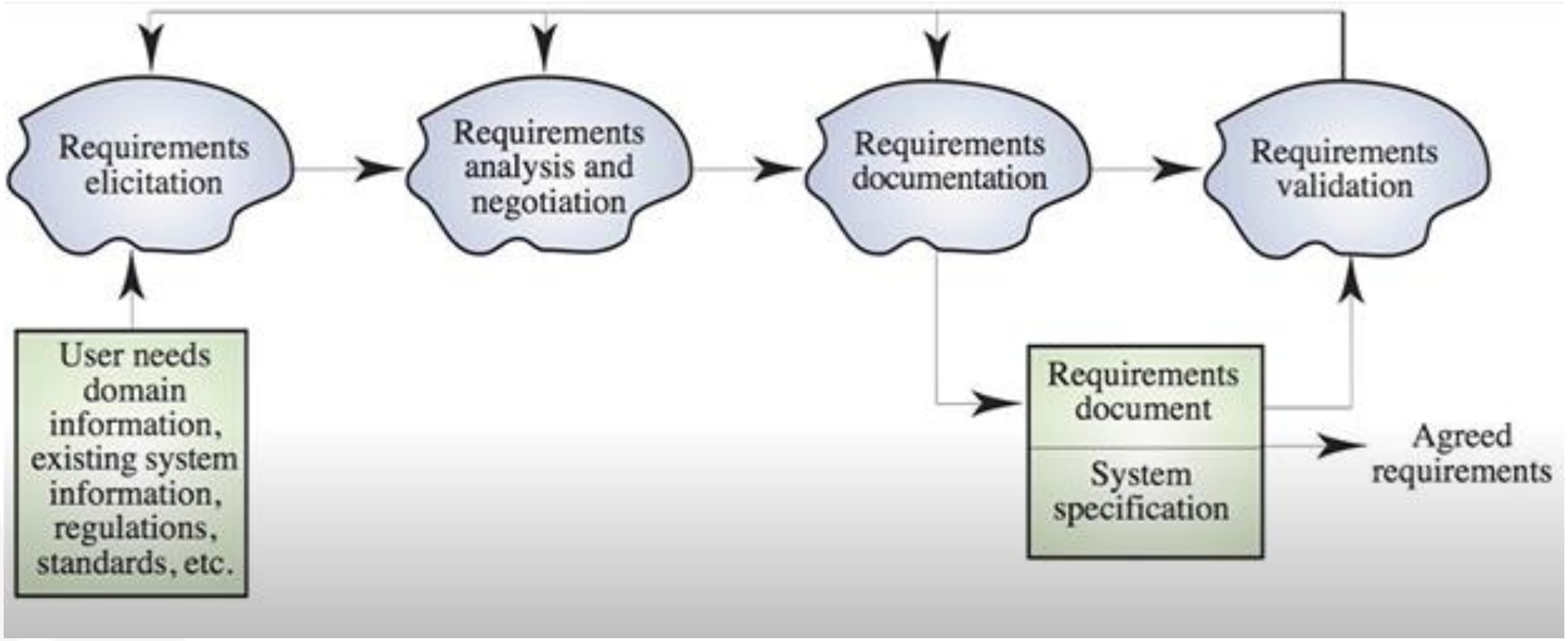
Elicitation

Analysis

Specification

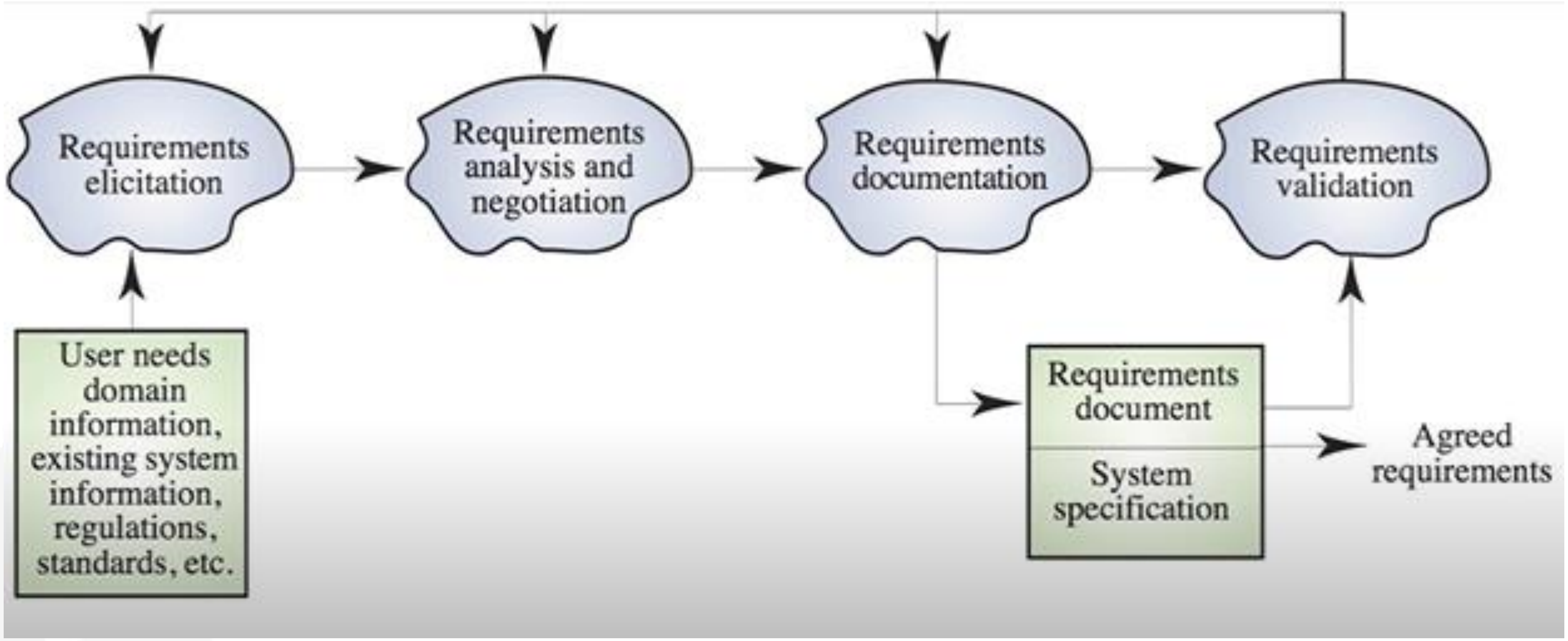
Validation

Requirements Engineering Process


Requirements elicitation
Requirements analysis and negotiation
Requirements documentation
Requirements validation
User needs domain information, existing system information, regulations, standards, etc.
Requirements document
System specification
Agreed requirements

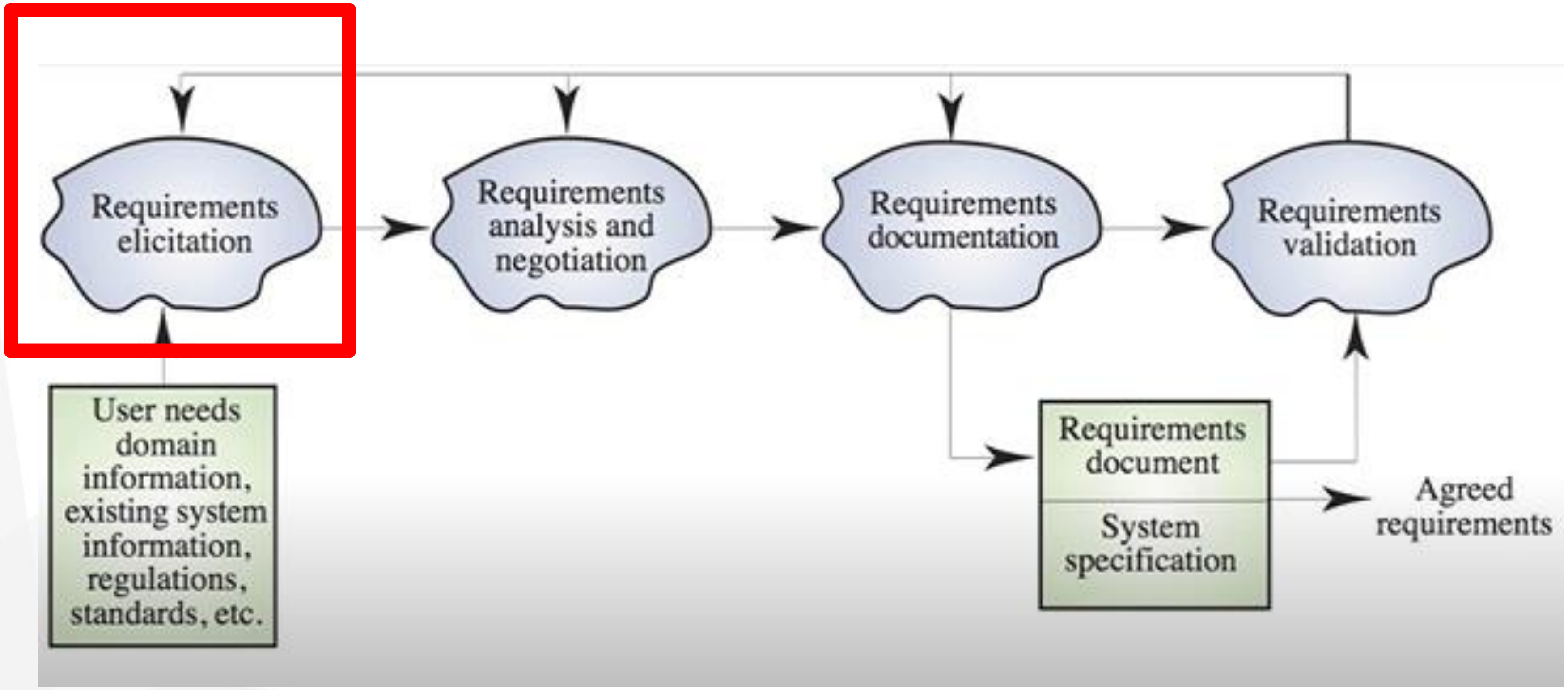
- ➢ **Requirements elicitation**: This is the first step, where we are trying to discover what are the requirements
- ➢ **Requirements analysis and negotiation**: The collected information are analyzed looking for inconsistencies and emissions
    - ❖ There is a constant back and forward between elicitation and analysis activity to develop and refine the requirements

Requirements Engineering Process


Requirements elicitation
Requirements analysis and negotiation
Requirements documentation
Requirements validation
User needs domain information, existing system information, regulations, standards, etc.
Requirements document
System specification
Agreed requirements

- **Requirements documentation**: We have to write the requirements in a such a way that those can be understandable to both the system stakeholders and the system engineers
- **Requirements Validation**: Requirements back to the stakeholder to see that what the stakeholder really wants?

Requirements Engineering Process

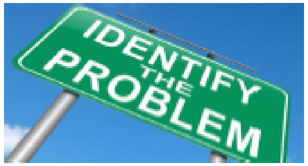

Requirements elicitation
Requirements analysis and negotiation
Requirements documentation
Requirements validation
User needs domain information, existing system information, regulations, standards, etc.
Requirements document
System specification
Agreed requirements

Requirements Elicitation

Purpose of Requirements Elicitation

Requirements elicitation focuses on describing the **purpose** of the system.

The client, the developers, and the users **identify a problem area** and **define a system** that addresses the problem.



Such a definition is called a **requirements specification** and serves as a contract between the client and the developers.

The **requirements specification** is structured and formalized during **analysis.**

Requirements Elicitation

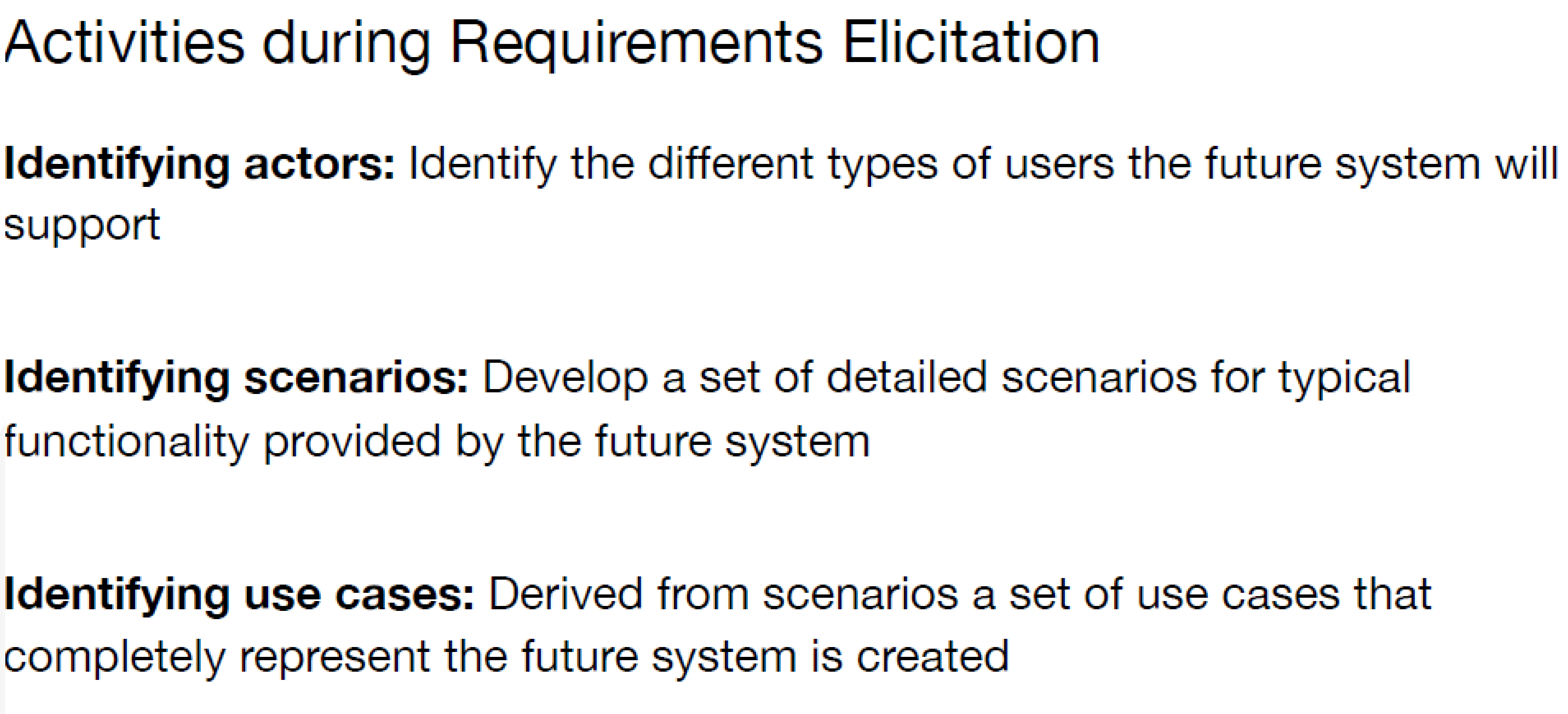
- Techniques:

  - ❑ **Interviewing**: where you talk to people about what they do.
  - ❑ **Observation**: Where you watch people doing their job to see what artifacts they use, how they use them, and so on.
  - ❑ **Analysis of Existing System**
  - ❑ **Brainstorming**: To invent new way of doing things or when much is unknown
  - ❑ **Scenario/ user story**
  - ❑ **Workshop**

Requirements Elicitation

- Problems of Requirement Elicitation
  - ❑ **Problems of scope**: The boundary of system is ill-defined. Or unnecessary details are provided.
  - ❑ **Problems of understanding**: The users are not sure of what they need, and don’t have full understanding of the prob lem domain.
  - ❑ **Problems of volatility**: the requirements change overtime.

Requirements Elicitation



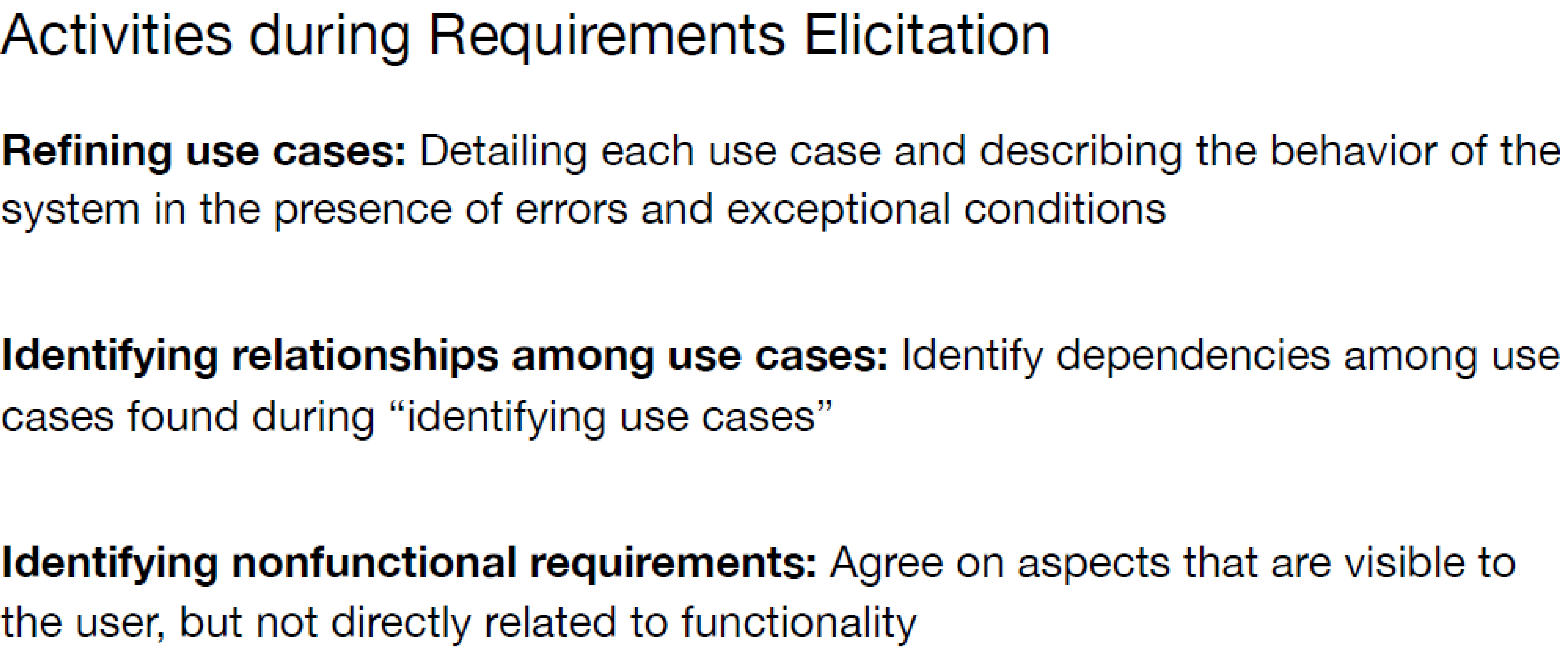
Activities during Requirements Elicitation

Identifying actors: Identify the different types of users the future system will support

Identifying scenarios: Develop a set of detailed scenarios for typical functionality provided by the future system

Identifying use cases: Derived from scenarios a set of use cases that completely represent the future system is created

Requirements Elicitation



Activities during Requirements Elicitation

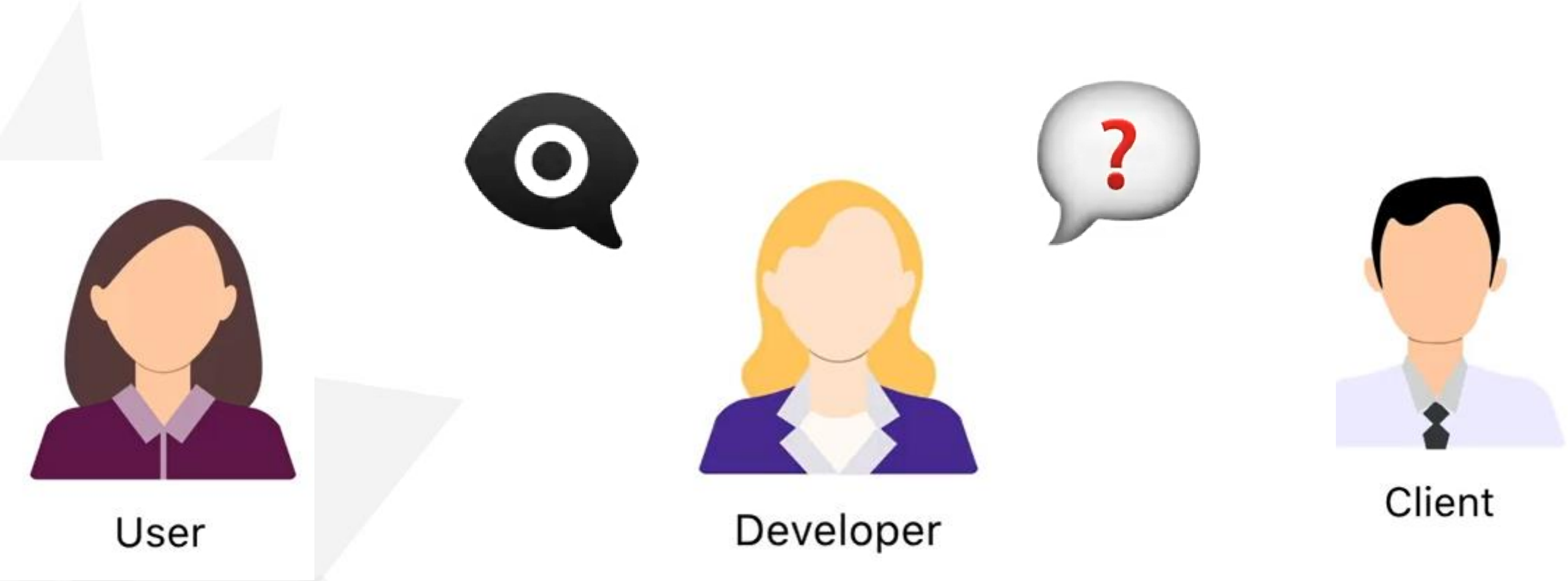
Refining use cases: Detailing each use case and describing the behavior of the system in the presence of errors and exceptional conditions

Identifying relationships among use cases: Identify dependencies among use cases found during “identifying use cases”

Identifying nonfunctional requirements: Agree on aspects that are visible to the user, but not directly related to functionality

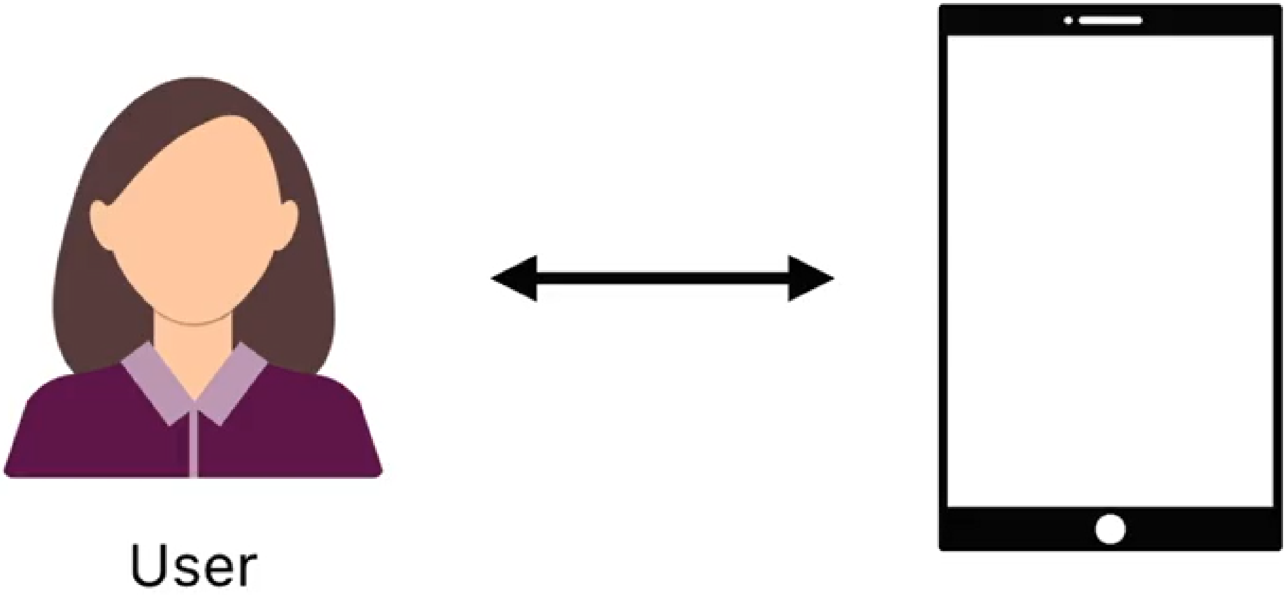
Requirements Elicitation

- **Scenario** enhances the requirements elicitation by providing a communication tool that can be understood by the user and client
- You as a software engineer **can identify the scenario by observing the user over asking the questions to the clients**



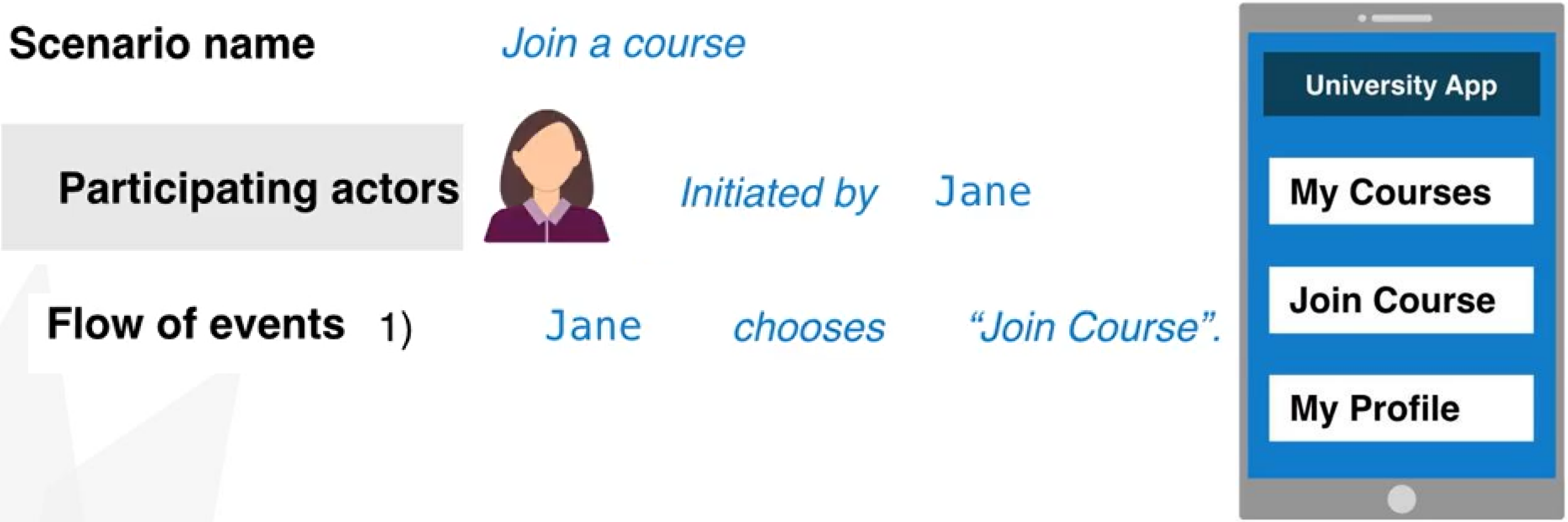
Requirements Elicitation

- **Scenario** describes the actions between users and the systems
- **It represents the concrete and informal description** of the functionalities of a system from the viewpoint of an actor



Requirements Elicitation

- ➢ Scenario has the following structure:



Scenario name *Join a course*

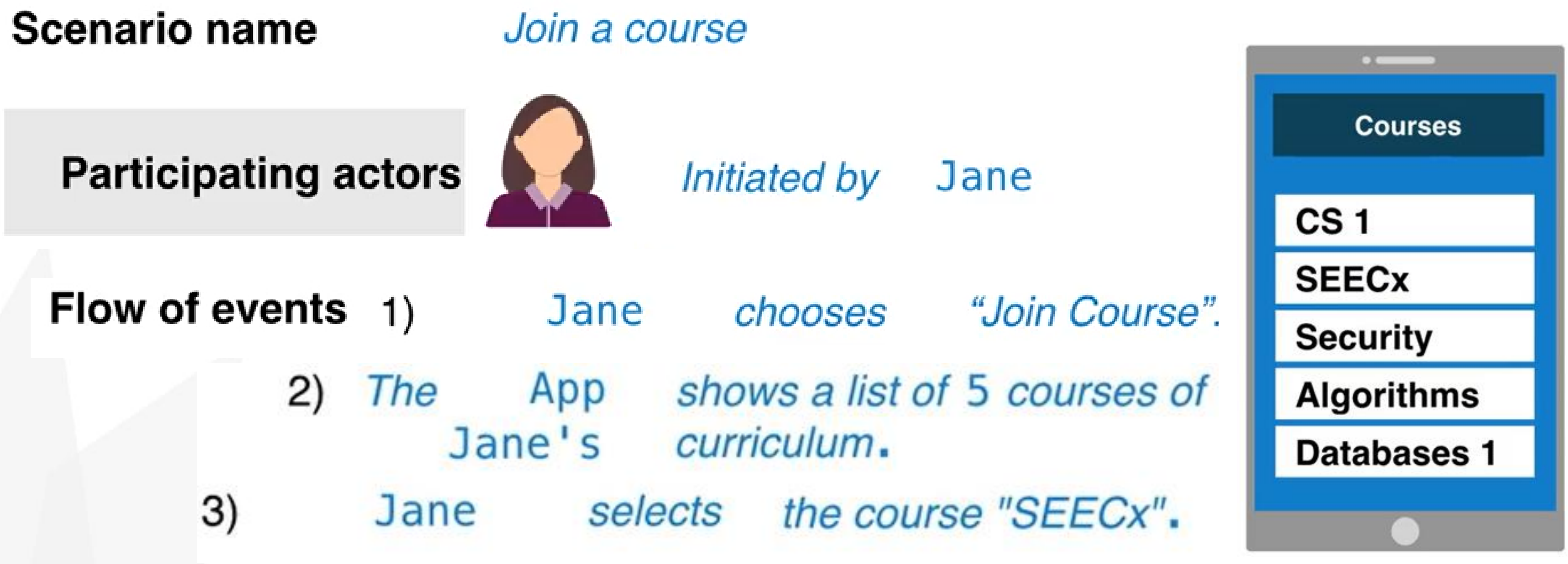
Participating actors *Initiated by* Jane

Flow of events 1) Jane *chooses* *“Join Course”.*

➢ **Flow of events** represent interactions step by step. They can be divided **as actor steps and system steps**.

Requirements Elicitation

- Scenario has the following structure:



Scenario name *Join a course*

Participating actors *Initiated by* Jane

Flow of events

1) Jane *chooses* *"Join Course".*
2) *The* App *shows a list of 5 courses of* Jane's *curriculum.*
3) Jane *selects* *the course "SEECx".*

Courses

CS 1

SEECx

Security

Algorithms

Databases 1

- Flow of events represent interactions step by step. They can be divided as actor steps and system steps.

Requirements Elicitation

- Scenario has the following structure:

Scenario name *Join a course*

Participating actors

Initiated by Jane

Flow of events

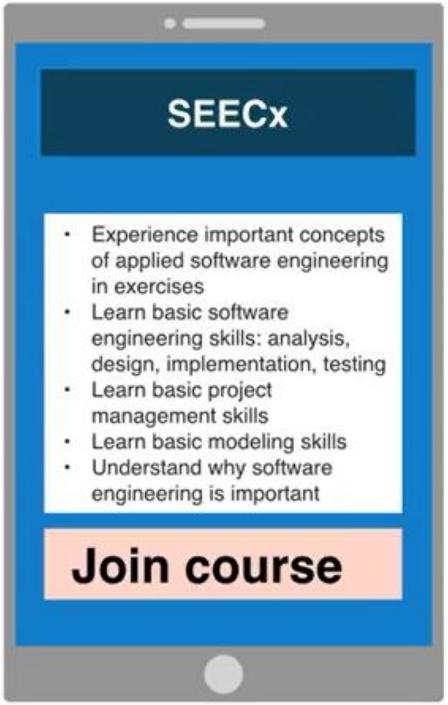
1) Jane *chooses* *“Join Course”.*

2) *The* App *shows a list of* 5 *courses of* Jane's *curriculum.*

3) Jane *selects* *the course "SEECx".*

4) *The* App *shows* SEECx's *details as a popup window.*

5) Jane *chooses* *“Join Course”.*



SEECx

- Experience important concepts of applied software engineering in exercises
- Learn basic software engineering skills: analysis, design, implementation, testing
- Learn basic project management skills
- Learn basic modeling skills
- Understand why software engineering is important

Requirements Elicitation

- ➢ Scenario has the following structure:

Scenario name *Join a course*

Participating actors

Initiated by Jane

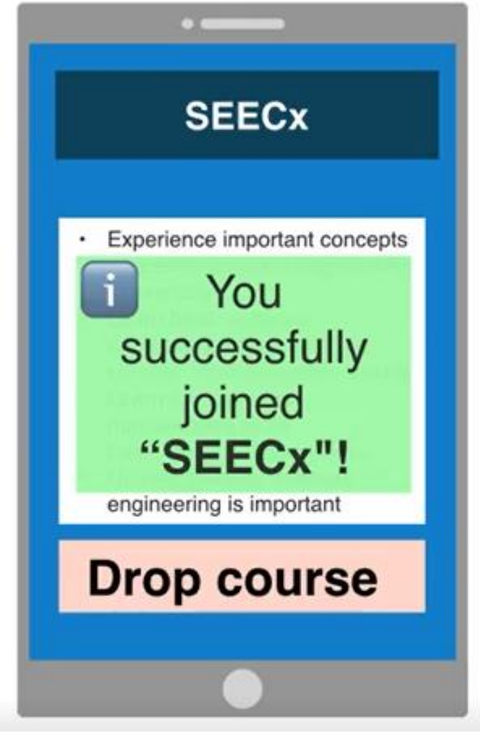
Flow of events 1) Jane *chooses* *"Join Course"*

2) *The* App *shows a list of* 5 *courses of* Jane's *curriculum.*

3) Jane *selects* *the course "SEECx".*

4) *The* App *shows* SEECx's *details as a popup window.*

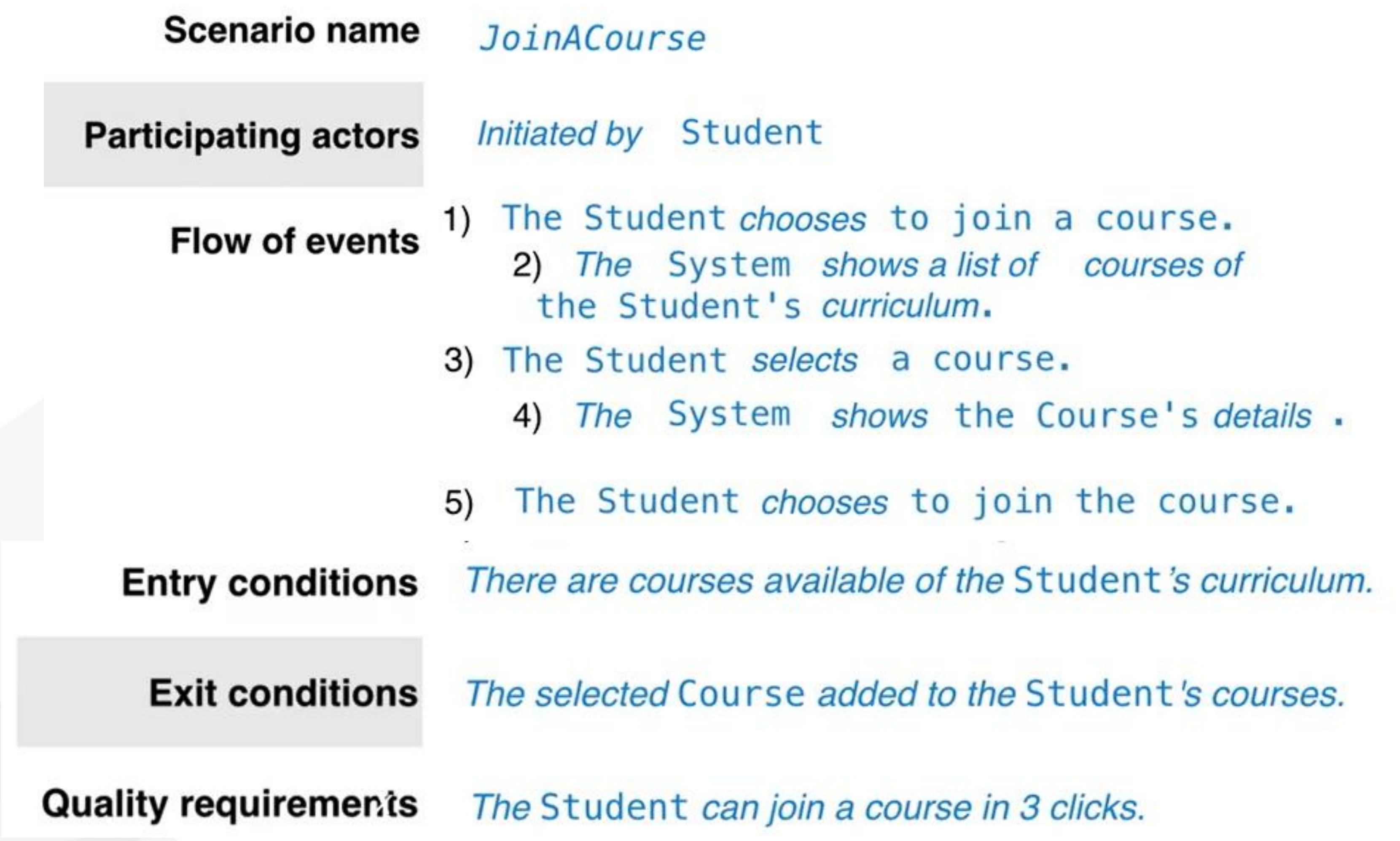
5) Jane *chooses* *"Join Course".*



You successfully joined "SEECx"!

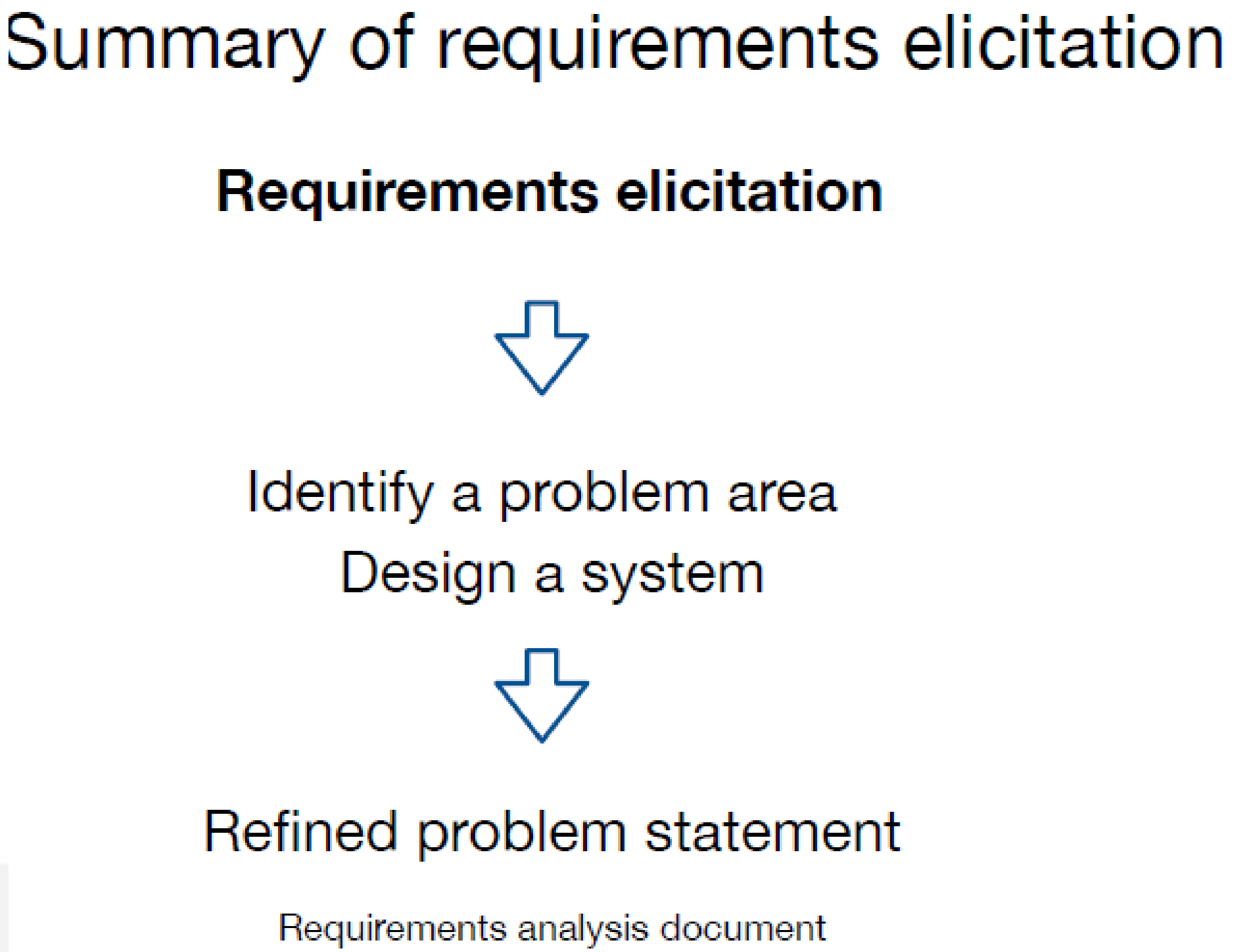
Requirements Elicitation

- From this scenario use cases have been generated
- **Use cases** describes all possible cases of one functionality



Scenario name	*JoinACourse*
Participating actors	*Initiated by* Student
Flow of events	1) The Student *chooses* to join a course. 2) *The* System *shows a list of courses of* the Student's *curriculum*. 3) The Student *selects* a course. 4) *The* System *shows* the Course's *details* . 5) The Student *chooses* to join the course.
Entry conditions	*There are courses available of the* Student*'s curriculum.*
Exit conditions	*The selected* Course *added to the* Student*'s courses.*
Quality requirements	*The* Student *can join a course in 3 clicks.*

Requirements Elicitation



Summary of requirements elicitation

Requirements elicitation

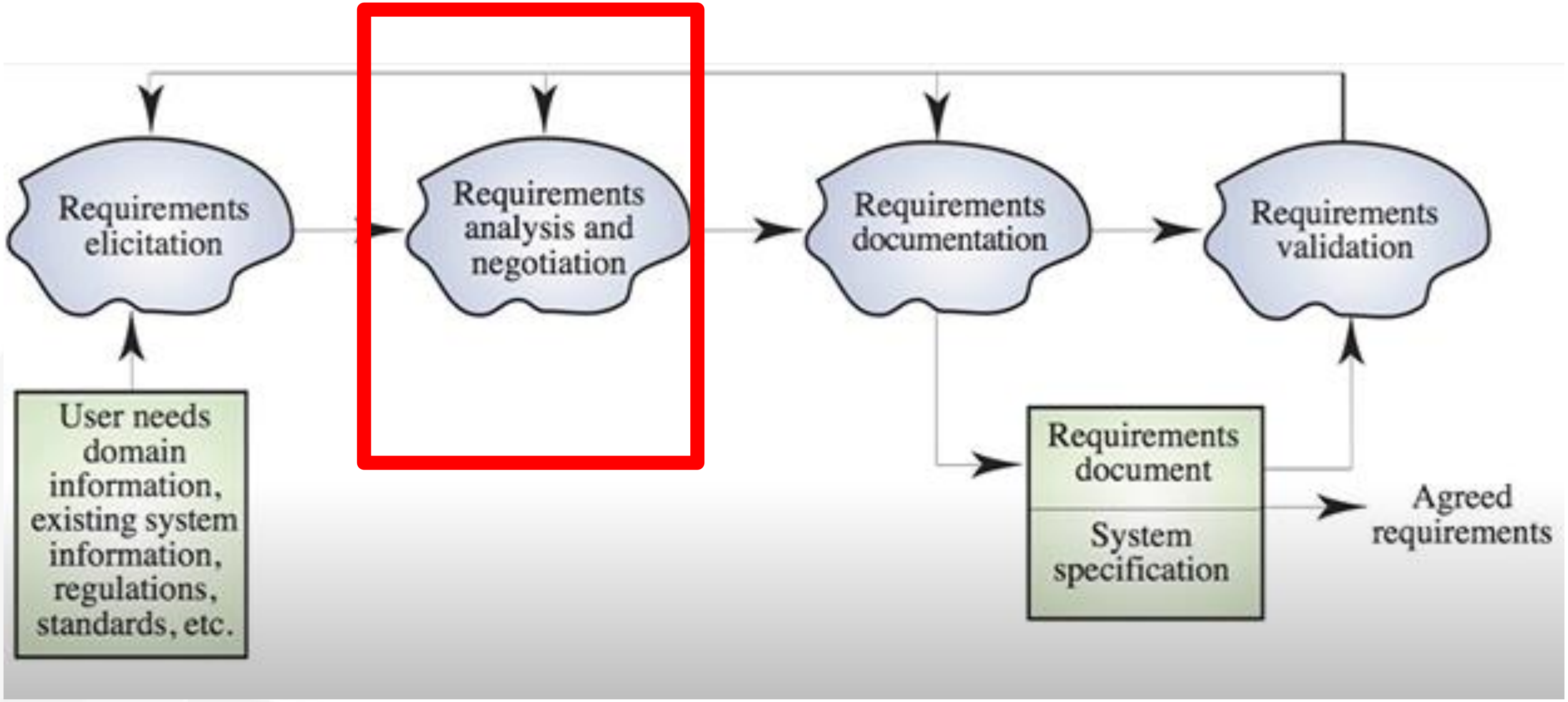
Identify a problem area

Design a system

Refined problem statement

Requirements analysis document

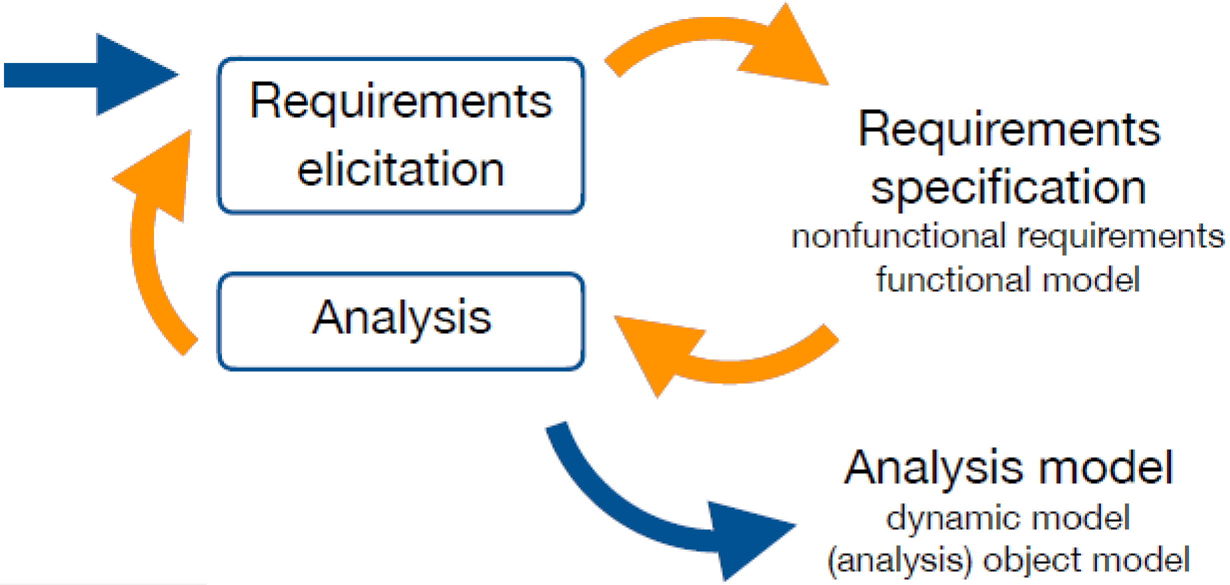
Requirement Engineering Process


Requirements elicitation
Requirements analysis and negotiation
Requirements documentation
Requirements validation
User needs domain information, existing system information, regulations, standards, etc.
Requirements document
System specification
Agreed requirements

Requirements Analysis

Purpose of Analysis

Analysis focuses on producing a model of the system, called the *analysis model*, which is *correct, complete, consistent, and verifiable*.



Requirements Analysis

- ➢ Main purpose of requirements analysis:
  - ❑ Clearly understand the user requirements,
  - ❑ Detect **inconsistencies**, **ambiguities**, and **incompleteness**.
- ➢ Incompleteness and inconsistencies:
  - ❑ Resolved through further discussions with the end-users and the customers.

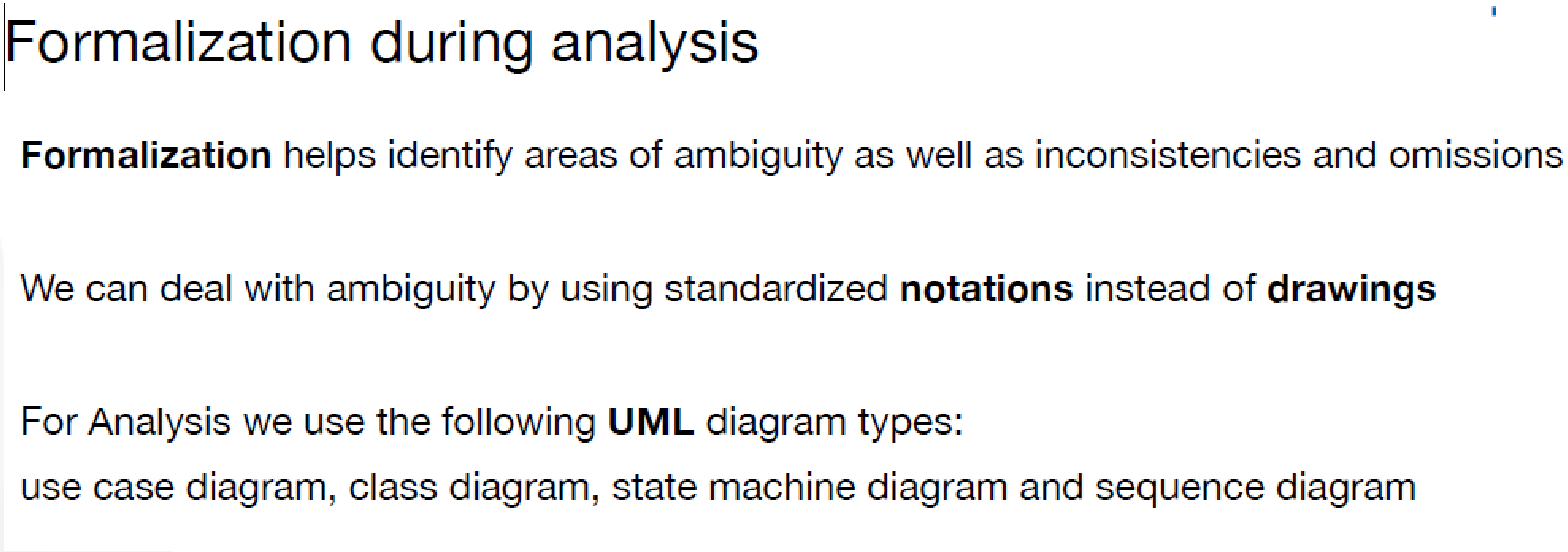
Requirements Analysis

- **Inconsistent Requirements**
  - ❑ Some part of the requirements contradicts with some other part.
- **Example**:
  - ❑ One customer says turn off heater and open water shower when temperature > 100 C
  - ❑ Another customer says turn off heater and turn ON cooler when temperature > 100 C

Requirements Analysis

- **Incomplete Requirements**: Some requirements have been omitted
  - ❑ Possibly due to oversight.
- **Example:**
  - ❑ The analyst has not recorded: when temperature falls below 90 C
    - ❖ heater should be turned ON
    - ❖ water shower turned OFF.

Requirements Analysis



Formalization during analysis

Formalization helps identify areas of ambiguity as well as inconsistencies and omissions

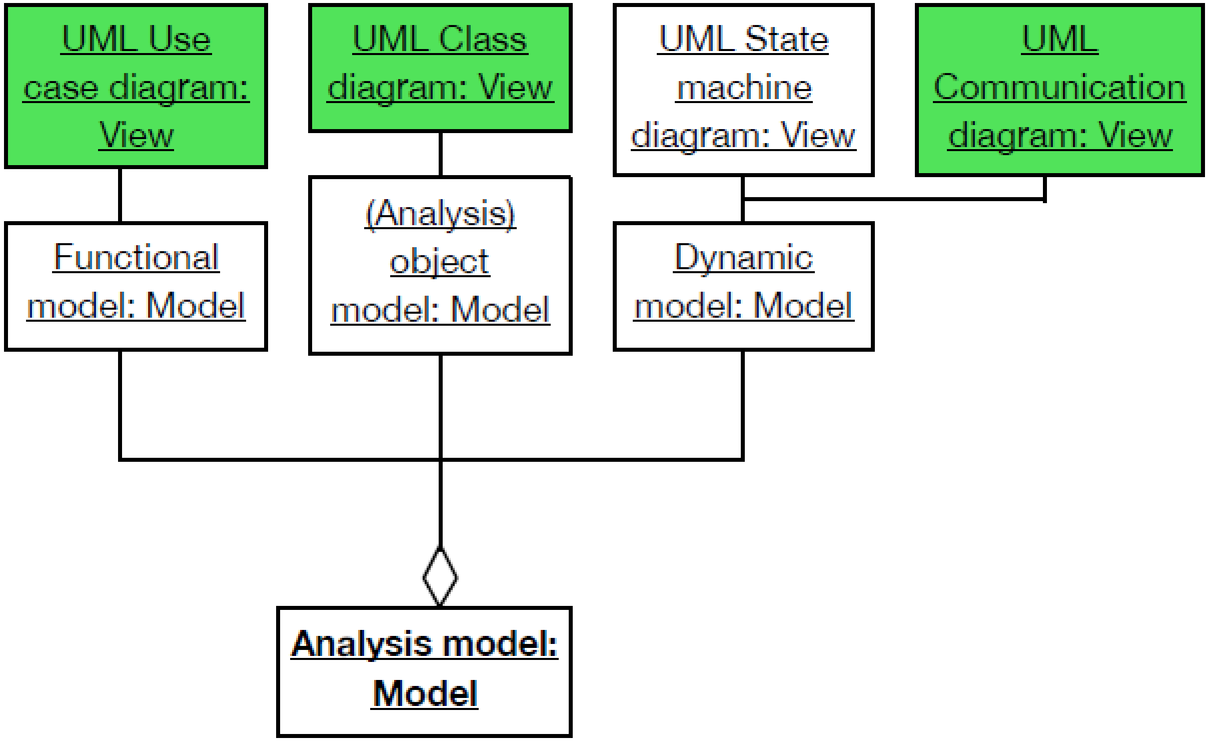
We can deal with ambiguity by using standardized **notations** instead of **drawings**

For Analysis we use the following **UML** diagram types:

use case diagram, class diagram, state machine diagram and sequence diagram

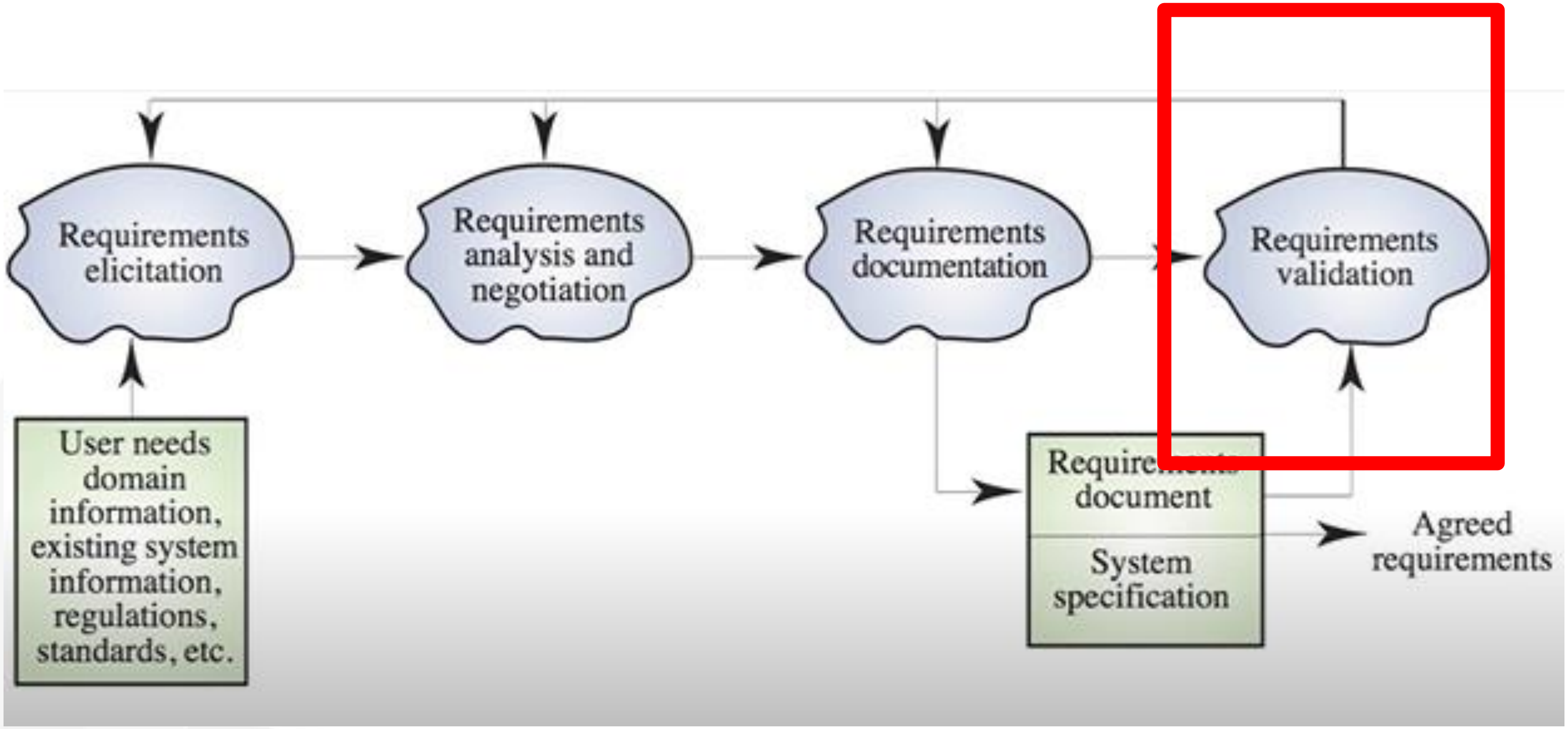
Requirements Analysis

Analysis model


UML Use case diagram: View
UML Class diagram: View
UML State machine diagram: View
UML Communication diagram: View
Functional model: Model
(Analysis) object model: Model
Dynamic model: Model
Analysis model: Model

Analysis Model can be Classified in three types : Functional Model, Object Model and Dynamic Model

Requirement Engineering Process


Requirements elicitation
Requirements analysis and negotiation
Requirements documentation
Requirements validation
User needs domain information, existing system information, regulations, standards, etc.
Requirements document
System specification
Agreed requirements

Requirements Validation

- Requirements validation makes sure that **requirements meet stakeholders' goals and don't conflict with them**.
- Requirements error costs are high so validation is very important
  - ❑ Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

Requirements Validation

- Validity. Does the system provide the functions which best support the customer’s needs?
- Consistency. Are there any requirements conflicts?
- Completeness. Are all functions required by the customer included?
- Realism. Can the requirements be implemented given available budget and technology
- Verifiability. Can the requirements be checked?

Requirements Validation

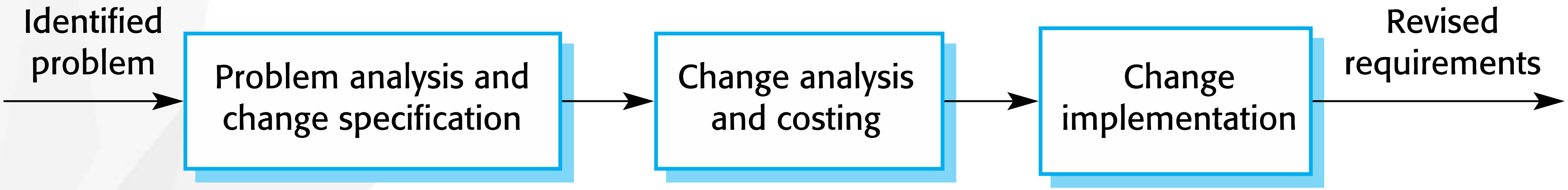
- Techniques:
  - ❑ **Requirements reviews**: Systematic manual analysis of the requirements.
  - ❑ **Prototyping**: Using an executable model of the system to check requirements.
  - ❑ **Test-case generation**: Developing tests for requirements to check testability.

Problem:
Requirements Change

- ➢ The business and technical environment of the system always changes after installation.
- ➢ The people who pay for a system and the users of that system are rarely the same people
- ➢ Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.

Solution:
Requirements Management

- Requirements management is the process of managing changing requirements during the requirements engineering process and system development.
- New requirements emerge as a system is being developed and after it has gone into use.
- Tool support: Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems.


Identified problem
Problem analysis and change specification
Change analysis and costing
Change implementation
Revised requirements