CMPS411
Spring 2018

Lecture 2

Requirements Elicitation and Analysis

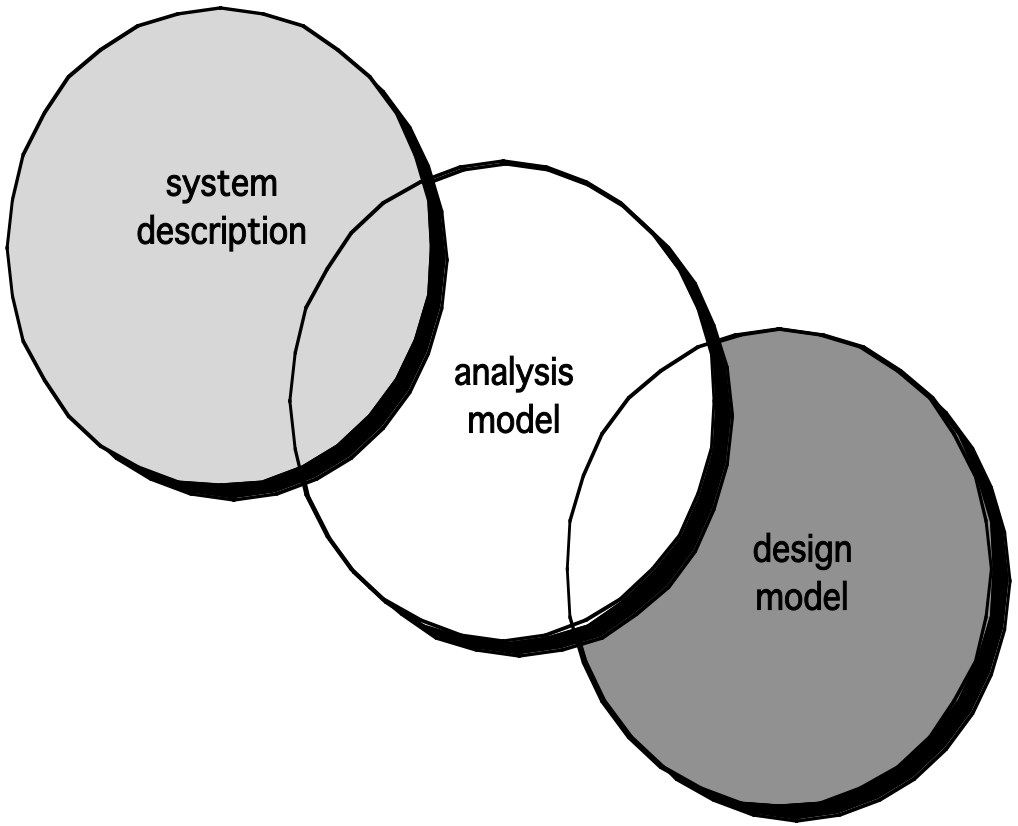
Topics Covered

- ✧ Requirement engineering
- ✧ Types of requirements
  - User and system requirements
  - Functional and non-functional requirements
- ✧ Techniques of requirements elicitation
- ✧ Scenario
- ✧ Domain analysis
- ✧ Review of requirements

What is a Requirement

- **A statement about the proposed system that all stakeholders agree.**
  - Short and concise piece of information on “what”
  - Says something about the system design (how)
  - All the stakeholders have agreed that it is valid
- **The process of establishing the services that**
  - The customer requires from a system, and
  - The constraints under which it operates and is developed.
- **The software requirements document**
  - It is the official statement of what is required of the system developers.
  - It should include both a definition of user requirements and a specification of the system requirements.
  - It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it.

A Bridge



Problems of Requirements Analysis

- Stakeholders don’t know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organisational and political factors may influence the system requirements.
- The requirements change during the analysis process. New stakeholders may emerge and the business environment may change.

Functional and Non-functional Requirements

- **Functional requirements**
  - Describe *what* the system should do
  - May state what the system should not do.
- **Non-functional requirements**
  - Constraints that must be adhered to during development
  - Often apply to the system as a whole rather than individual features or services.

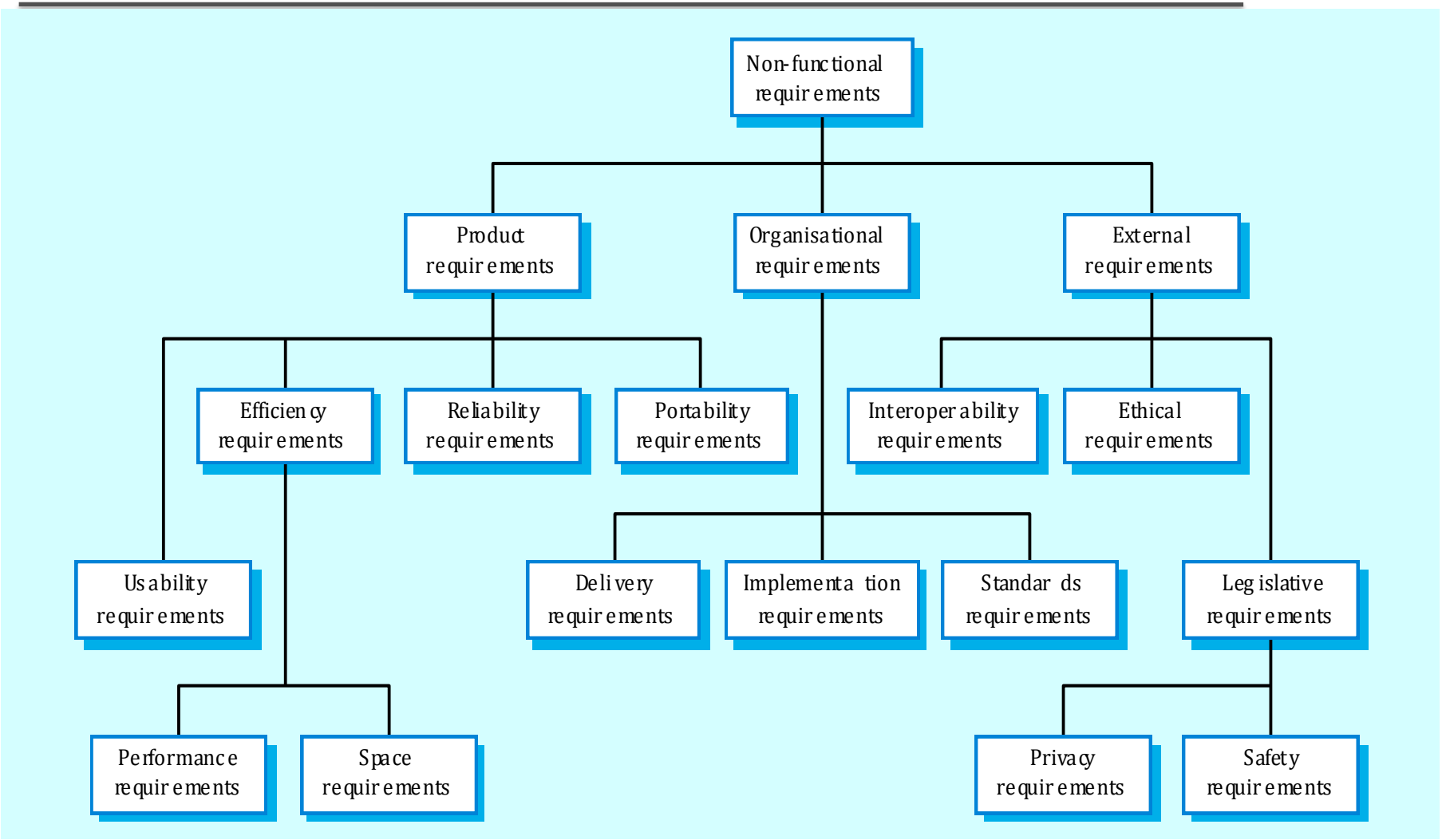
Functional Requirements

- Describe functionality or system services
- What *inputs* the system should accept
- What *outputs* the system should produce
- What data the system should *store* that other systems might use
- What *computations* the system should perform
- The *timing and synchronization* of the above

Non-Functional Requirements

- These define system properties and constraints e.g.
  - Reliability, response time, and storage requirements.
  - Constraints are I/O device capability, system representations, etc.
- Non-functional requirements may be more critical than functional requirements.
- If non-functional requirements are not met, the system may be useless.

Non-Functional Requirement Types



Examples of Nonfunctional Requirements in a Hospital System

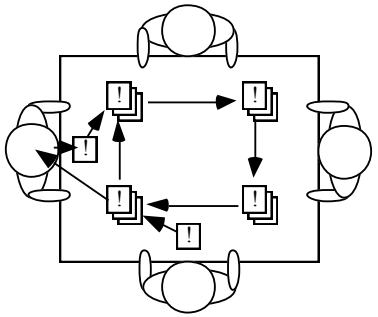
- The system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day (**Availability)**
- Users of the hospital system shall authenticate themselves using their health authority identity card (**Security**).
- The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized (**Usability)**
- The system shall implement patient privacy provisions as set out in privacy policy of the hospital (**Privacy)**
- The response time of the system should be un-noticeable **(Efficiency)**

Example: Requirements for a House

- I'd like to build a building.
  - What do you want to do with it?
- I'd like to sleep in it.
  - You mean you want to build a house?
- Yes.
  - What kind of house?
- A big one, with everything – the block is 15m by 50m.
  - How many bedrooms do you want?
- Well, I have 2 children, so I guess 3 bedrooms
  - OK, so 3 bedrooms…
- Wait! We're planning on another child, and sometimes friends stay over, so maybe 4/5 bedrooms?
- …
- Later: I need 100MB network wiring throughout the study and lounge with output ports 15cm above the floor and 20cm in from the walls based on IEEE standard XX34T…

Techniques for Elicitating and Analysing Requirements I

- Brainstorming
  - Appoint an experienced moderator
  - Arrange the attendees around a table
  - Decide on a ‘trigger question’
  - Ask each participant to write an answer and pass the paper to its neighbour



- Prototyping
- Scenario
- Domain analysis

Techniques for Elicitating and Analysing Requirements II

- Observation
  - Read documents and discuss requirements with users
  - Shadowing important potential users as they do their work
    - ask the user to explain everything he or she is doing
  - Session videotaping
- Interviewing
  - Conduct a series of interviews
    - Ask about specific details
    - Ask about the stakeholder's vision for the future
    - Ask if they have alternative ideas
    - Ask for other sources of information
    - Ask them to draw diagrams

Gathering and Analysing Requirements using Prototyping

✧ Prototyping

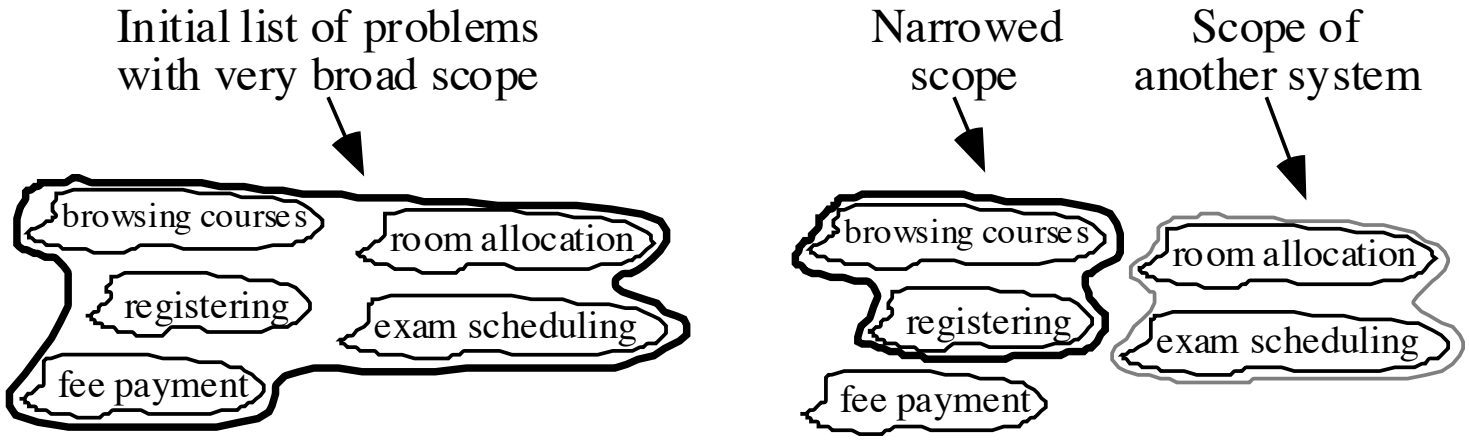
- The simplest kind: *paper prototype*.
  - a set of pictures of the system that are shown to users in sequence to explain what would happen
- The most common: a mock-up of the system's user interface (UI)
  - Written in a rapid prototyping language
  - Does *not* normally perform any computations, access any databases or interact with any other systems
  - May prototype a particular aspect of the system

Gathering and Analysing Requirements using Scenarios

- Scenarios are real-life examples of how a system can be used.
- They should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.

Defining the Scope of the System

- ✧ The system scope must be precisely defined
- ✧ System boundary should be clearly marked
- ✧ Narrow the *scope* by defining a more precise problem
  - List all the things you might imagine the system doing
    - Exclude some of these things if too broad
    - Determine high-level goals if too narrow
- ✧ Example: A university registration system



Domain Analysis of the System

- Requirements engineering process also include domain analysis
- In domain analysis a software engineer learns about the domain to better understand the problem:
    - The *domain* is the general field of business or technology in which the clients will use the software
    - A *domain expert* is a person who has a deep knowledge of the domain
    - Example of domain:
        - Medical (Hospital management system)
        - Education (University student system)
        - Finance (Banking system)

Key points

- Requirements for a software system set out what the system should do and define constraints on its operation and implementation.
- Functional requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out.
- Non-functional requirements often constrain the system being developed and the development process being used.
- Various techniques are used to formulate requirements
- Scenario-based
- Domain analysis.

References

- TIMOTHY C LETHBRIDGE: OBJECT-ORIENTED SOFTWARE ENGINEERING: PRACTICAL SOFTWARE DEVELOPMENT USING UML AND JAVA.
- R. Pressman: *Software Engineering*: *A Practitioner's Approach*, 2010.
- Blaha, M. and Rumbaugh, J.: Object-Oriented Modelling and Design with UML. Pearson Prentice-Hall, 2005. ISBN: 0-13-196859-9.