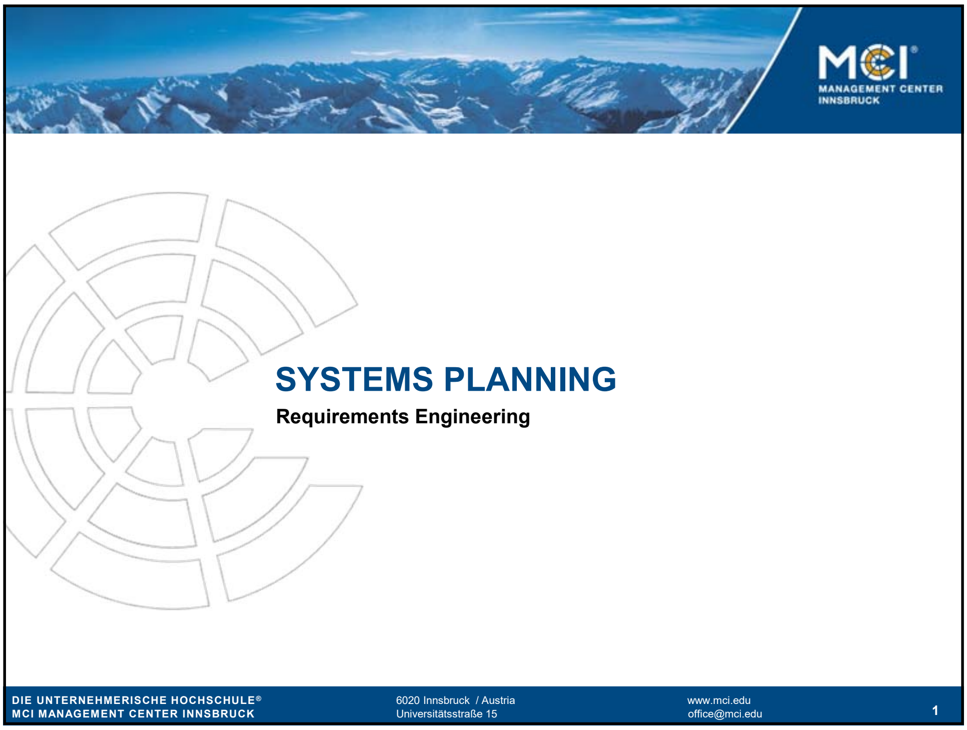
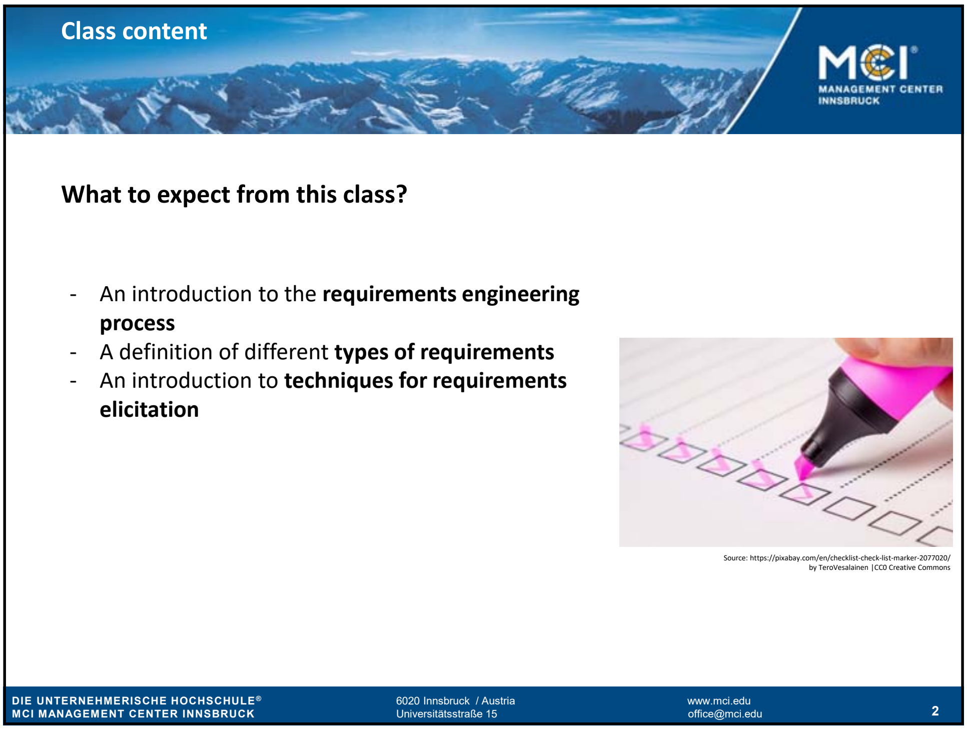


SYSTEMS PLANNING

Requirements Engineering

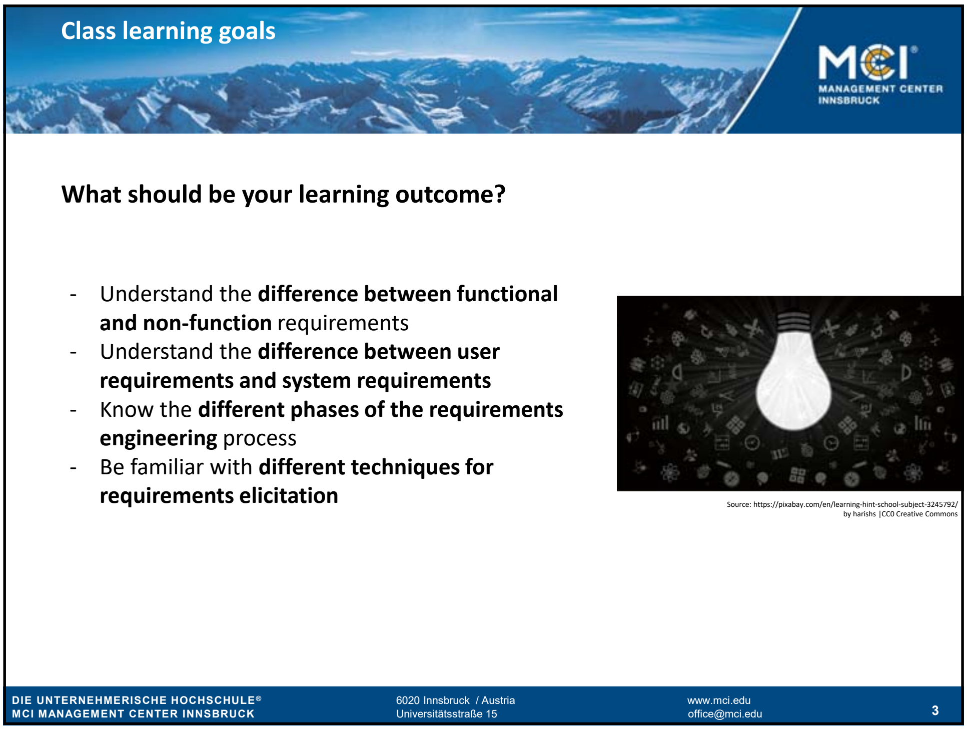


Class content

What to expect from this class?

- An introduction to the **requirements engineering process**
- A definition of different **types of requirements**
- An introduction to **techniques for requirements elicitation**

Source: https://pixabay.com/en/checklist-check-list-marker-2077020/
by TeroVesalainen |CC0 Creative Commons

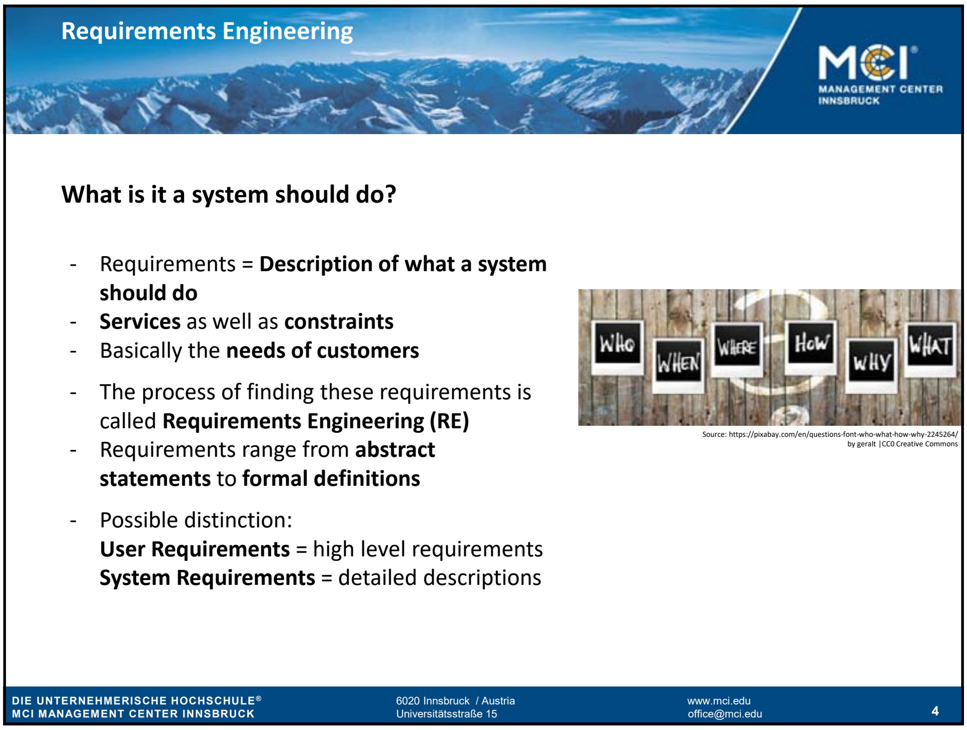


Class learning goals

What should be your learning outcome?

- Understand the **difference between functional and non-function** requirements
- Understand the **difference between user requirements and system requirements**
- Know the **different phases of the requirements engineering** process
- Be familiar with **different techniques for requirements elicitation**

Source: https://pixabay.com/en/learning-hint-school-subject-3245792/ by harishs |CC0 Creative Commons

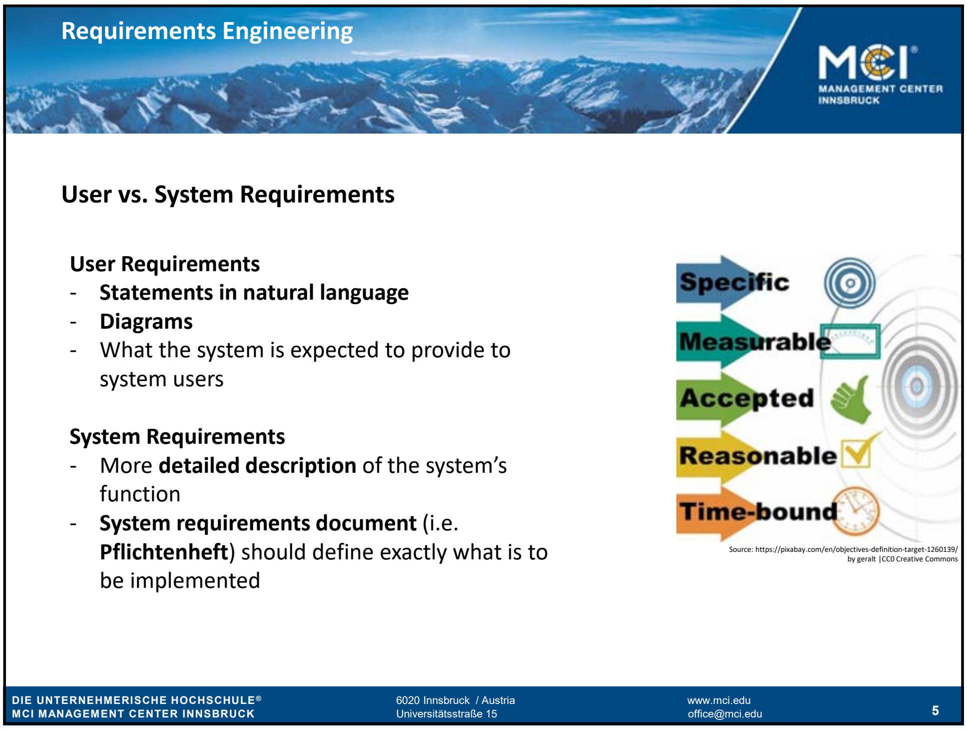


Requirements Engineering

What is it a system should do?

- Requirements = **Description of what a system should do**
- **Services** as well as **constraints**
- Basically the **needs of customers**
- The process of finding these requirements is called **Requirements Engineering (RE)**
- Requirements range from **abstract statements** to **formal definitions**
- Possible distinction:
  User Requirements = high level requirements
  System Requirements = detailed descriptions

Source: https://pixabay.com/en/questions-font-who-what-how-why-2245264/ by geralt |CC0 Creative Commons



Requirements Engineering

User vs. System Requirements

User Requirements

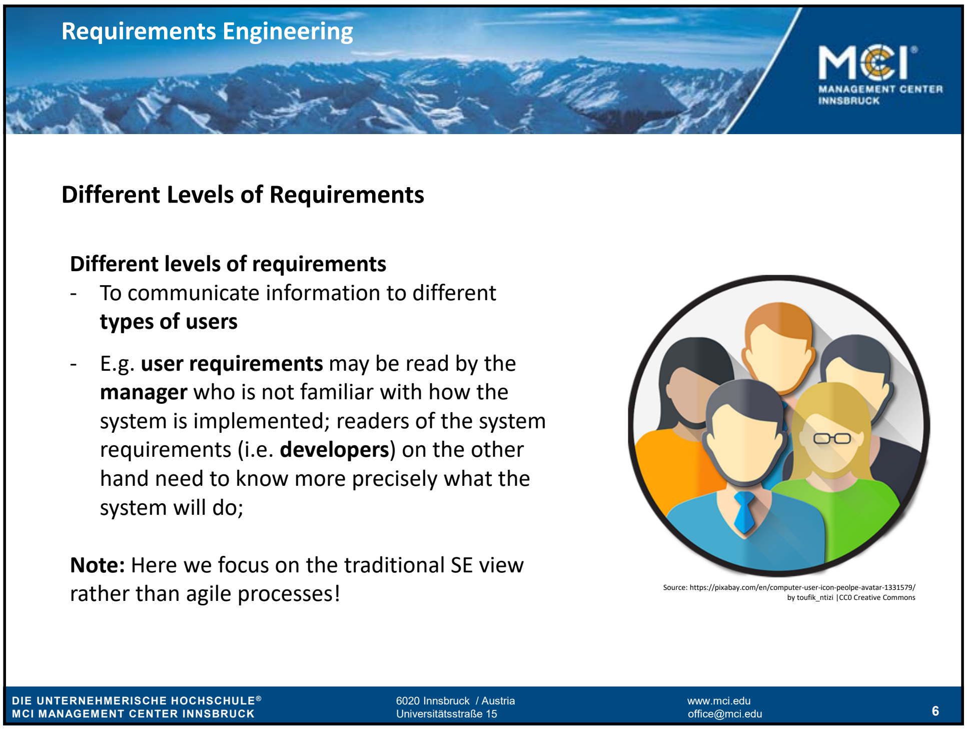
- **Statements in natural language**
- **Diagrams**
- What the system is expected to provide to system users

System Requirements

- More **detailed description** of the system's function
- **System requirements document** (i.e. **Pflichtenheft**) should define exactly what is to be implemented

Specific
Measurable
Accepted
Reasonable
Time-bound

Source: https://pixabay.com/en/objectives-definition-target-1260139/ by geralt |CC0 Creative Commons



Requirements Engineering

Different Levels of Requirements

Different levels of requirements

- To communicate information to different **types of users**
- E.g. **user requirements** may be read by the **manager** who is not familiar with how the system is implemented; readers of the system requirements (i.e. **developers**) on the other hand need to know more precisely what the system will do;

Note: Here we focus on the traditional SE view rather than agile processes!

Source: https://pixabay.com/en/computer-user-icon-peolpe-avatar-1331579/ by toufik_ntizi |CC0 Creative Commons

Functional and Non-functional Requirements

Functional Requirements
- What services should the system provide
- How should it react to inputs
- How should it behave in particular situations

Non-functional Requirements
- Constraints on the services or functions a system should provide
- E.g. Usability, Security, etc.

Note: The distinction between different types of requirements is not clear-cut. E.g. the non-functional security can translate into a functional "integration of a user authentication facility"

Source: https://pixabay.com/en/woman-burnout-multitasking-face-1733881/
by geralt |CC0 Creative Commons

Functional Requirements

- Describe **what the system should do**
- When **user requirements** then they are usually expressed in an **abstract way**
- Range from very **general descriptions to specific workflows**

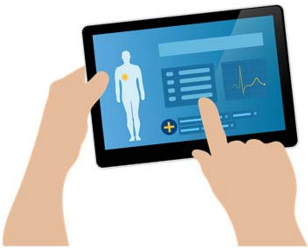
Source: https://pixabay.com/en/gear-gears-function-interaction-240137/
by geralt |CC0 Creative Commons

Functional Requirements: Example

Mental Health Care Patient Management System (MHC-PMS):

1. A user **shall be able to search the appointments list** for all clinics;
2. The system **shall generate each day, for each clinic, a list of patients** who are expected to attend appointments that day;
3. Each **staff member** using the system **shall be uniquely identified** by his or her eight-digit employee number;

Note: Different levels of detail (cf. requ. 1 and requ. 3)



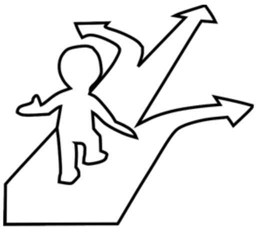
Source: https://pixabay.com/en/ehr-emr-electronic-medical-record-1476525/
by geralt |CC0 Creative Commons

Functional Requirements: Challenges

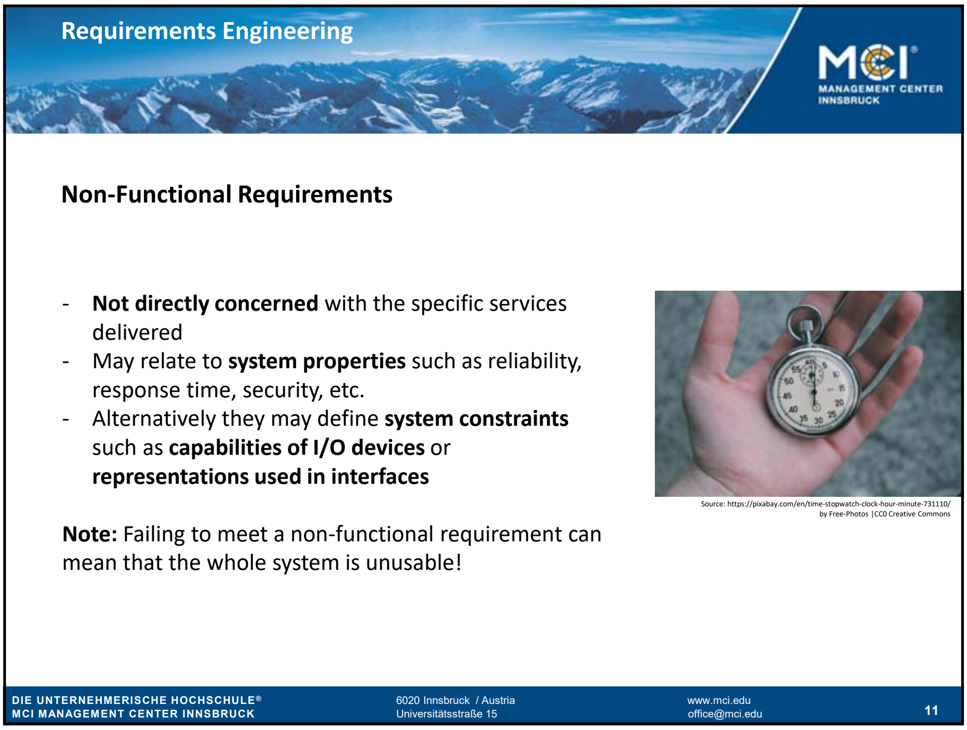
Imprecise or ambiguous requirements

- e.g. search may mean that given a specific name the system would **search the entire list of appointments**
- System developers may expect a user **to first specify the clinic to search in**

Theory vs. practical use
In theory the functional system specification should be both complete and consistent. In practice, for **large, complex systems**, it is practically impossible as it is **easy to make mistakes** and **hard to consider all stakeholders and their interests**.



Source: https://pixabay.com/en/confusion-left-right-straight-311388/
by Clker-Free-Vector-Images |CC0 Creative Commons



Requirements Engineering

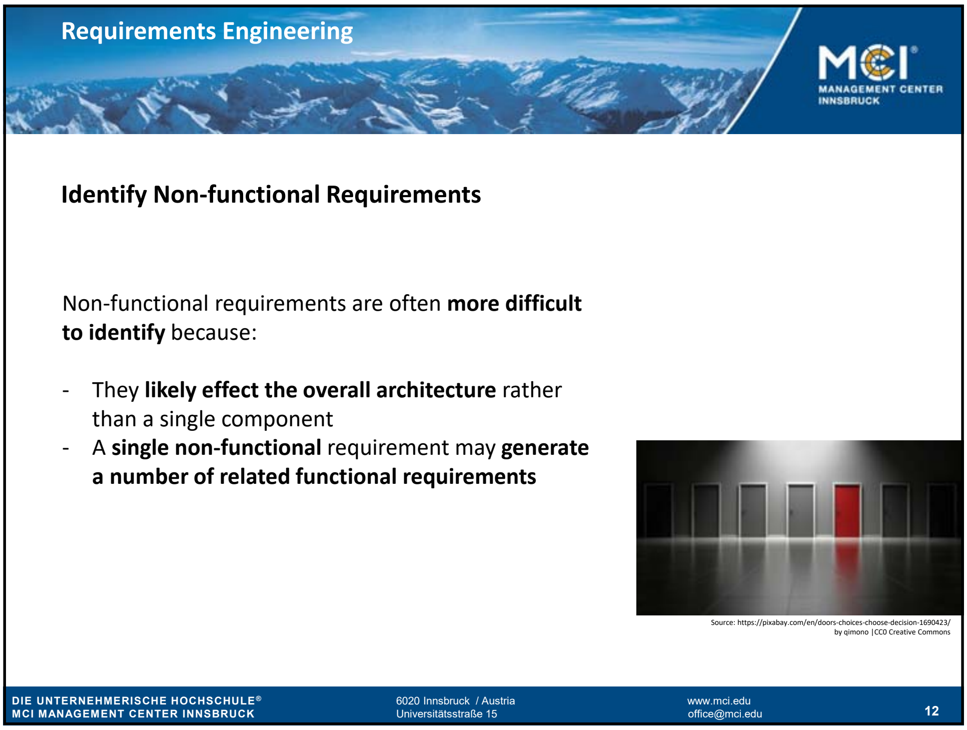
MCI MANAGEMENT CENTER INNSBRUCK

Non-Functional Requirements

- **Not directly concerned** with the specific services delivered
- May relate to **system properties** such as reliability, response time, security, etc.
- Alternatively they may define **system constraints** such as **capabilities of I/O devices** or **representations used in interfaces**

Note: Failing to meet a non-functional requirement can mean that the whole system is unusable!

Source: https://pixabay.com/en/time-stopwatch-clock-hour-minute-731110/
by Free-Photos |CC0 Creative Commons



Requirements Engineering

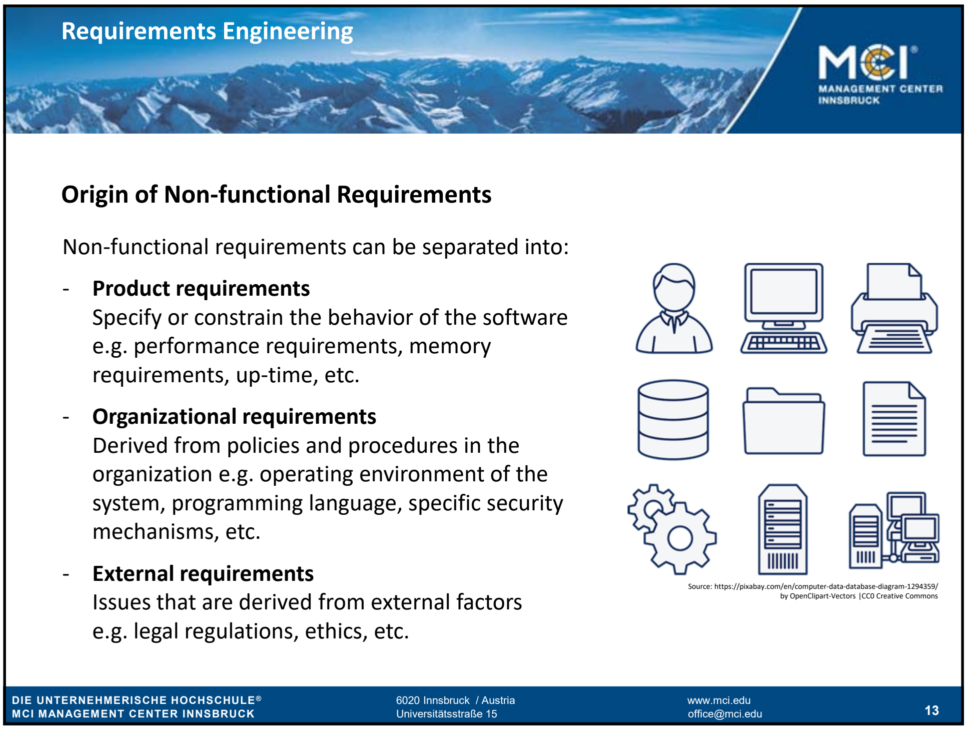
MCI MANAGEMENT CENTER INNSBRUCK

Identify Non-functional Requirements

Non-functional requirements are often **more difficult to identify** because:

- They **likely effect the overall architecture** rather than a single component
- A **single non-functional** requirement may **generate a number of related functional requirements**

Source: https://pixabay.com/en/doors-choices-choose-decision-1690423/
by qimono |CC0 Creative Commons



Requirements Engineering

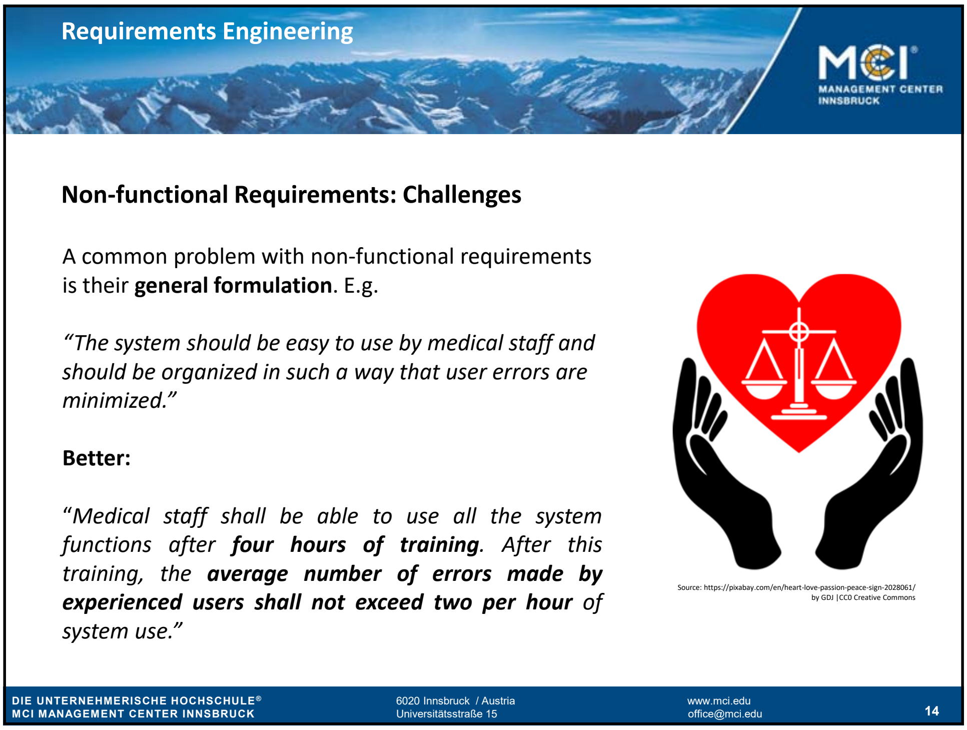
MCI MANAGEMENT CENTER INNSBRUCK

Origin of Non-functional Requirements

Non-functional requirements can be separated into:

- **Product requirements**
  Specify or constrain the behavior of the software e.g. performance requirements, memory requirements, up-time, etc.
- **Organizational requirements**
  Derived from policies and procedures in the organization e.g. operating environment of the system, programming language, specific security mechanisms, etc.
- **External requirements**
  Issues that are derived from external factors e.g. legal regulations, ethics, etc.

Source: https://pixabay.com/en/computer-data-database-diagram-1294359/ by OpenClipart-Vectors |CC0 Creative Commons



Requirements Engineering

MCI MANAGEMENT CENTER INNSBRUCK

Non-functional Requirements: Challenges

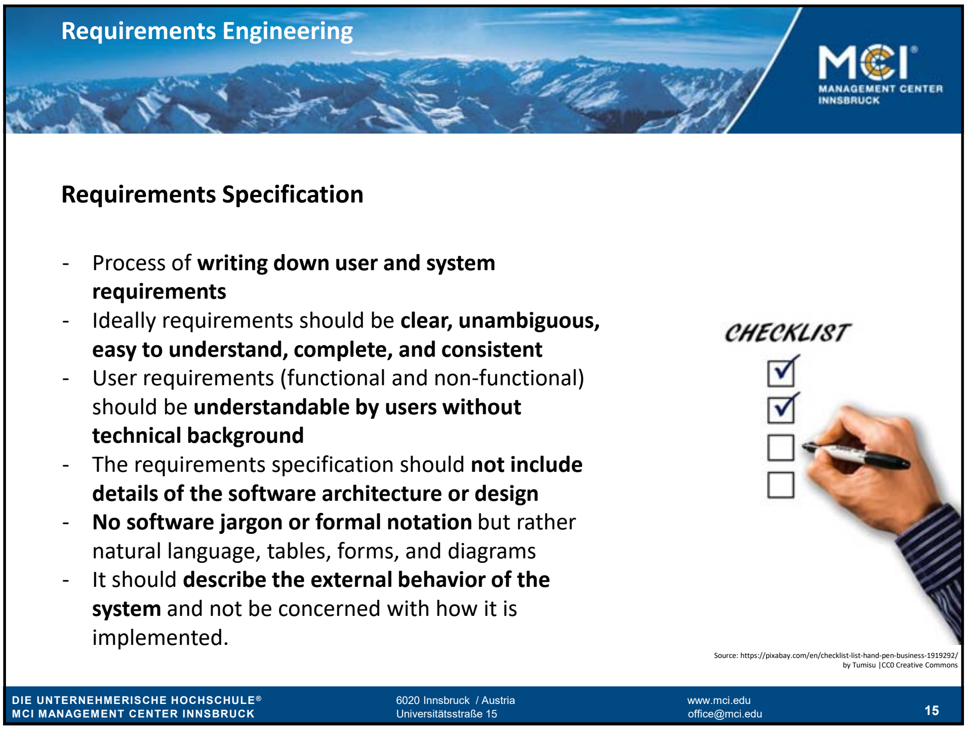
A common problem with non-functional requirements is their **general formulation**. E.g.

“The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.”

Better:

*“Medical staff shall be able to use all the system functions after **four hours of training**. After this training, the **average number of errors made by experienced users shall not exceed two per hour** of system use.”*

Source: https://pixabay.com/en/heart-love-passion-peace-sign-2028061/ by GDJ |CC0 Creative Commons

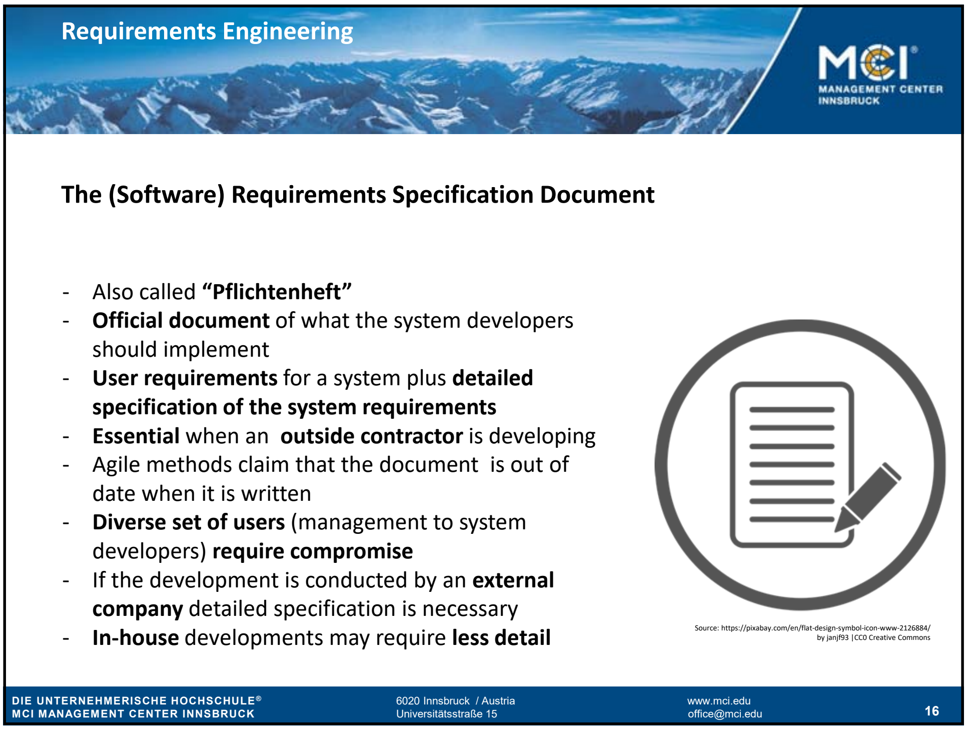


Requirements Engineering

Requirements Specification

- Process of **writing down user and system requirements**
- Ideally requirements should be **clear, unambiguous, easy to understand, complete, and consistent**
- User requirements (functional and non-functional) should be **understandable by users without technical background**
- The requirements specification should **not include details of the software architecture or design**
- **No software jargon or formal notation** but rather natural language, tables, forms, and diagrams
- It should **describe the external behavior of the system** and not be concerned with how it is implemented.

Source: https://pixabay.com/en/checklist-list-hand-pen-business-1919292/
by Tumisu |CC0 Creative Commons



Requirements Engineering

The (Software) Requirements Specification Document

- Also called **"Pflichtenheft"**
- **Official document** of what the system developers should implement
- **User requirements** for a system plus **detailed specification of the system requirements**
- **Essential** when an **outside contractor** is developing
- Agile methods claim that the document is out of date when it is written
- **Diverse set of users** (management to system developers) **require compromise**
- If the development is conducted by an **external company** detailed specification is necessary
- **In-house** developments may require **less detail**

Source: https://pixabay.com/en/flat-design-symbol-icon-www-2126884/
by janjf93 |CC0 Creative Commons

Requirements Specification: Challenges

Often it is difficult to exclude information because:

- You may have to design an **initial architecture to help structure the specification**
- The system must **incorporate with existing systems**
- You require a **specific architecture to satisfy non-functional requirements**

Note: While user requirements are usually written in natural language supplemented by diagrams, graphical models (e.g. UML) are useful when you need to show how a state changes or describe a sequence of actions

Source: https://pixabay.com/en/files-paper-office-paperwork-stack-1614223/
by myrfa |CC0 Creative Commons

Structured Natural language Specification

- **Natural language** has been used for writing requirements since the **beginning of software engineering**
- It is **expressive, intuitive, and universal**
- But also potentially **vague, ambiguous and depends on the background of the reader**
- **Structured Natural Language Specification reduces the freedom** of the writer by enforcing a standard way of expression
- E.g. **VOLERE requirements engineering (Robertson & Robertson, 1999)** method which uses **cards with specific fields to write down requirements** (i.e. one card per requirement)

Source: https://pixabay.com/en/stack-letters-letter-handwriting-447579/
by Andrys |CC0 Creative Commons

Requirements Engineering Process

Four high-level activities:

- Assessing whether system is useful to the business (**feasibility study**)
- Discovering requirements (**elicitation and analysis**)
- Converting these requirements into some standard form (**specification**)
- Checking that the system does what the customer wants (**validation**)

Early in the process most effort will be spent on **high-level business requirements, non-functional requirements and user requirements.**

Later more effort will be put into understanding the **detailed system requirements**.

Source: https://pixabay.com/en/machine-mechanical-eye-blue-look-1776925/
by infographics | CC0 Creative Commons

Requirements Elicitation and Analysis

Software engineers work with customers and end-users

- **Requirements discovery**
  Interact with stakeholders to discover potential requirements
- **Requirements classification and organization**
  Group and organize requirements into coherent clusters
- **Requirements prioritization and negotiation**
  Solve problems when stakeholders have conflicting requirements
- **Requirements specification**
  End of one round, start of the next iteration; formal and informal documents are produced

Source: https://pixabay.com/en/arrows-diagram-process-2027262/
by kmicican | CC0 Creative Commons

Requirements Discovery

- Gathering **information about the required system**
- Sources include **documentation, system stakeholders, and specifications** of similar systems
- You interact with stakeholders through **interviews** and **observations**
- You use **scenarios and prototypes** to help stakeholders understand what the system will be like
- Stakeholders: **end users, managers, external stakeholders**
- **Looking at other systems** interacting with the system helps establish **different viewpoints**

Source: https://pixabay.com/en/mindmap-brainstorm-idea-innovation-2123973/
by TeroVesalainen |CC0 Creative Commons

Interviewing

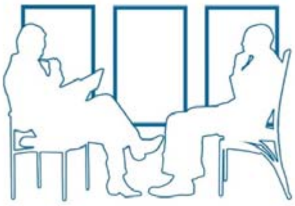
Formal and informal interviews are part of most requirements engineering processes

- **Closed interviews:** pre-defined set of questions
- **Open interviews:** no pre-defined agenda

Characteristics of good interviewers

- **Open minded** i.e. no pre-conceived idea and willing to change mind
- **Prompt interviewee** to get discussion going

Note: Interviews are liable to miss essential information hence they should be used in conjunction with other requirement elicitation techniques.



Source: https://pixabay.com/en/interview-superior-staff-2207741/
by geralt |CC0 Creative Commons

Scenarios

- Real-life examples are **easier to relate to**
- Scenarios are **easier to understand and consequently to criticize**
- Requirements engineers use them to **formulate system requirements**
- **Detail is added step by step**

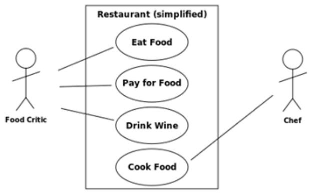
Scenarios may include

- A **description of what the system and users expect**
- A **description of the normal flow** of events
- A **description of what can go wrong**
- Information about **other activities**
- A description about **the system state** when it is finished

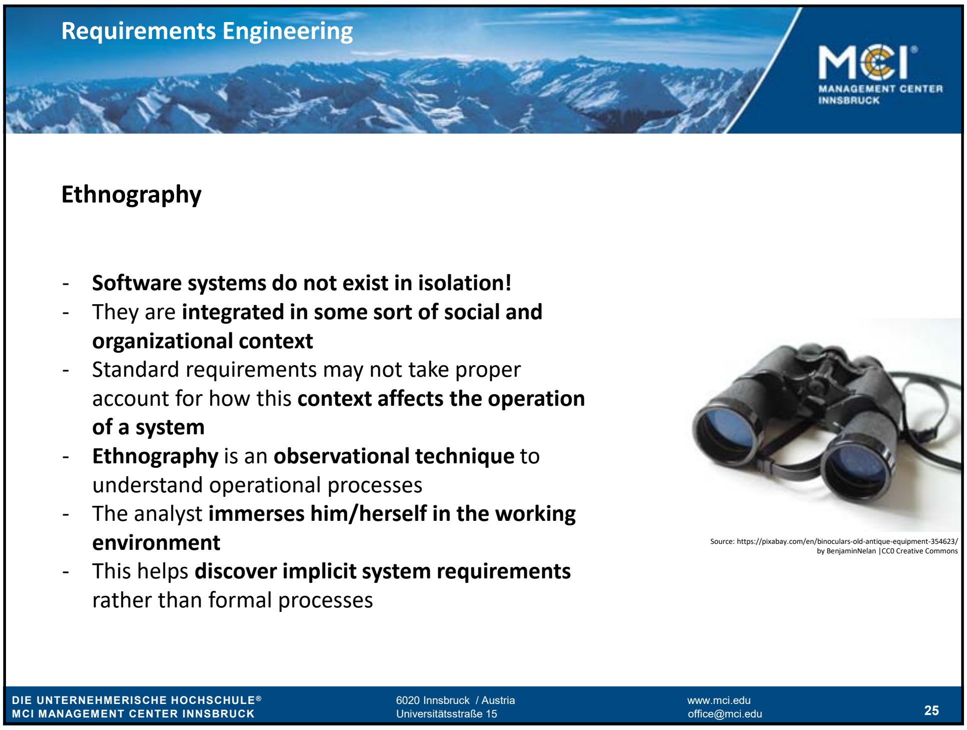
Source: https://pixabay.com/en/questions-who-what-how-why-where-2998901/
by geralt |CC0 Creative Commons

Use Cases

- **Requirements discovery technique**
- Fundamental feature of **UML**
- Simplest form identifies **actors** involved and **names the types of interaction**. This is **supplemented by additional information.**
- High **level use case diagrams** use **stick figures** to represent **actors**; each **class of interaction** is represented as a **named ellipse**
- There is **no clear distinction between scenarios and use cases**
- Some say that **a scenario is a single use case**



Source: https://upload.wikimedia.org/wikipedia/commons/7/71/UML_Use_Case_diagram.svg
by Slashme |CC0 Creative Commons



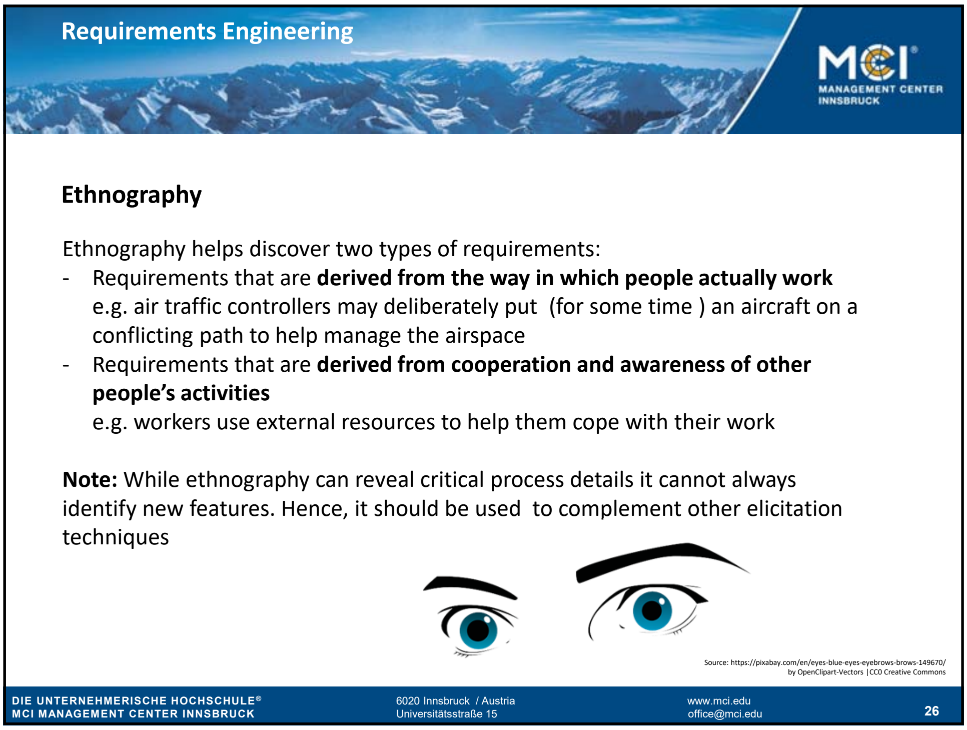
Requirements Engineering

MANAGEMENT CENTER INNSBRUCK

Ethnography

- **Software systems do not exist in isolation!**
- They are **integrated in some sort of social and organizational context**
- Standard requirements may not take proper account for how this **context affects the operation of a system**
- **Ethnography** is an **observational technique** to understand operational processes
- The analyst **immerses him/herself in the working environment**
- This helps **discover implicit system requirements** rather than formal processes

Source: https://pixabay.com/en/binoculars-old-antique-equipment-354623/
by BenjaminNelan |CC0 Creative Commons



Requirements Engineering

MANAGEMENT CENTER INNSBRUCK

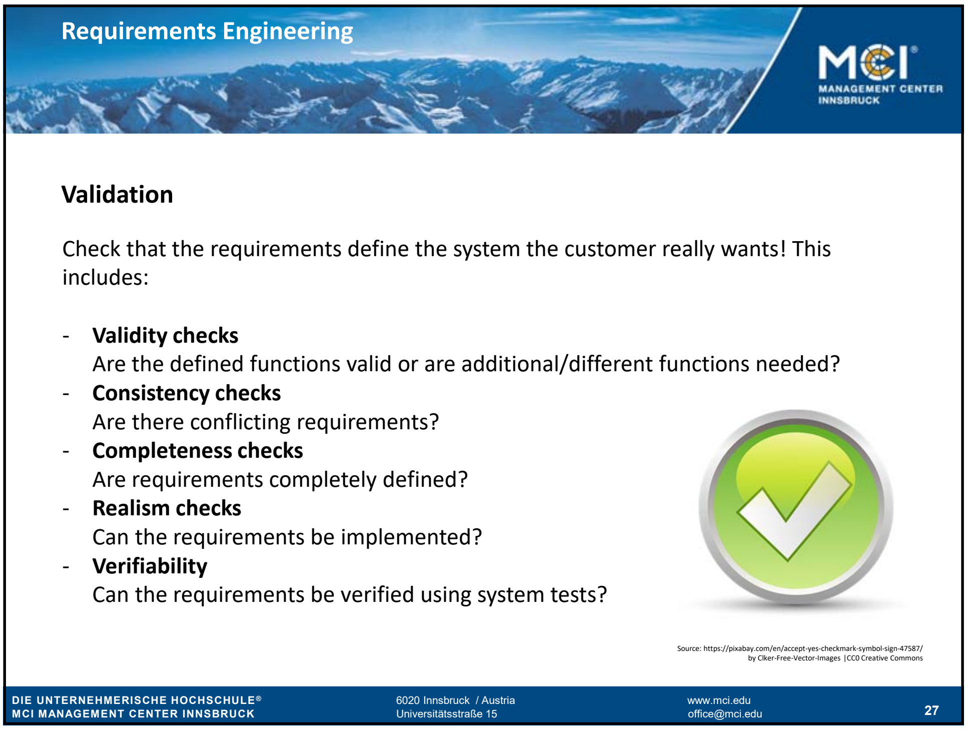
Ethnography

Ethnography helps discover two types of requirements:

- Requirements that are **derived from the way in which people actually work**
  e.g. air traffic controllers may deliberately put (for some time) an aircraft on a conflicting path to help manage the airspace
- Requirements that are **derived from cooperation and awareness of other people's activities**
  e.g. workers use external resources to help them cope with their work

Note: While ethnography can reveal critical process details it cannot always identify new features. Hence, it should be used to complement other elicitation techniques

Source: https://pixabay.com/en/eyes-blue-eyes-eyebrows-brows-149670/
by OpenClipart-Vectors |CC0 Creative Commons



Requirements Engineering

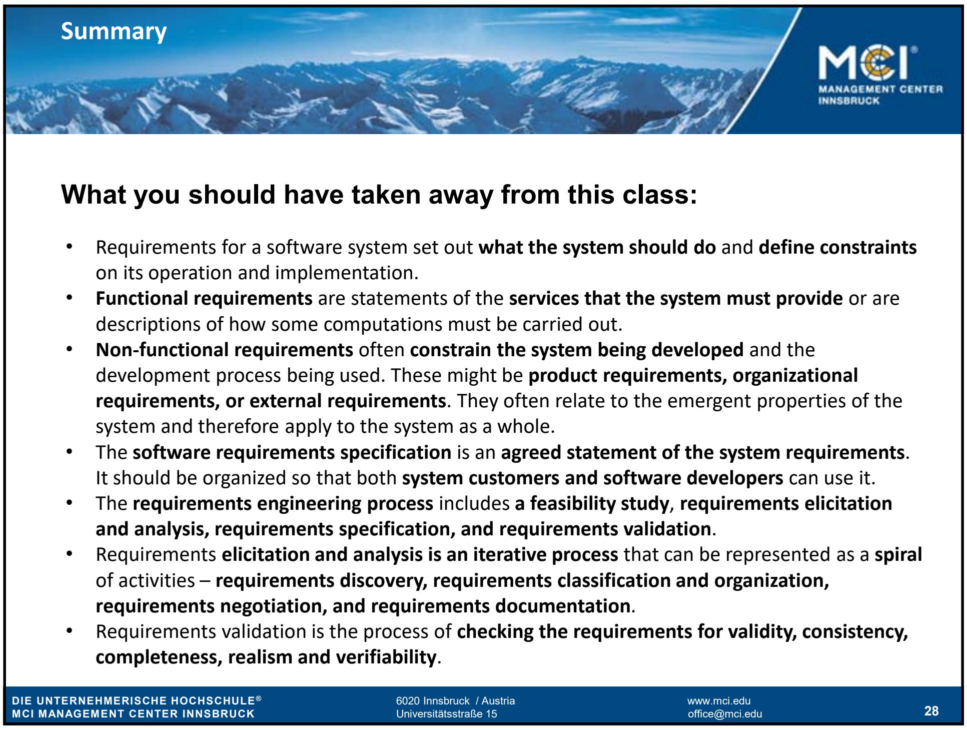
MCI MANAGEMENT CENTER INNSBRUCK

Validation

Check that the requirements define the system the customer really wants! This includes:

- **Validity checks**
  Are the defined functions valid or are additional/different functions needed?
- **Consistency checks**
  Are there conflicting requirements?
- **Completeness checks**
  Are requirements completely defined?
- **Realism checks**
  Can the requirements be implemented?
- **Verifiability**
  Can the requirements be verified using system tests?

Source: https://pixabay.com/en/accept-yes-checkmark-symbol-sign-47587/
by Clker-Free-Vector-Images |CC0 Creative Commons

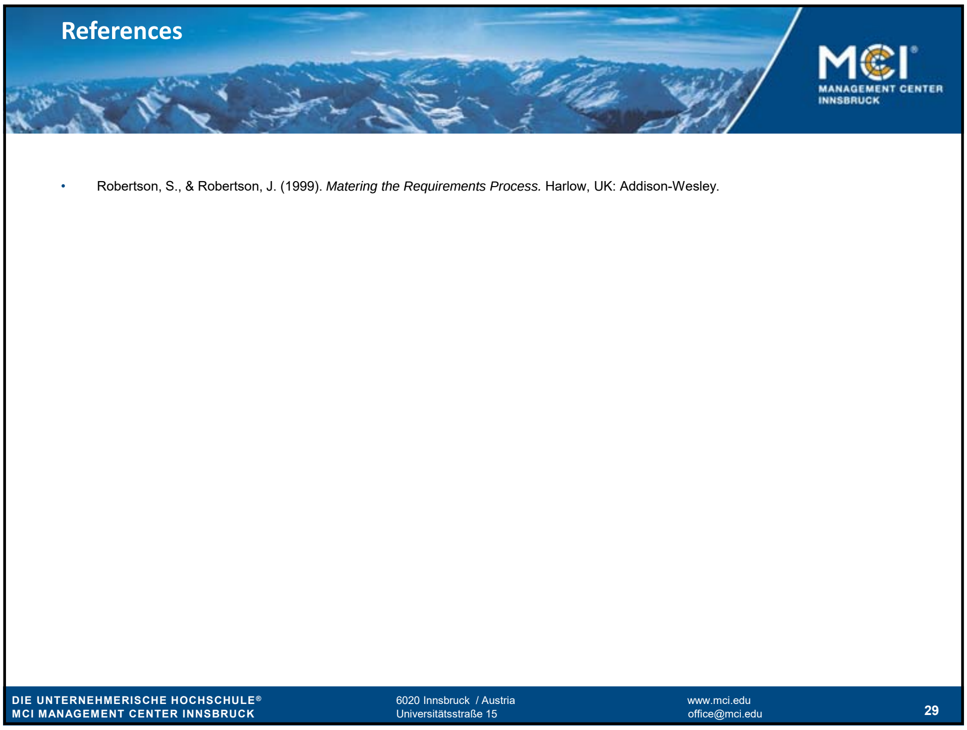


Summary

MCI MANAGEMENT CENTER INNSBRUCK

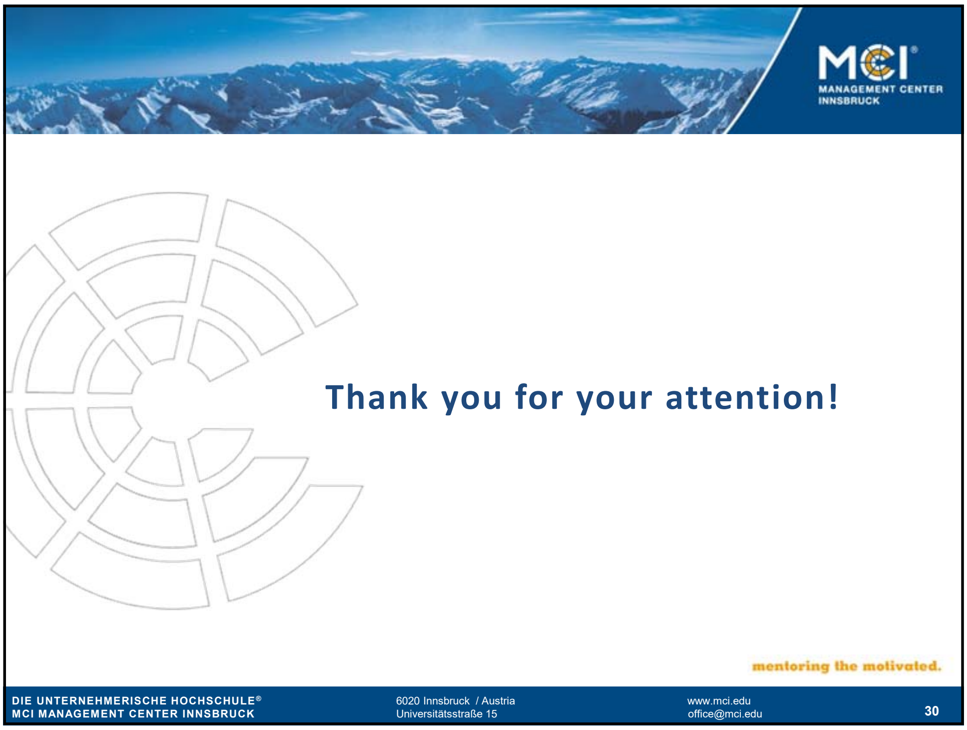
What you should have taken away from this class:

- 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. These might be **product requirements, organizational requirements, or external requirements**. They often relate to the emergent properties of the system and therefore apply to the system as a whole.
- The **software requirements specification** is an **agreed statement of the system requirements**. It should be organized so that both **system customers and software developers** can use it.
- The **requirements engineering process** includes **a feasibility study**, **requirements elicitation and analysis, requirements specification, and requirements validation**.
- Requirements **elicitation and analysis is an iterative process** that can be represented as a **spiral** of activities – **requirements discovery, requirements classification and organization, requirements negotiation, and requirements documentation**.
- Requirements validation is the process of **checking the requirements for validity, consistency, completeness, realism and verifiability**.



References

- Robertson, S., & Robertson, J. (1999). *Matering the Requirements Process.* Harlow, UK: Addison-Wesley.



Thank you for your attention!

mentoring the motivated.