SE - 2
Requirement engineering

Lydie du Bousquet

Lydie.du-bousquet@imag.fr

In collaboration with J.-M. Favre, I. Parissis, Ph. Lalanda, Y. Ledru

Schedule

- What / Who / Why …
  requirement engineering
- Serious game: about requirement elicitation
- Serious game: try to do it
- More about Requirement specification

Who is concerned
by requirement engineering?

People

- The **customer** side

  pays for the product and
  usually decides the requirements

- The **supplier/provider** side

  produces a product for a customer

- The **User**

  operates or interacts directly with the product
  may be different from the customer

- What are requirements?
- What is requirement engineering?

Need to know the requirements to be able to build the program

Requirements are **statements** of

- **what** the system must **do**,
- **how** it must **behave**,
- the **properties** it must exhibit,
- the **qualities** it must possess, and
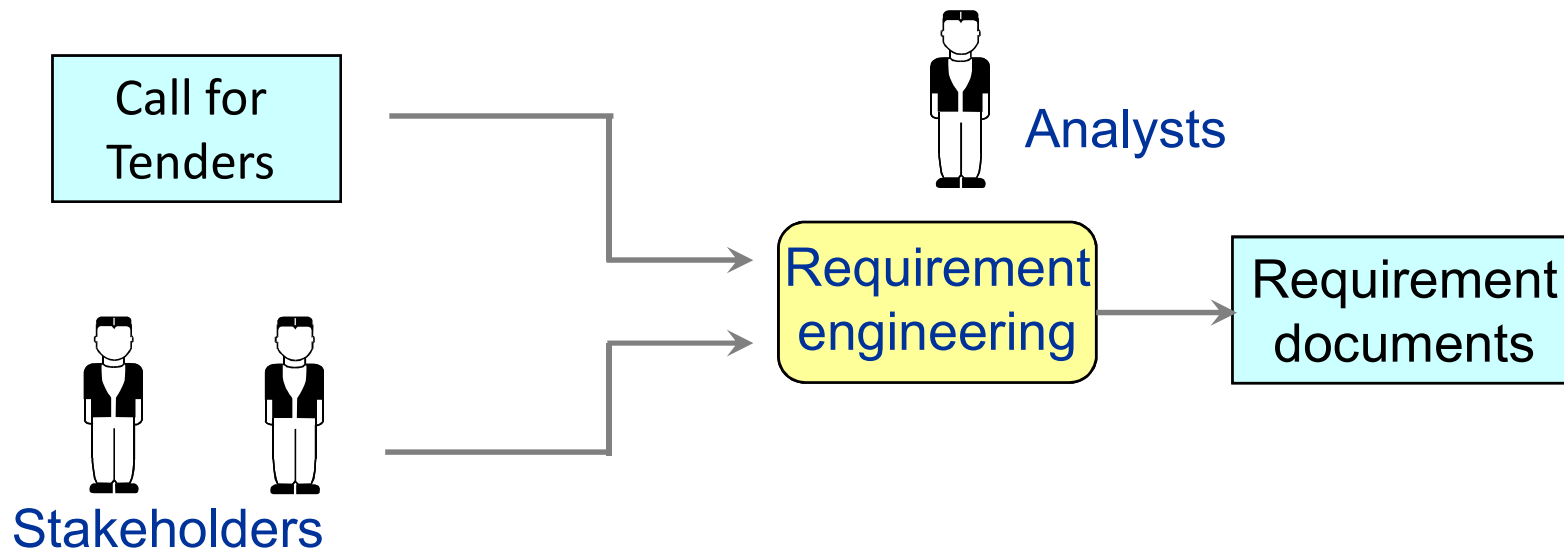- the **constraints** that the system and its development must satisfy

Requirement engineering
Objectives

- Find out the needs and constraints of the customers
- Specify them in a dedicated document

Requirement engineering
Objectives

WHAT?

- **Call for tender**
  - first expression of the customer's **needs** and **constraints**
  - basis for a bid for a contract
- **Stakeholders**
  - All the **people** having an **interest** in the project
  - Customer side: users, experts, managers, sales men ...
  - Supplier side: sales men, development teams, architects, managers, strategist
- **Requirements**
  - Come from the **customer side**
  - Written by customer, supplier, or both in a **Software Requirements Specification** document (SRS)

- Why requirement engineering?

Requirement Engineering

The **hardest** single part of building
a software system
is deciding **precisely**
what to build. . .

Deciding **precisely** what to build is hard

Deciding **precisely what to build...**

- Customer may **not know** precisely what he want

- Needs may **change**

- There may be **conflicts**

- Problem may be **difficult** to be understood

- Customer and supplier may speak **different languages**

- Many different kinds of information

I want something to get me across town in the shortest time

Requirement Impacts

- **Legal** impacts
  - Basis of the **contract** between customer and supplier
- **Economic** impacts
  - Cost of **correcting** wrong requirements
  - **Relevance** of the marketed product
- **Social** impacts
  - Wrong requirements may cause **disasters**
- **Usage** impacts
  - **Acceptance** or **rejection** of a software

- How is organized Requirement engineering?
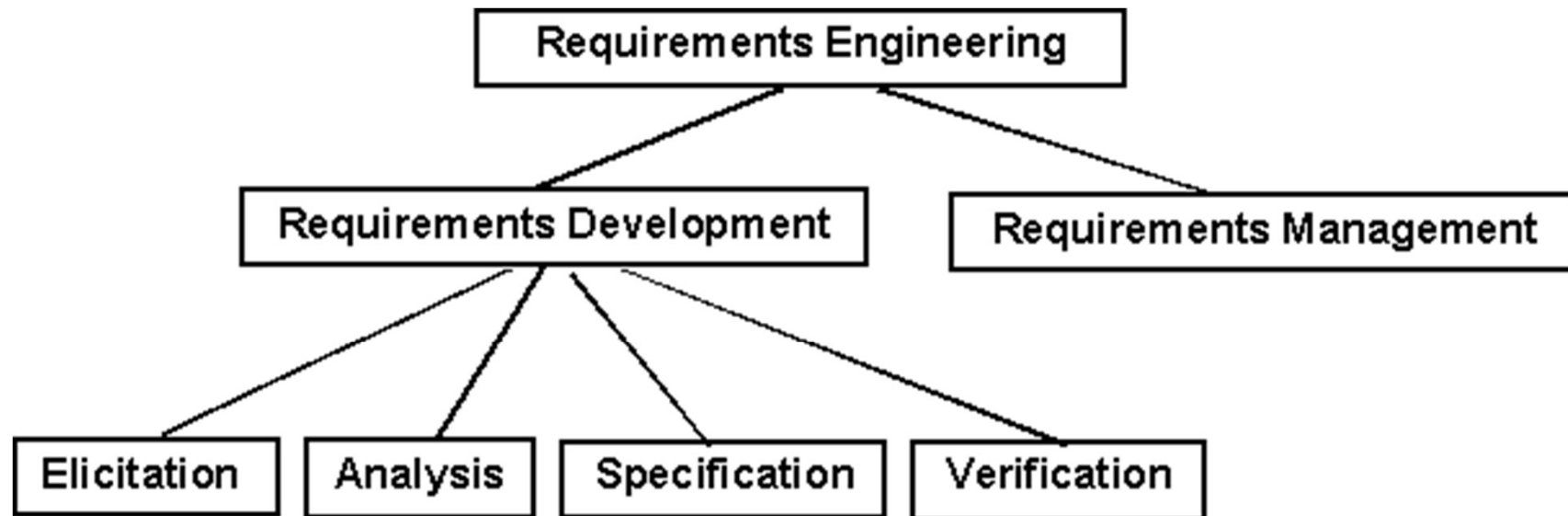
How is organized Requirement engineering?

How?

Figure 2. Subdisciplines of requirements engineering.

http://www.processimpact.com/articles/telepathy.html
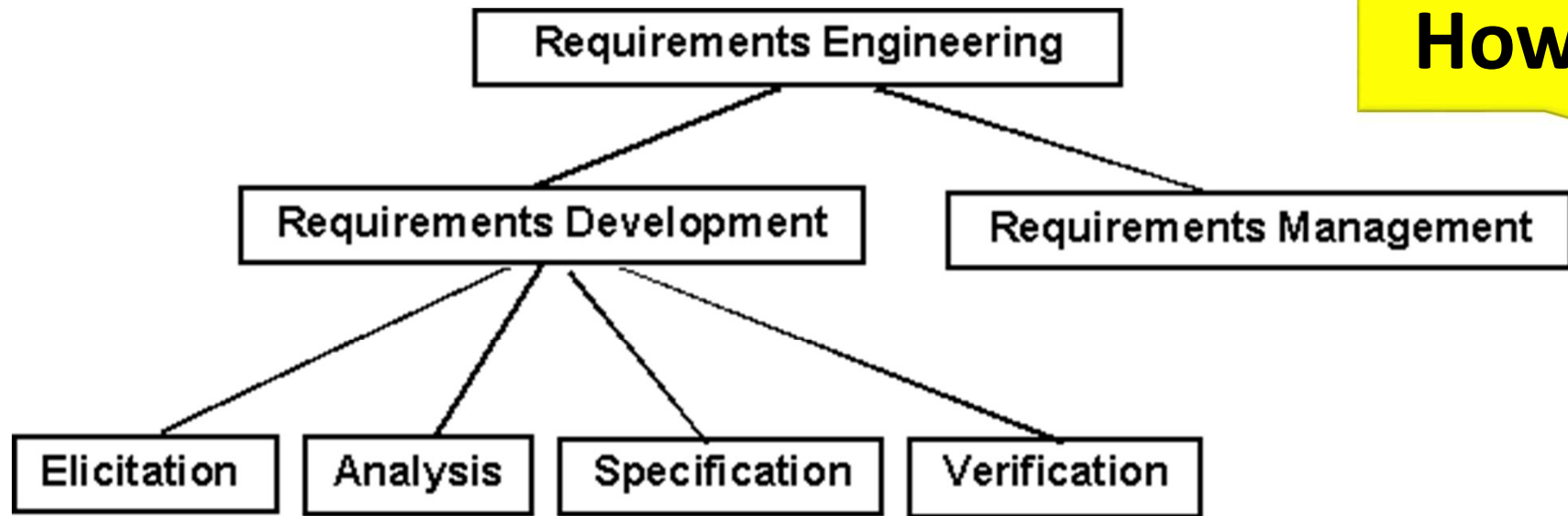
Figure 2. Subdisciplines of requirements engineering.

How is organized Requirement Engineering?

REQUIREMENT DEVELOPMENT

Elicitation > Analysis > Spec. > Verif.

Requirement elicitation

- Learning and understanding the needs of the users
- To avoid confusion between stakeholders and analyst
  – Understand the application domain
  – Identifying the source of requirements
  – Analyzing the stakeholders
  – Selecting techniques, approaches and tool to use
  – Eliciting the requirements
- 4 types of methods
  – Conversational (based on conversation)
  – Observational (based on observation)
  – Analytic
  – Synthetic

Requirement elicitation -2
Difficulties

- Users are not fully aware of what they will obtain
- They may not make the difference between
  - What they need and what they have
  - What they want and what they need
- They may not want to work on the problem
- They may use specific language
- They may have forgotten some important information
- The analyst may want to find a solution before knowing the problem or may conclude too quickly

Requirement elicitation -3
Example of **communication problem**

- The user requests to change an incorrect algorithm on the existing system
  - Analyst: "How often this algorithm is used ?"
  - User: "Never"
  - So the request is ignored

- Of course, the reason why the algorithm is not used is because it is **incorrect**!
  - Be careful to rapid conclusion
  - Be careful not to decide for the user

Requirement elicitation -4
Example of **communication problem**

- The user "My software is too slow"

- Analyst 1:
  "I think it is due to the hadware"

- Analyst 2:

  "Could you tell me why you think it is slow?"

  Who do you trust more? Why?

Requirement analysis

- Analyze the results of elicitation
  - are the answers consistent?
  - identify trouble spots/conflicts
  - identify limits?
  - identify most important requirements?
- Possibly iterate over elicitation again
- Conflict resolution

Requirement specification

- Process of writing down the requirements

- No standard nor methods

- From informal to formal

- Functional and non-functional

- **Software Requirements Specification** document (SRS)

Detailed after

Requirement verification

- Process of checking that the result is OK
  - **Unitary**: only one thing by requirement
  - **Complete**: no missing information
  - **Consistent**: no contradiction among requirements
  - **Unambiguous**: objective facts, comprehensible, …
  - **Prioritized**: level of importance is given
  - **Traceable**: source/reason/links are documented
  - **Verifiable**: can be checked at the end
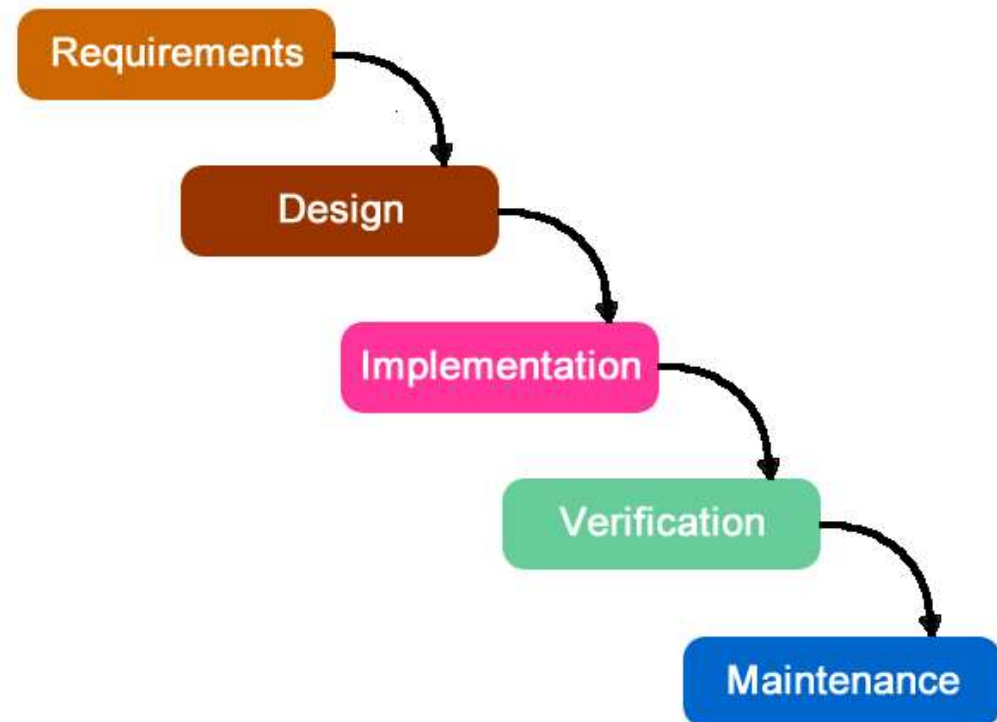 - …

- When requirement engineering?

Requirements influence all the software activities
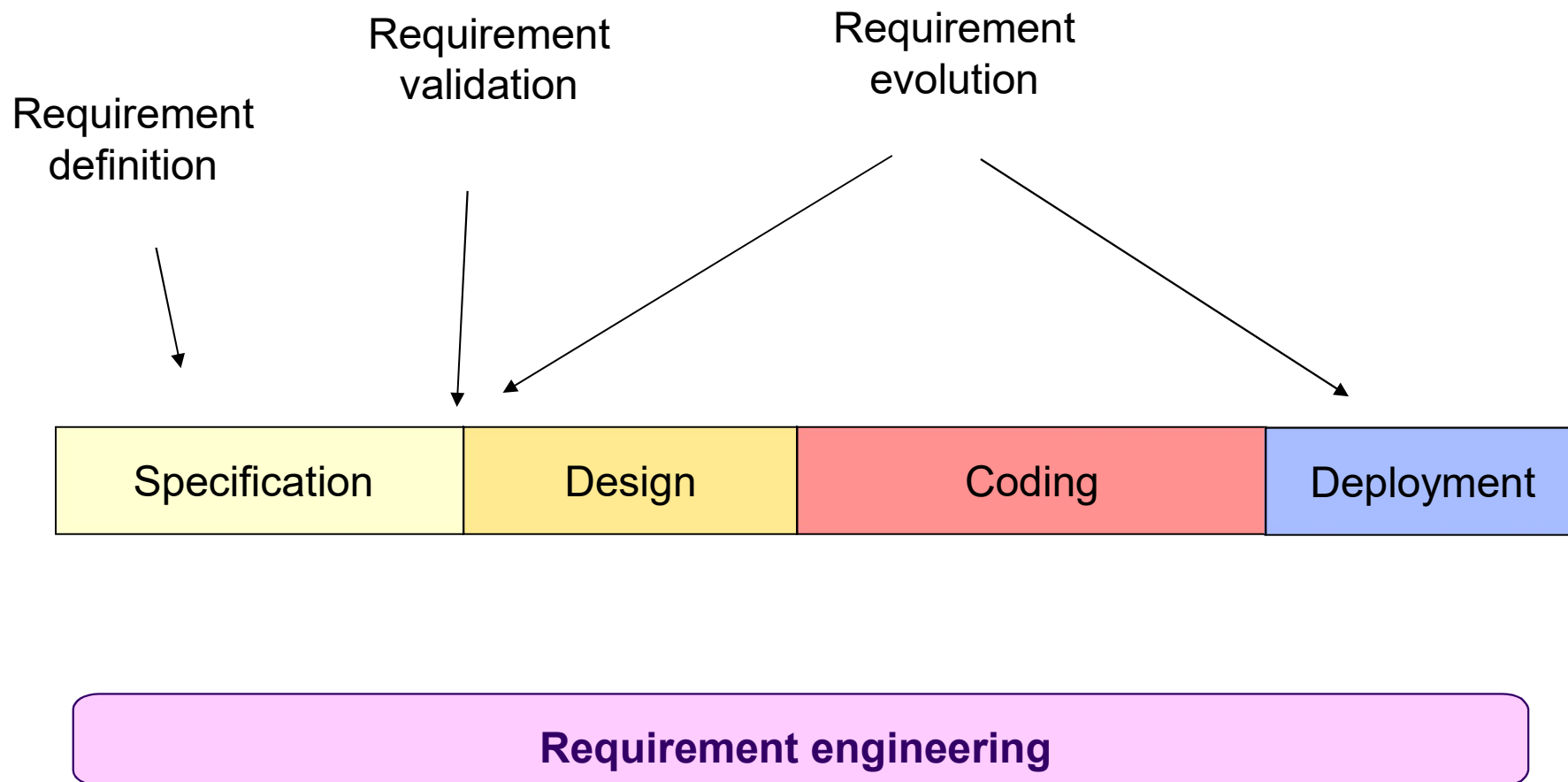
- Design
  - Architecture
  - Detailed design
- Implementation
- Validation
- Acceptance, ...

Requirements and life cycle

WHEN?

Requirement
definition

Requirement
validation

Requirement
evolution

Specification	Design	Coding	Deployment

Requirement engineering

Requirement evolution and traceability

- User requirement may **evolve** (will)
- Should be **taken into account**
  - Possible if the initial elicitation work, analysis and validation has been carefully carried out
- **Impacts** should be evaluated
  - Possible only if traceability mechanisms
  - Tool can help to automate the links among the requirements

Schedule

- What / Who / Why … requirement engineering
- Serious game: about requirement elicitation
- Serious game: try to do it
- More about Requirement specification

Requirement elicitation

Week 2

Technique	Good for	Kind of data	Plus	Minus
Questionnaires	Answering specific questions	Quantitative and qualitative data	Can reach many people with low resource	The design is crucial. Response rate may be low. Responses may not be what you want
Interviews	Exploring issues	Some quantitative but mostly qualitative data	Interviewer can guide interviewee. Encourages contact between developers and users	Time consuming. Artificial environment may intimidate interviewee
Focus groups and workshops	Collecting multiple viewpoints	Some quantitative but mostly qualitative data	Highlights areas of consensus and conflict. Encourages contact between developers and users	Possibility of dominant characters
Naturalistic observation	Understanding context of user activity	Qualitative	Observing actual work gives insight that other techniques cannot give	Very time consuming. Huge amounts of data
Studying documentation	Learning about procedures, regulations, and standards	Quantitative	No time commitment from users required	Day-to-day work will differ from documented procedures

[1] Preece, Rogers, and Sharp "Interaction Design: Beyond human-computer interaction", p214

Exercise 4

Week 2 – Requirement engineering Schedule

- What / Who / Why … requirement engineering
- Serious game: about requirement elicitation
- Serious game: try to do it
- More about Requirement specification

Serious Game: CyberVideo

- Teams of 4-5 students
  - 1 Client
  - 3 or 4 providers
- Starting document
  - Providers: Call for tender (few lines)
  - Client: informal description (3 pages)
- Objectives Providers:
  - Ask client questions in order to find all the requirements described in the informal description
  - Propose a specification document

Serious Game: CyberVideo

Providers:

- What elicitation method did you use?
- Did you chose it on purpose?
- Could you have done differently? How?

- What was difficult?
- How are you going to write the specification?

Schedule

- What / Who / Why … requirement engineering
- Serious game

- More about Requirement specification
- Exercises / Homework
  - Express requirements for CyberVideo
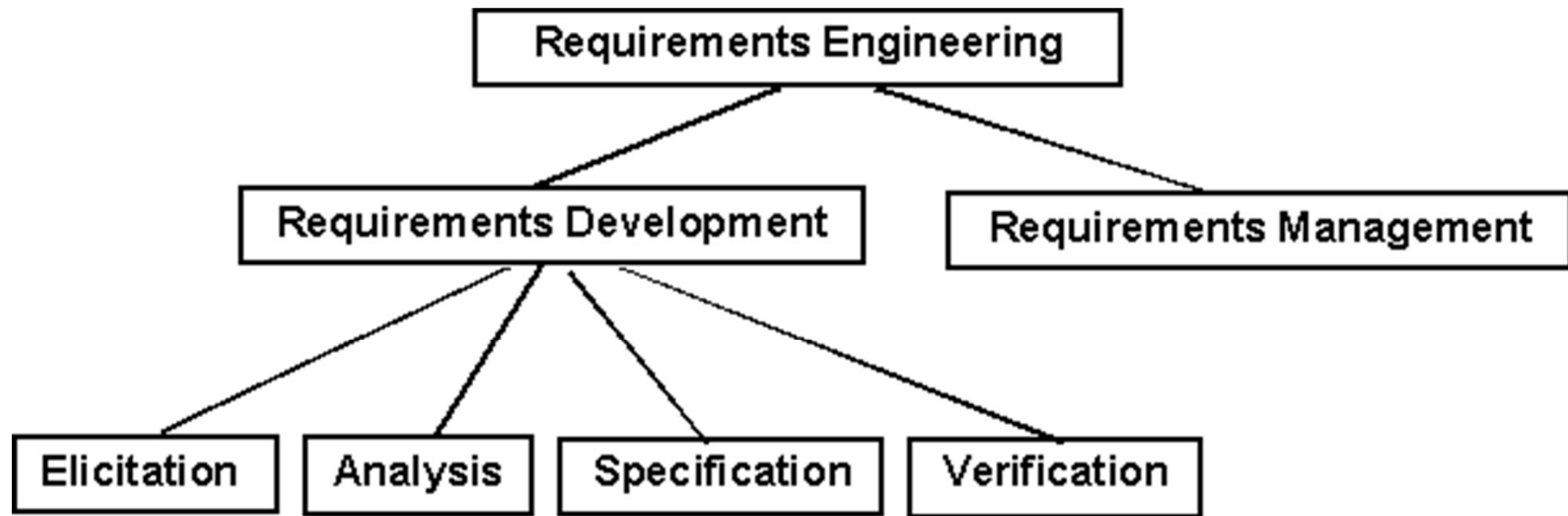 - Read (and learn) about requirement elicitation

Figure 2. Subdisciplines of requirements engineering.

How is organized Requirement Engineering?

> Requirements development

REQUIREMENT SPECIFICATION

What is a requirement?

- Express what customers want or need
  - A « **need** » is something **mandatory** that we must have
  - A « **want** » is **nice** to have but not always mandatory
- A requirement can be
  - A goal
  - A provided function
  - A quality
  - A property (domain, organization)
  - A constraint

Functional requirements

- **Services provided** by the system
  - Description of the expected function or behavior
  - A general property
  - Expected UI

- Example (**Software library**)
  - The software has to manage books borrowing
  - The checkout function starts by reading the subscriber card
  - A subscriber has to pay 20 euros per year
  - A subscriber is defined by his name, age, etc.
  - All needed information have to be displayed in a single window

Functional requirements – Davis, 93

- ## An **object**
  - A **client** is identified by his name, age and address
- ## A **function**
  - A client can **borrow** up to 5 books
- ## A **state**
  - A book is **available, borrowed or lost**

- ## Several of them together

Objects – Davis, 93

- **Entity** clearly identified
  - Concept related to the software
- Requirements specify the objects:
  - Name and meaning
  - Structure
  - Scope
- Library Example
  - A **client** is identified by his **name**, **age** and **address**
  - A **book** is defined by its **title** and **author(s)**

Functions – Davis, 93

- **Activity** clearly defined in the domain
  - Tasks, services, processes
  - Related to the software
- Requirements specify the functions:
  - Name
  - Interface, data
  - Behavior
  - Demanded ressources
- Library Example
  - **Book checkout**: to borrow a book, the client has first to show his library card and then the book

State – Davis, 93

- Characterize the situation of an entity
  - Can be expressed as a predicate
  - Can change over the time
  - Influence the behavior of the entity
- Requirements specify the states
  - All possible states
  - Transitions
  - Possible properties
- Library Example
  - A book is **available, borrowed or lost**

Objects, functions, states

- Requirements may established **relations** among **objects**, **functions** and **states**
  - A **client** can **borrow** a book when he has paid his bill and has less than 5 **borrowed** **books**

- Analysis methods focus on single aspect
  - Objects
  - Functions
  - States

Capturing functional requirements
Example of questions – Pfleeger

- Functions
  - What does the system should do?
  - When?
  - Several functional modes?
  - Appropriate responses to stimuli ?

- Data
  - Format?
  - How long should be kept?

Non-functional requirements
Constraints

A constraint under which a software
operates or is developed

Process
- Tools
- Standards

Domain
- Usage
- Regulation / Law

Development
- COTS (OS, middleware, …)
- Methods

Context
- Existing applications
- People

Non-functional requirements
Constraints

- ## Example (Software library)

  - UML **must** be used for the modeling phase (at design)

  - An architectural design document **must** be provided

  Process

  - The system **must** use Oracle for persistency functions

  - Java annotations **may** be used for development

  Development

  - No historic is maintained for subscribers

  Domain

  - The system **must** interface with legacy systems

  Context

Non-functional requirements
Qualities

- **External** or **internal** qualities of
  - the provided functions
  - the global system

- **Includes**
  - Security, Logging
  - Storage, Configuration
  - Performance, Cost
  - Interoperability, Flexibility
  - Accessibility, Disaster recovery

- Must be **quantified** (to be evaluated)

Non-functional requirements
Qualities

- ## What about… ?
  - ### "The system should be easy to use"
  - ### "The system should be robust and quick"

Non-functional requirements
Qualities

- What about... ?
  - "The system should be easy to use"
  - "The system should be robust and quick"

- Different interpretation

- Source of conflict

=> Need to quantify the NF-requirements

Non-functional requirements
Qualities

- Example (**Software library**)
  - Book checkout must be made **in less than 1 minute**
  - Any function must be **done in less than 2 minutes**
  - Backtracking must **always** be possible when borrowing a book
  - The software system must be **available 6 days a week**

Example

- First formulation
  - The system should be easy to use for a experimented user and should be organized to limit the number of errors.

- Second formulation
  - A user with 5 years of experience should be able to use the system after 2 hour long formation.
  - After the formation, the average number of errors made by a user should not be more than 2 by day.

Performance

- Quality perceived by **users** (external)

- Requirements to be specified

  – Number of transactions per second

  – Arrival rate of inputs

  – Refreshing time

  – Response time for a given pattern of events

  – What to do when expected quantities are exceeded

    - Failure, ignorance of additional inputs, degraded services

Usability

- Quality perceived by **users** (external)
- Requirements to be specified
  - Provided UI
  - Error messages
  - keyboard shortcut
  - Backtrack possibilities
  - Techniques to help users and to improve confidence
  - Amount of expected training
  - Facilities to avoid misuses

Availability
& Reliability

- Quality perceived by **users** (external)
- Requirements to be specified
  - Max. number of bug per Kline during integration
  - Min. duration without a problem
  - Is there a time to perform maintenance?
  - Maximum time allowed for restarting the system
  - Must backup copies be stored at a different location?
  - Must the system detect and isolate fault
- Difficult to assess !

Security

- Quality perceived by **users** (external)
- Requirements to be specified
  - Must access to the system or information be controlled?
  - Should each user's data isolated from the data of each other?
  - Should user programs be isolated from other programs and from the operating system?
  - Should precautions be taken against theft or vandalism

Maintenability

- Quality perceived by **engineers** (internal)
- Requirements to be specified
  - When and in what ways might the system be changed in the future?
  - How easy should it be to add features to the system?
  - How easy should it be to port the system from one platform (computer, OS) to another

Schedule

- What / Who / Why … requirement engineering

- Serious game

- More about Requirement specification

- Exercises
  - Express requirements for CyberVideo
  - Read (and learn) about requirement elicitation

Exercises/HomeWork

- Express some requirements about CyberVideo
  - Functional
  - Non-Functional
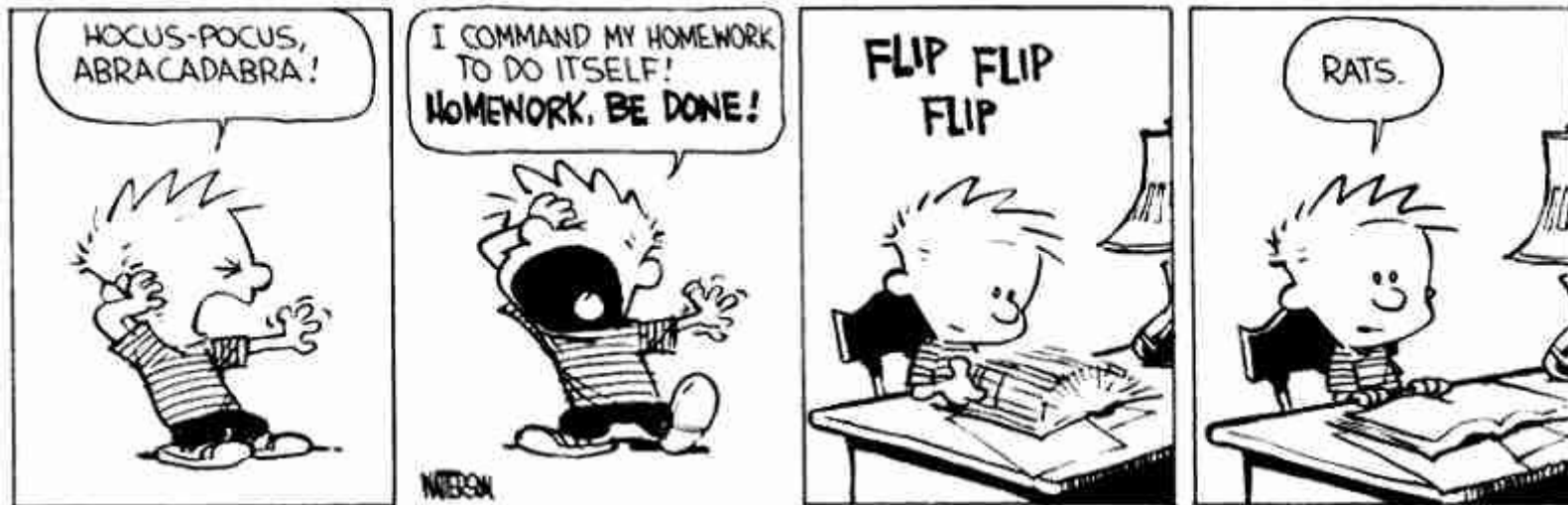
Exercise

- Read and learn about requirement elicitation
- At the end of the work, you should be able to
  - cite several elicitation methods
  - explain the principle of a given elicitation method
  - give advantages/limits of a given elicitation method

For the final evaluation

- You should know
  - Challenges and issues of requirement engineering
  - Advantages and limits of the different requirement elicitation methods to choose the most appropriate one
- You should be able to
  - Identify incorrect formulation of NF requirement
  - Propose an alternative formulation