Dr. Michael Eichberg
Software Engineering
Department of Computer Science
Technische Universität Darmstadt

Software Engineering

Requirements Engineering

- The following slides are primarily based on the contents of the following books:
    - Applying UML and Patterns; Craig Larman;
    - Software Engineering; Ian Sommerville

TECHNISCHE UNIVERSITÄT DARMSTADT

Requirements Engineering Using Natural Language

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Statement:

Mary had a little lamb.

Meaning:

?

to have

1. to hold in possession as a property
2. to trick or fool someone (been had by a partner)
3. to beget or bear (have a baby)
4. to partake (have as a dinner)
5. …

Statement:

Mary had a little lamb.

Meaning:

?

a lamb

1. a young sheep less than one year
2. the young of various other animals (antelope etc.)
3. a person as gentle or weak as a lamb
4. a person easily cheated or deceived
5. the flesh of lamb used as food
6. ...

Statement:

Mary had a little lamb.

Meaning:

?

Statement:

Mary had a little lamb.

Meaning:

?

have	lamb	meaning
1	1	Mary owned a little sheep under one year...
3	2	Mary gave birth to an antelope.
…	…	…

Statement:

Shoes must be worn.

Meaning:

?

Requirements Engineering

- Brief Introduction

Requirements are the descriptions of

- **the services provided by the system** and
- **the operational constraints**.

W.r.t. the level of description, we can distinguish two types of requirements: (A) **User** and (B) System **requirements**.

- User requirements
  They state - in natural languages plus diagrams - what services the system is expected to provide and the constraints under which it must operate.
  Usually written (stated) by the customer.
  (dt. Grundlage für das „Lastenheft“)
- …

W.r.t. the level of description, we can distinguish two types of requirements: (A) User and (B) **System requirements**.

- …
- **System requirements** (dt. Systemanforderungen)
  Set out the system's functions, services and operational constraints in detail.
  Often these requirements are written down in the system requirements document (also called a functional specification) which is typically part of the contract. Hence, the requirements should be precise.
  Written by the software developer/contractor.
  (dt. Grundlage für das „Pflichtenheft")

User Requirement Definition(s)

Exemplified

- The Library system shall keep track of all data required by copyright licensing agencies in the UK and elsewhere.
- …

System Requirement Definition(s)
Exemplified

- On making a request for a document from the library system, the requestor shall be presented with a form that records details of the user and the request made.
- The library system's request forms shall be stored on the system for five years from the date of the request.
- All library system request forms must be indexed by user, by the name of the material requested and by the supplier of the request.
- The library system shall maintain a log of all requests that have been made to the system.
- …

From the point-of-view of a developer, we can distinguish (A) **Functional** and (B) Non-functional **requirements**.

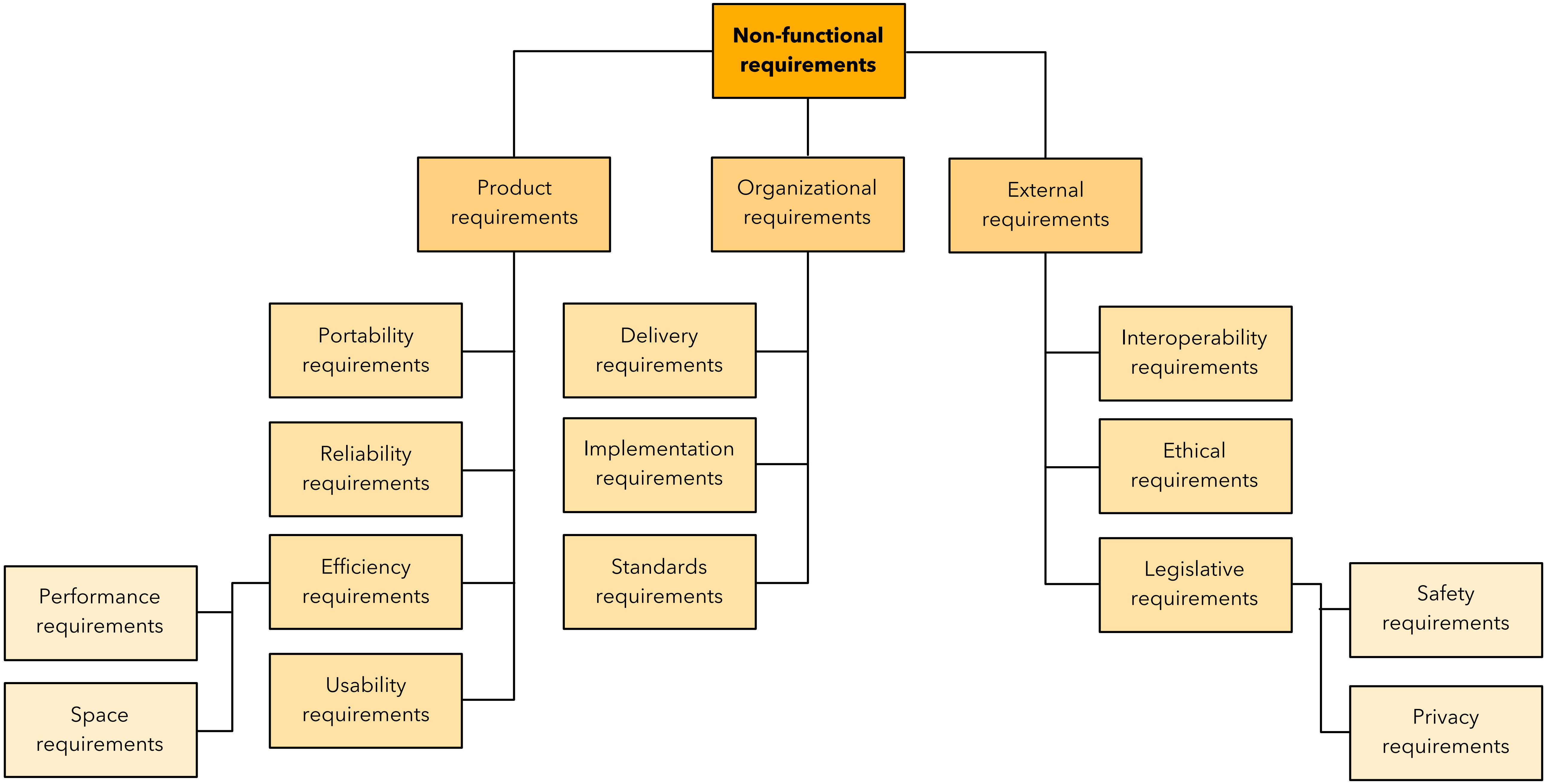
- Functional requirements
  They specify the services that the system should provide, how the system should (not) react to particular inputs and how the system should (not) behave in particular situations.
  To fulfill these requirements it is usually necessary to “write code”.

From the point-of-view of a developer, we can distinguish (A) Functional and (B) **Non-functional requirements**.

- Non-functional requirements
  Constraints on the services or functions offered by the system; including: timing constraints, constraints on the development process and standards.
  They often apply to the system as a whole.
  It is usually not directly possible to write some well-defined piece of code to fulfill these requirements. However, it is sometimes possible to write tests/setup test systems to test that the requirements are satisfied!

The boundaries between functional and non-functional requirements are not always clear-cut.
If you take a more detailed look on a non-functional requirement (e.g. “the system has to be secure”) it might result in the identification of functional requirements.

“Some” Types of Non-functional Requirements



Ian Sommerville - Software Engineering 8; Addison Wesley 2007

Organizational Requirements

- Delivery requirements, e.g.,
  - how the product is delivered (online, packaged,…) and how it can be used: “only” online or off- and online
- Implementation requirements, e.g.,
  - the programming language that has to be used
  - the tools/platform that has to be used
- Standards requirements, e.g.,
  - the development process that has to be used

Examples

External Requirements

- Requirements that are external to the system, e.g.,
  - legislative requirements, e.g.,
    - the usage of certain images may not be allowed in certain regions/may require an age restriction
    - how the data is stored and archived
    - regarding the protection of sensitive information

Examples

Product Requirements

- Portability, e.g.,
  - it can be used on Linux, Mac,…
  - it is compatible with all Android Phones which have a resolution of at least W x H pixels
- Reliability, e.g.,
  - the probability that the software will work properly in a specified environment (the latter has to be well defined!)
- Efficiency requirement
  - it must not use more than x MB Ram
  - it must complete the computation in x seconds

Examples

Often non-functional requirements are more critical than individual functional requirements.

(E.g. a banking system that does not support the export of the bank statement as PDF is probably still useable; if the system is not secure, it is worthless).

Often non-functional requirements are more critical than individual functional requirements.
(E.g. a banking system that does not support the export of the bank statement as PDF is probably still useable; if the system is not secure, it is worthless).
But, how to evaluate if a non-functional requirement is met?

(Let's assume that we are going to develop a new web shop)

How about the following non-functional requirement:

The user interface should be easy to use.

?

not very useful

(Let’s assume that we are going to develop a new web shop)

How about the following non-functional requirement:

~~The user interface should be easy to use.~~

?

- Let's assume that we are going to develop a new web shop
  How about the following non-functional requirement:
  - The number of forms that fail server-side validation due to input errors should be less than Y percent.
  - An average user[1] should be able to make an order in less than X minutes.
  - The number of not completed transactions should not exceed Z percent.
  - After one day of training an agent should be able to handle twice as many orders.

These requirements could be an indirect measure of a user interface's quality.

[1] "The average user" is defined elsewhere.

Domain requirements are derived from the application domain of the system rather from the needs of the system users.

- Usually **expressed using domain-specific terminology**; hard to understand by software engineers
- **May not be explicitly stated** since they are obvious to the domain expert
- Can be functional or non-functional

The “Software Requirements Document” states what the developers should implement.

(Software Requirements Document ~dt. Pflichtenheft)

- The document has a diverse set of users/stakeholders:
    - System customers
    - Managers
    - System engineers
    - System test engineers
    - System maintenance engineers
- …

The "Software Requirements Document" states what the developers should implement.

(Software Requirements Document ~dt. Pflichtenheft)

- ...
- The level of detail depends on:
  - The type of system
  - The development process that is used
  - Where the system is build: external contractor or in-house

The IEEE/ANSI 830/1998 Standard for Structuring a Requirements Document

1. **Introduction**
   a. Purpose of the requirements document
   b. Scope of the product
   c. Definitions, acronyms and abbreviations
   d. References
   e. Overview
2. **General description**
   a. Product perspective
   b. Product functions
   c. User characteristics
   d. General constraints
   e. Assumptions and dependencies
3. **Specific requirement**
4. **Appendices**
5. **Index**

This can include a specification of the functionality that is not in the scope.

Statement:

Mary had a little lamb.

Meaning:

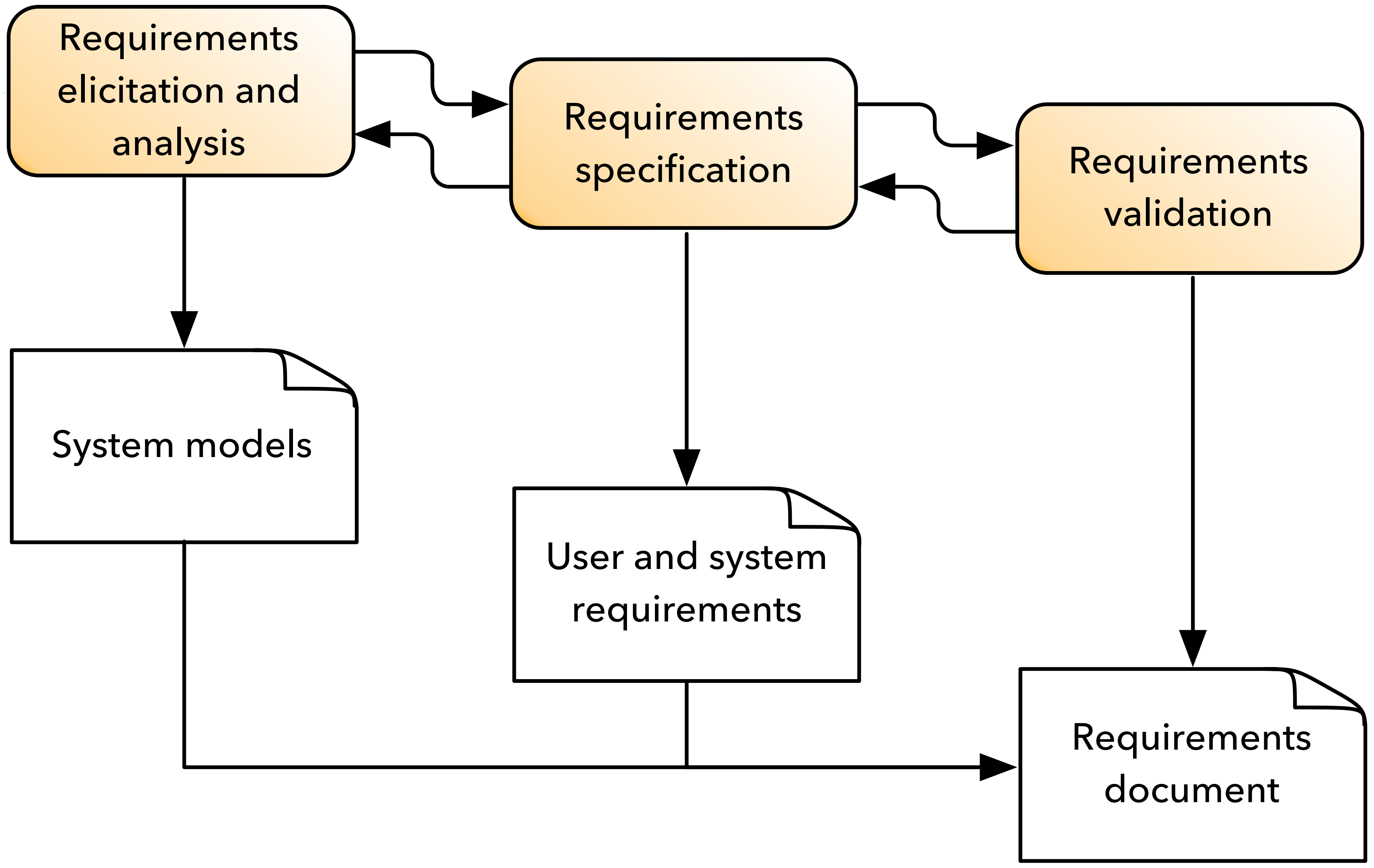
?

have	lamb	meaning
1	1	Mary owned a little sheep under one year...
3	2	Mary gave birth to an antelope.
...	...	...

That's why the system requirements specification should have a glossary.

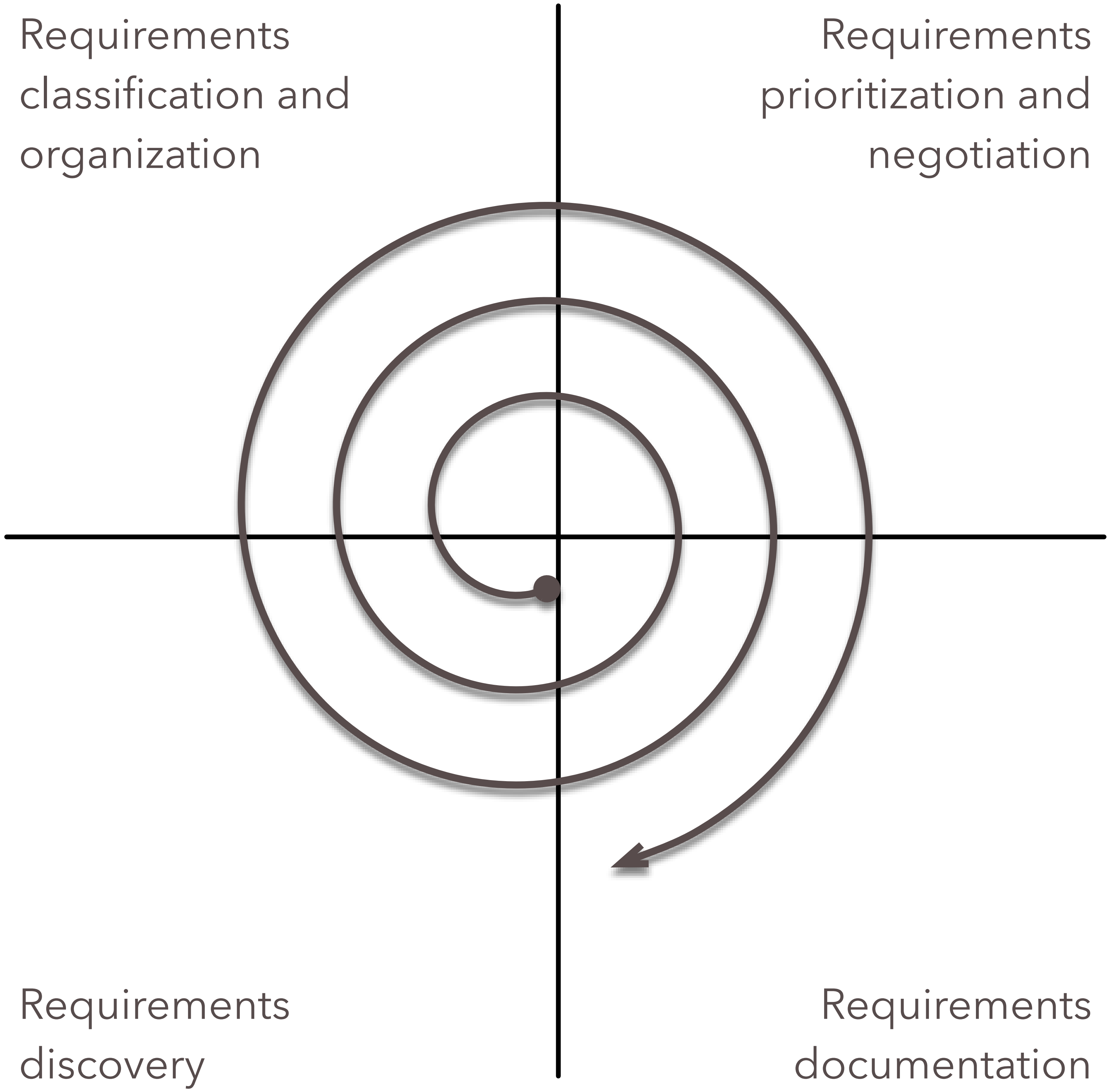
- Requirements are the descriptions of the services provided by the system and the operational constraints Requirements are described in the system requirements specification.
- Requirements engineering is the process of:
  - *finding out,*
  - *analyzing*,
  - *documenting* and
  - *checking* theses services and constraints.
- The system requirements document is created and maintained during requirements engineering

The Requirements Engineering Process


Requirements elicitation and analysis
Requirements specification
Requirements validation
System models
User and system requirements
Requirements document

The Requirements Elicitation and Analysis Process

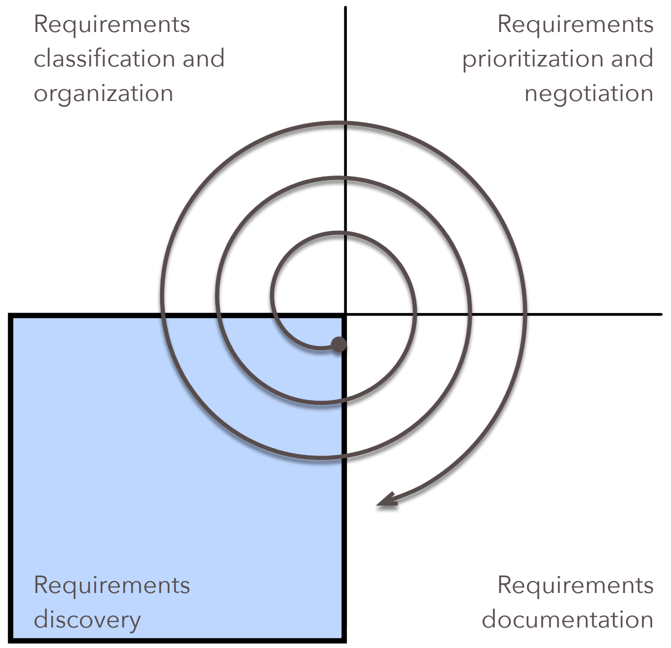
(Requirements Elicitation =dt. Anforderungsermittlung)


Requirements classification and organization
Requirements prioritization and negotiation
Requirements discovery
Requirements documentation

Requirements discovery is the process of interacting with stakeholders in the system to collect their requirements.

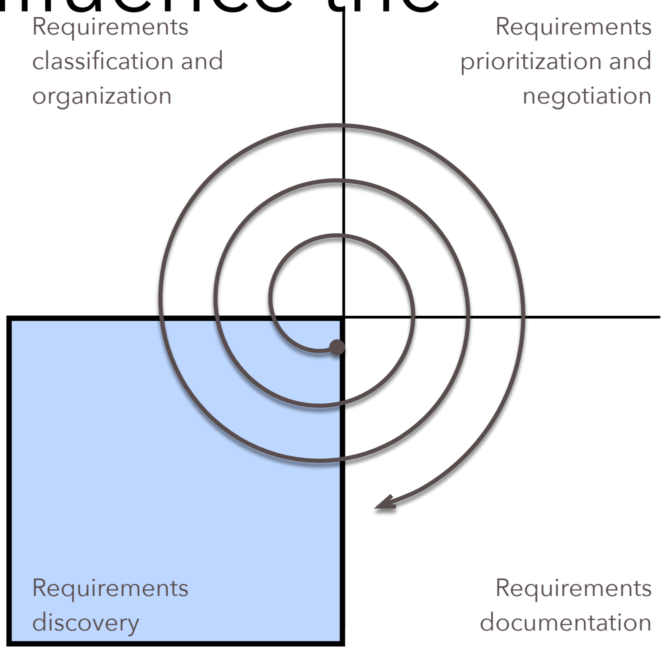
- Major problem:
  How to systematically discover requirements?

- An approach:
  Viewpoint-oriented approaches.


Requirements classification and organization
Requirements prioritization and negotiation
Requirements discovery
Requirements documentation

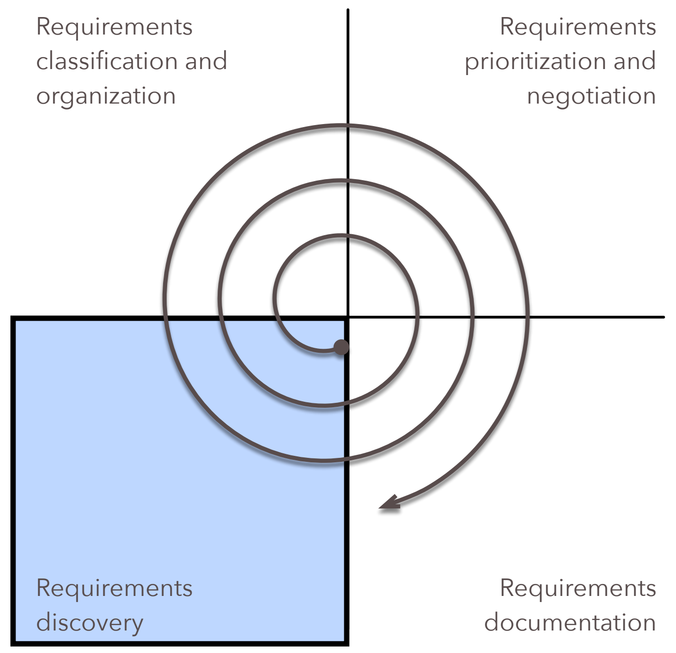
Viewpoint-oriented approaches

- Generic types of viewpoints:
  - **Interactor viewpoints**
    People that will interact with the system.
  - **Indirect viewpoints**
    Stakeholders that influence the requirements, but who will not directly use the system.
  - **Domain viewpoints**
    Domain characteristics and constraints that influence the system requirements.
- …


Requirements classification and organization
Requirements prioritization and negotiation
Requirements discovery
Requirements documentation

Viewpoint-oriented approaches

- …
- During the elicitation you should try to identify more specific viewpoints.
- After discovering the most important viewpoints start with them to discover the requirements.


Requirements classification and organization
Requirements prioritization and negotiation
Requirements discovery
Requirements documentation

Techniques for Requirements Elicitation

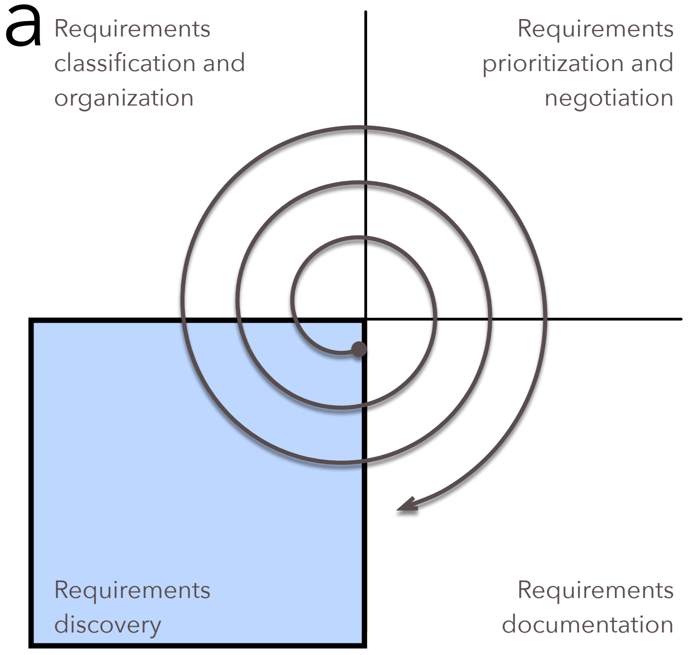
- **Interviews**

  Interviews should only be used alongside other requirements techniques, because interviewees use domain knowledge the interviewer may not be familiar with, are reluctant to reveal the actual structure, and may even work against the project if (they think) their job is at stake.

  Types of interviews:

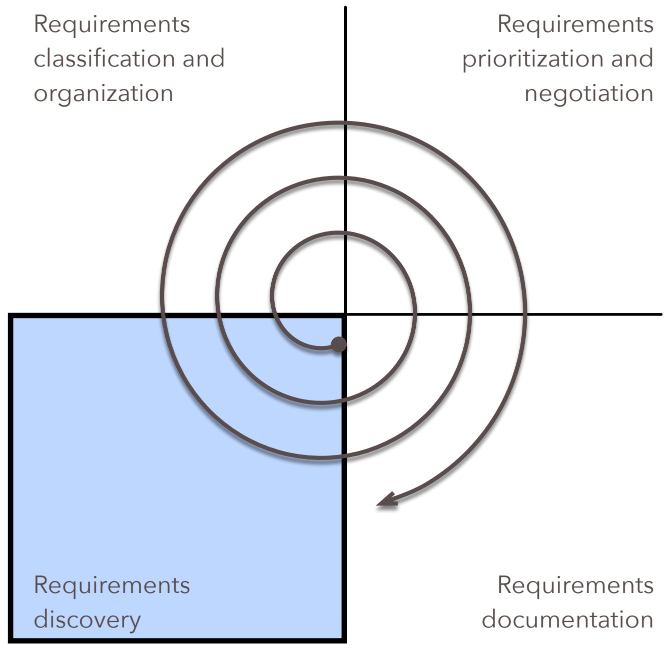
  - **Closed interviews** where the stakeholder answers a predefined set of questions
  - **Open interviews** with no predefined agenda

- …


Requirements classification and organization
Requirements prioritization and negotiation
Requirements discovery
Requirements documentation

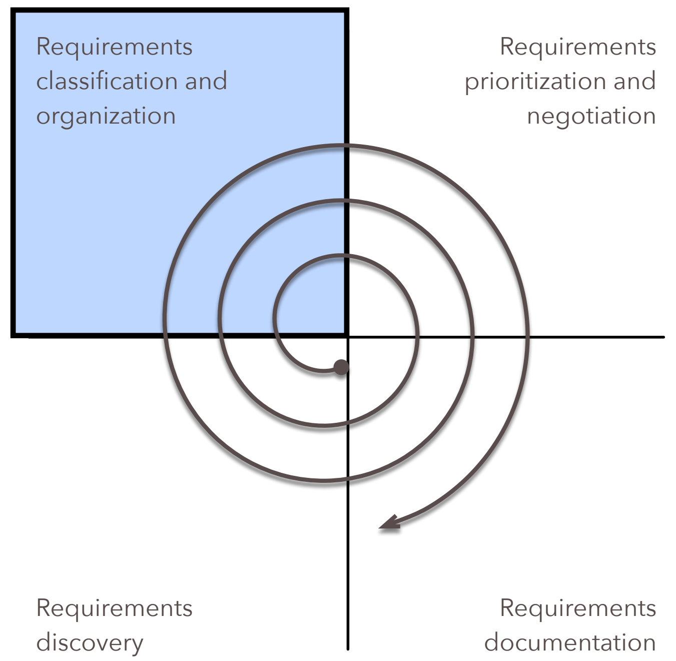
Techniques for Requirements Elicitation

- …
- **Scenarios**
  Scenarios cover possible interactions with the system. The interactions are roughly outlined at the beginning and are detailed during the elicitation.
  Most people can understand and critique a scenario of how they might interact with the system. Scenarios are particularly useful for adding detail to an outline requirements description.
- *Use cases (will be covered later on)*
- …


Requirements classification and organization
Requirements prioritization and negotiation
Requirements discovery
Requirements documentation

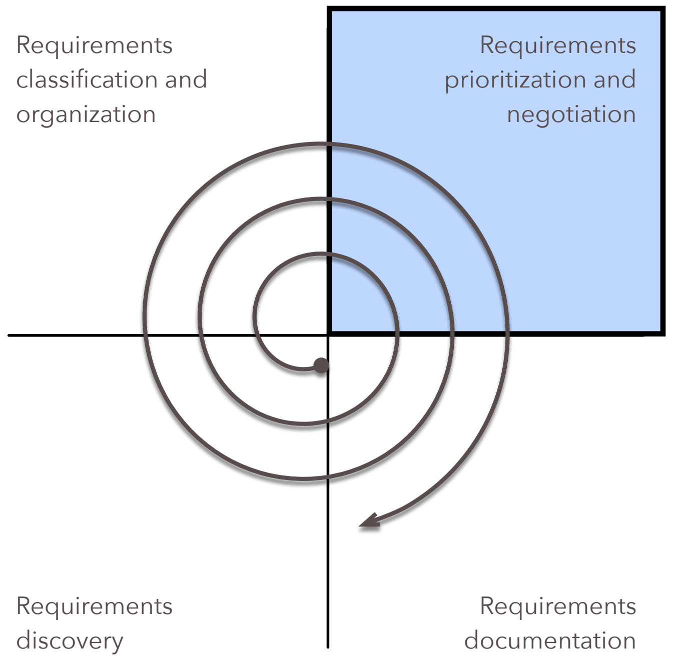
Requirements classification and organization

- Given the unstructured collection of requirements, the requirements are grouped and organized into coherent clusters
- A possible model for categorizing the requirements is the FURPS+ Model:
  - **F**unctional
  - **U**sability
  - **R**eliability
  - **P**erformance
  - **S**upportability
  - +
    - Implementation
    - Interface
    - Operations
    - Packaging
    - Legal


Requirements classification and organization
Requirements prioritization and negotiation
Requirements discovery
Requirements documentation

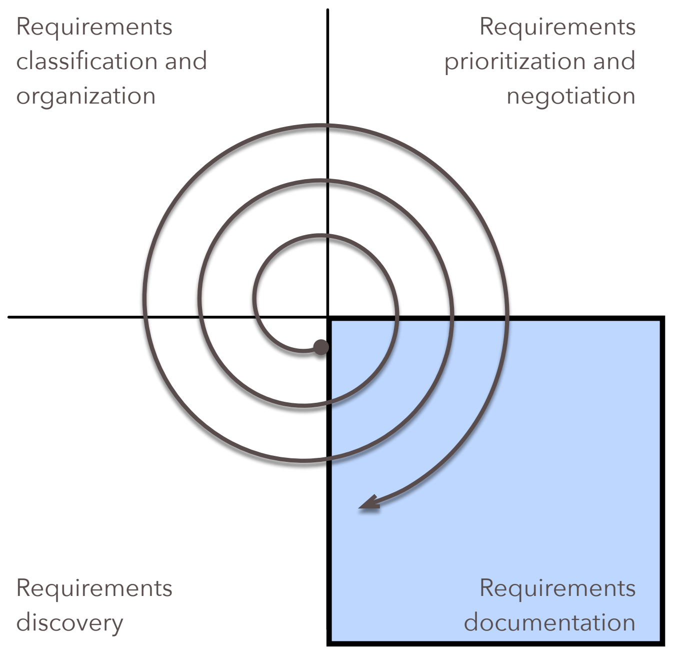
Requirements prioritization and negotiation

- The requirements are prioritized and conflicts are found and resolved through negotiation


Requirements classification and organization
Requirements prioritization and negotiation
Requirements discovery
Requirements documentation

Requirements documentation

- The requirements are documented and used as input for the next round in the spiral
  (The produced documents may be formal or informal.)


Requirements classification and organization
Requirements prioritization and negotiation
Requirements discovery
Requirements documentation

Requirements validation is concerned with showing that the requirements actually define the system that the customer wants.

Requirements validation tries to find problems with the requirements.

- Checks that are carried out during requirements validation:
  - **Validity checks**
    Do the requirements capture the right functions; do we need additional or other functionality?
  - **Consistency checks**
    Check that the requirements are not conflicting.
  - **Completeness checks**
    Do the requirements define all functions and constraints as intended by the system user?
  - …

Requirements validation is concerned with showing that the requirements actually define the system that the customer wants.

Requirements validation tries to find problems with the requirements.

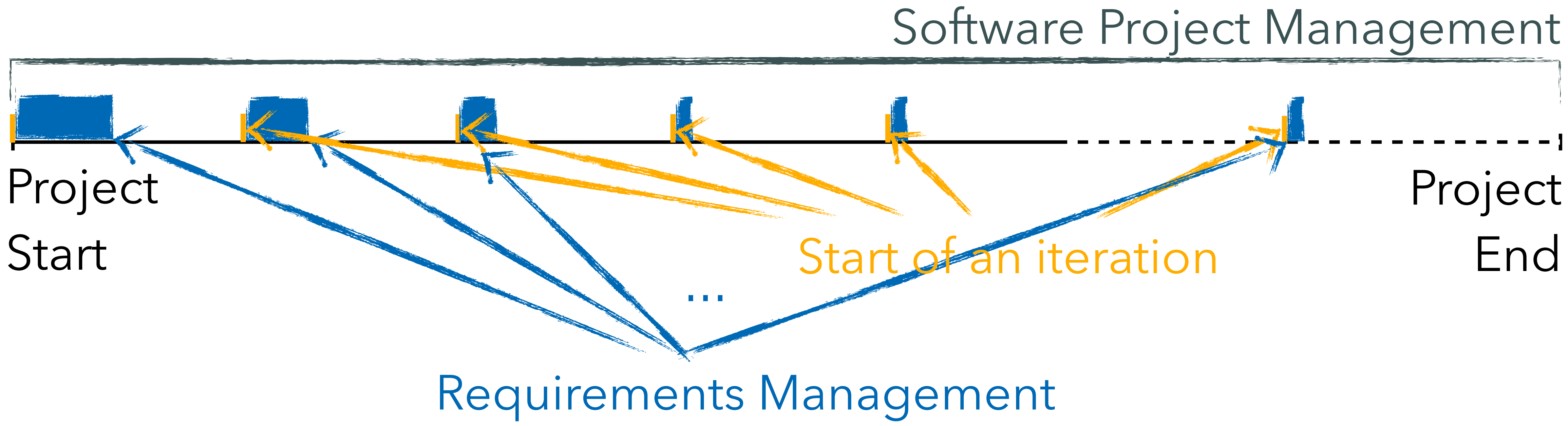
- Checks that are carried out during requirements validation:
  - …
  - **Realism check**
    Can the requirements reasonably be implemented?
  - **Verifiability**
    Is it possible to develop a test that checks if a requirement is fulfilled?
  - **Traceability**
    Is each requirement traceable to its source (where does the requirement come from)?

Requirements Engineering Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- Different types of requirements:
  - functional and non-functional requirements
  - user and system requirements
- Requirements Engineering Process

- The goal of this lecture is to enable you to systematically carry out small(er) commercial or open-source projects.


Software Project Management
Project Start
Project End
Start of an iteration
...
Requirements Management