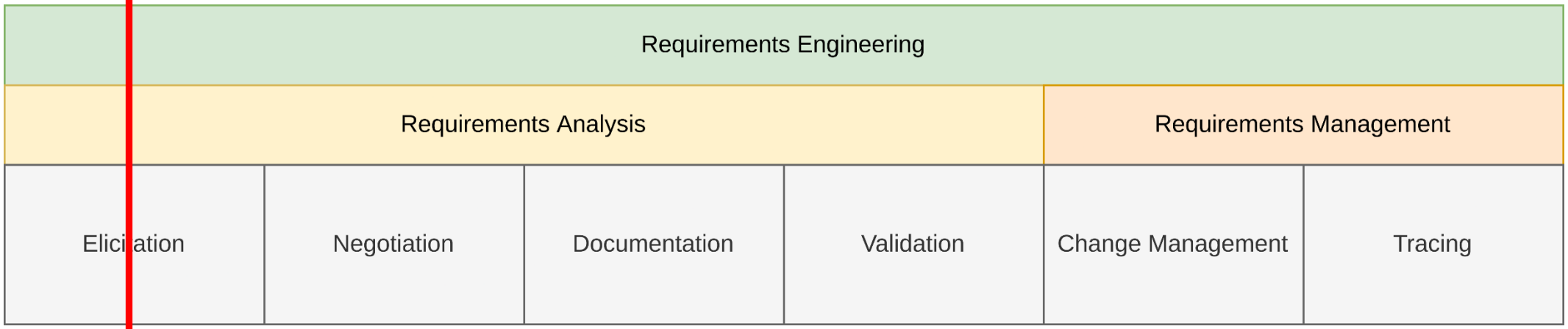
Requirement Engineering

Lecture 2: System Context Boundaries and Types of Requirements

Prof. Dr. Benjamin Leiding
M.Sc. Anant Sujatanagarjuna

General Requirements Engineering Process

Overview



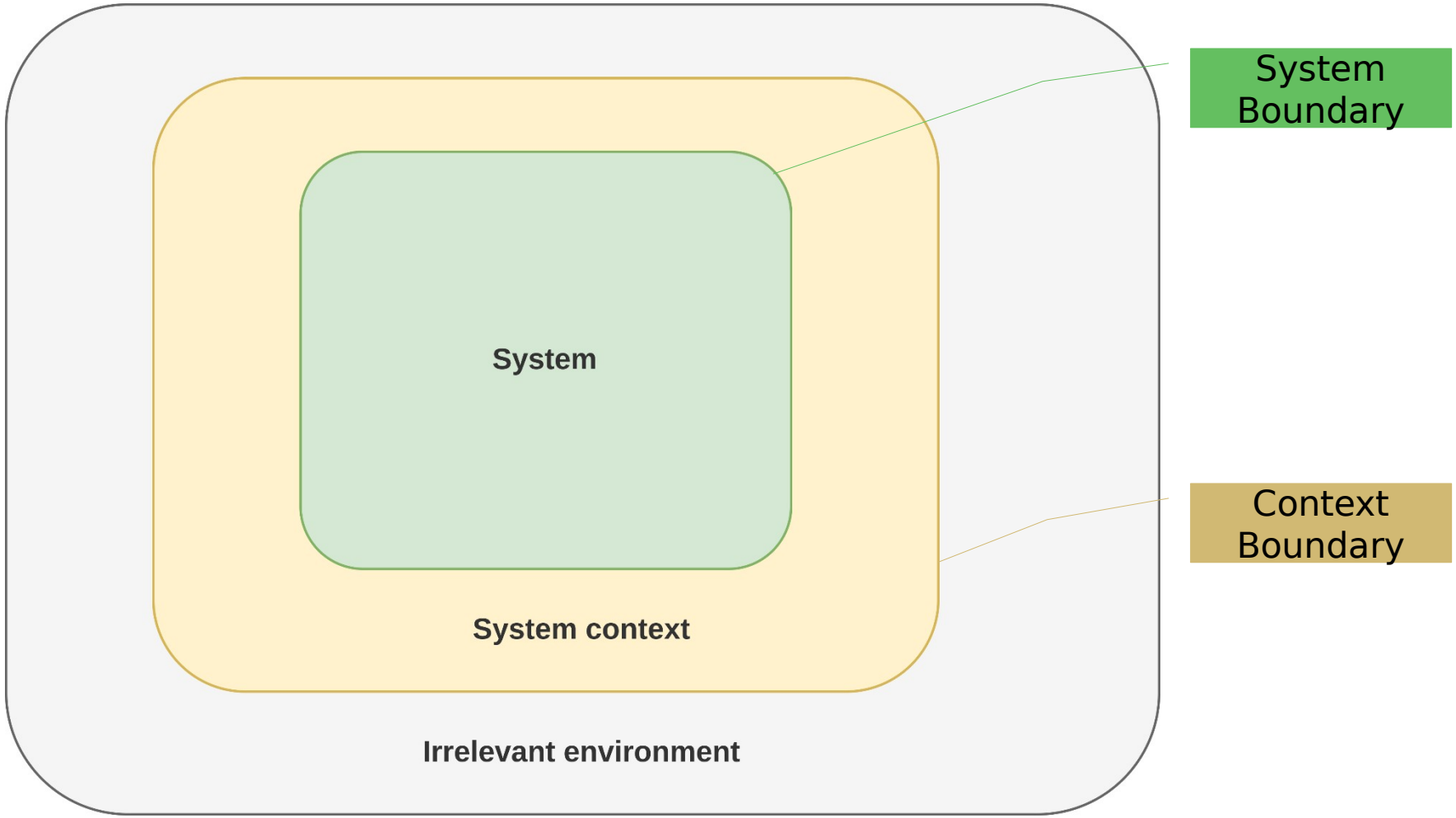
Lecture 2: System Context Boundaries

Content

1. System Context
2. System Boundary
3. Context Boundary
4. Types of Requirements

SYSTEM CONTEXT

System Context



Based on K. Pohl, C. Rupp (2011) – Requirements Engineering Fundamentals: A study guide for Requirements Engineering Foundation Level

System Context

Why?

- Software system is always embedded in an environment
- Environment influences definition of requirements
- Environment might consist of, e.g., (im)material objects ((non-)technical systems), people, technologies, business processes, laws, sensors, existing software components, etc.

→ Ignoring the environment will most likely lead to defects in the requirement specification

System Context

Some Examples for Limiting Factors

- Developers only trained in Java
  - Excludes other language options
- Input data comes from a publicly available database
  - Excludes arbitrary input formats
- eXtreme Programming should be used
  - Can only use technologies for which the required tooling is available, e.g., automated unit tests
- An existing library should be re-used
  - Excludes other libraries that provide similar features
  - Requires development that is compatible to the available library

System Context

Wrong Context – so what?

- In extreme cases:
  - Critical bugs that prevent the system execution
  - System may not be deployable
  - The implementation may be impossible
- Less extreme cases:
  - Late changes to requirements increase costs
  - Removal of features
  - Uncritical bugs
  - Users just don’t like the software

System Context

Wrong Context – so what?

- In extreme cases:
  - Critical bugs that prevent the system execution
  - System may not be deployable
  - The implementation may be impossible
- Less extreme cases:
  - Late changes to requirements increase costs
  - Removal of features
  - Uncritical bugs
  - Users just don’t like the software

Wrong Context → Wrong Requirements

System Context

Definition

"The system context is the part of the system environment relevant for defining, understanding, and interpreting the system requirements. The system context consists of four context facets: the subject facet, the usage facet, the IT system facet, and the development facet."

K. Pohl, C. Rupp (2011) – Requirements Engineering Fundamentals: A study guide for Requirements Engineering Foundation Level

System Context

Context as Origin of Requirements

- Without knowing the context, requirements cannot be defined properly
  - Otherwise, requirements may be outside of the context
- Example:
  - Medical applications require the fulfillment of medical software standards
    - The standards are part of the context
    - The standards may prohibit certain requirements, that contradict the standard
      - e.g., publish data
    - The standards may lead to requirements, in order to fulfill it
      - e.g., pseudonymize data

System Context

Understanding the Context is Important

- Understanding the context leads to understanding the requirements
  - Some requirements cannot be understood without the context
    - Example:
      - The sending of E-Mails must be according to RFC 821. ← Defines SMTP
      - Without knowledge about RFC 821 not understandable
- The “why” of requirements often originates from the context
  - Example:
    - All documents must be encrypted with AES-256.
    - Usually this is unreasonable
    - If the documents are classified and you know that AES-256 is NSA approved, this makes sense

System Context

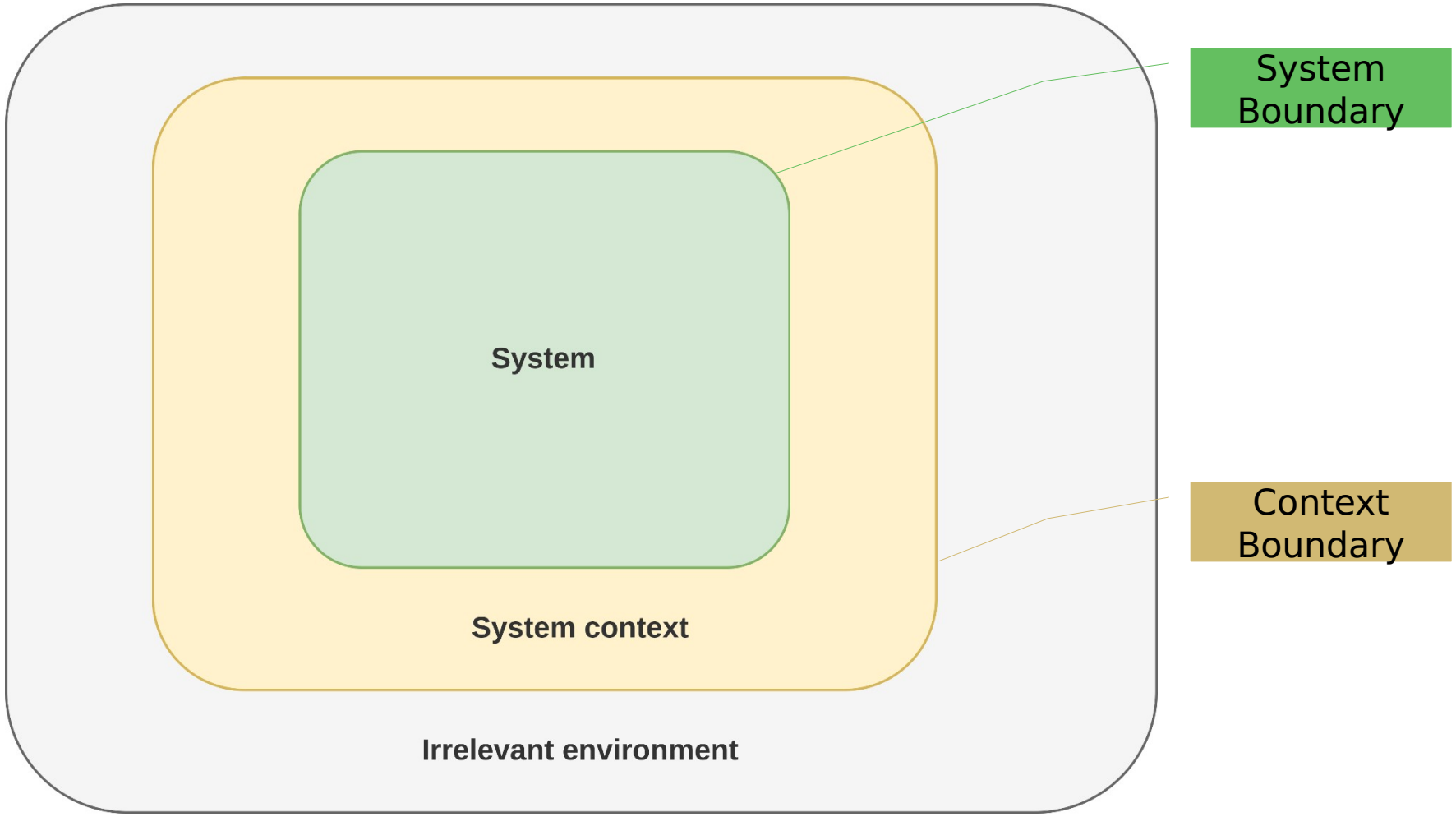
Defining the System and Context Boundaries

- Responsibility of the requirements engineer

- Both system boundary and context boundary must be defined
  - System boundary:
    - Which aspects belong to the system and which aspects belong to the system context?
  - Context boundary:
    - Which aspects belong to the system context and which aspects are irrelevant (i.e., not in the system context)?

SYSTEM BOUNDARY

System Boundary



Based on K. Pohl, C. Rupp (2011) – Requirements Engineering Fundamentals: A study guide for Requirements Engineering Foundation Level

System Boundary

Overview

- The system boundary defines the scope of the development

- Example:
  - If a credit card payment system is developed as part of the system, it is within the system boundary
  - If an existing credit card payment system is reused, it is outside of the system boundary

System Boundary

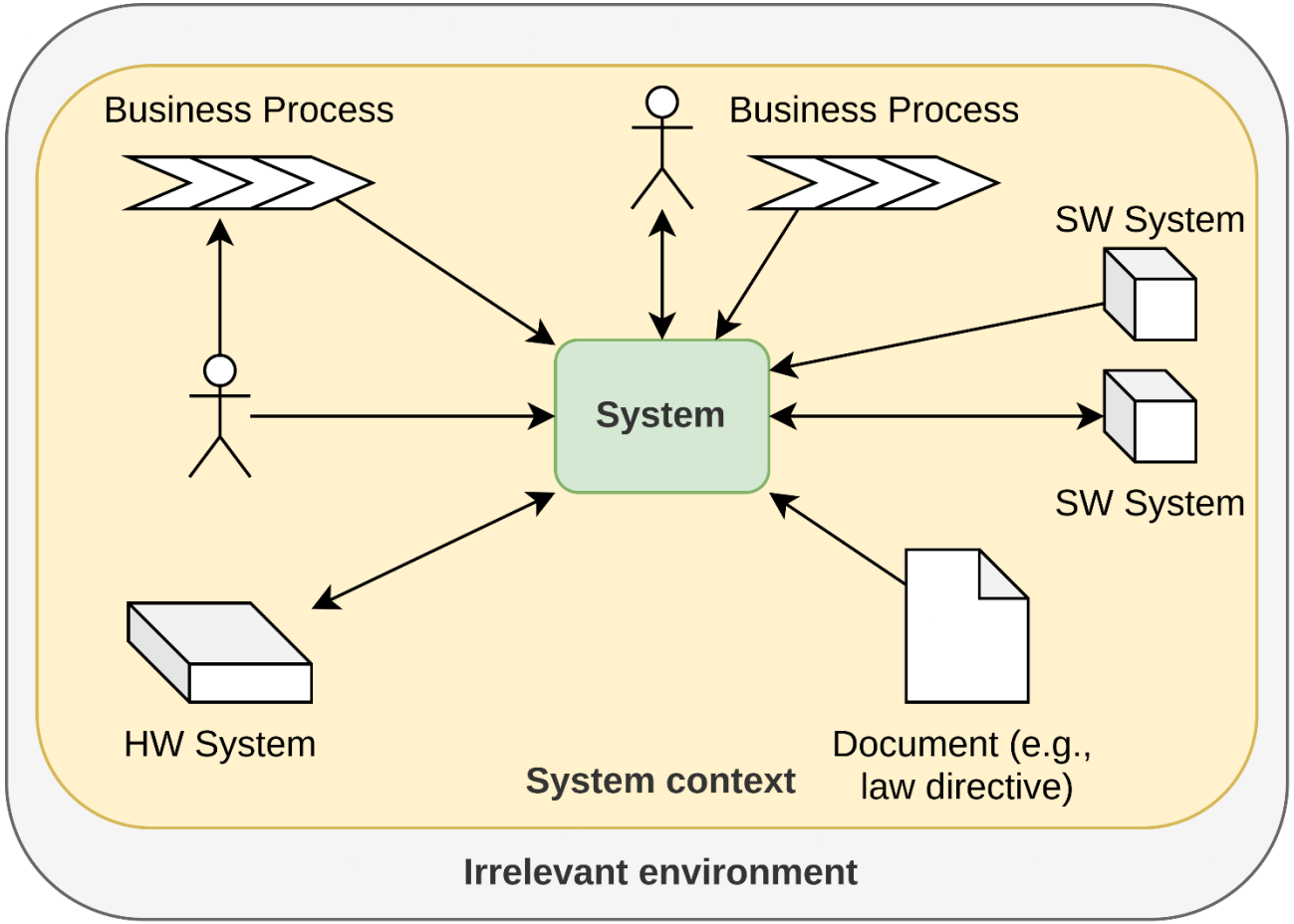
Definition

"The system boundary separates the system to be developed from the system context. The system boundary separates the parts that belong to the system and can hence be changed during the development process from the parts of the system context that cannot be changed during the development process."

K. Pohl, C. Rupp (2011) – Requirements Engineering Fundamentals: A study guide for Requirements Engineering Foundation Level

System Boundary

Influences on the System Boundary



Based on K. Pohl, C. Rupp (2011) – Requirements Engineering Fundamentals: A study guide for Requirements Engineering Foundation Level

System Boundary

Interfaces of the System

- Two types of interfaces
  - Sources provide inputs for the system
  - Sinks receive output from the system
- Possible sources and sinks:
  - (Groups of) Stakeholders
  - Existing Systems
- Interfaces are used ...
  - for monitoring the environment
  - to provide functionalities to the environment
  - to influence parameters of the environment
  - to control operations of the environment

System Boundary

Interface Types

- Sources and Sinks require various interfaces
  - Depends on the functionality of the source/sink
- Common examples:
  - Human-machine interfaces
    - Mouse, keyboard, touchscreen, emergency off switch
  - Hardware interface
    - SD card slot, USB port
  - Software interface
    - Library, Web service
- The interfaces may impose constraints
- The interfaces may be sources of requirements

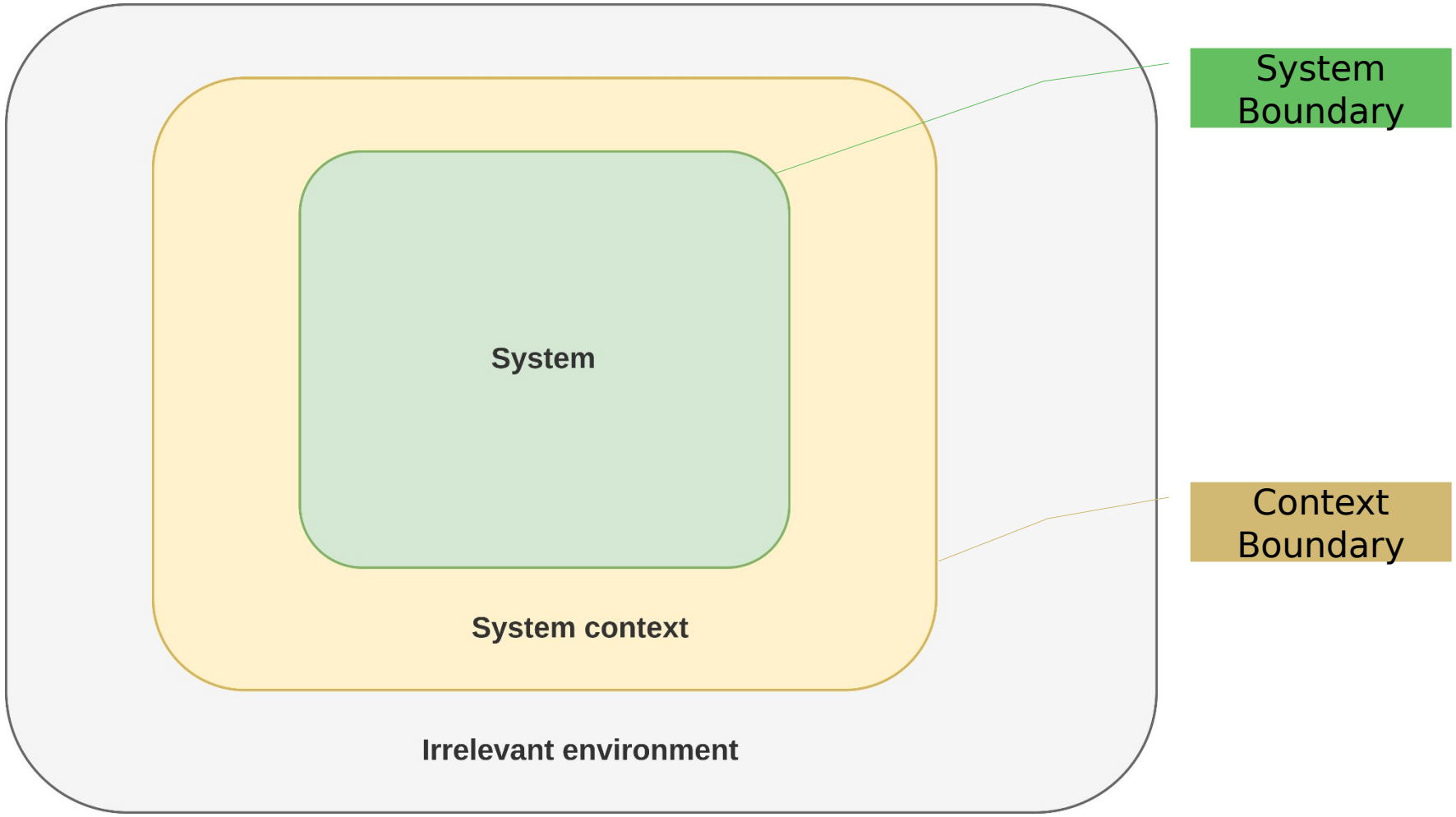
System Boundary

Development of the Boundary

- Not all elements of the system boundary are known at the beginning of a project
  - Interfaces unknown, desired functionalities unknown, ...
  - System boundary often not defined until the end of the requirements engineering process
- Leads to a “gray zone” between the system and the context
  - System boundary may shift within the gray zone
  - Gray zone itself may shift
  - At the end of the requirements engineering, no gray zone left

CONTEXT BOUNDARY

Context Boundary



Based on K. Pohl, C. Rupp (2011) – Requirements Engineering Fundamentals: A study guide for Requirements Engineering Foundation Level

Context Boundary

Overview

- Distinguishes between what is relevant for the requirements engineering and what is not.
- Similar to the system boundary: may shift and have a gray zone

Context Boundary

Definition

"The context boundary separates the relevant part of the system environment from the irrelevant environment which contains all those aspects that do not need to be considered during the system development."

K. Pohl, C. Rupp (2011) – Requirements Engineering Fundamentals: A study guide for Requirements Engineering Foundation Level

Context Boundary

Completeness of the Context Boundary

- Complete definition of the context boundary virtually impossible
  - Sometimes unclear, whether an aspect belongs to the system context or not
    - Cannot always be resolved during the requirements engineering
  - It is possible that after the requirements engineering, there is still a gray zone in the context
    - Different from the system context, where the gray zone is resolved

Context Boundary

Documenting the System Context

- Often through natural language

- Diagrams also very useful
  - Use case diagrams
    - Actors (people, other systems) and their usage relationship to the system
  - Data flow diagrams
    - Flow of data between the sources and sinks

- Typically, a mixture of several documentation forms

TYPES OF REQUIREMENTS

Types of Requirements

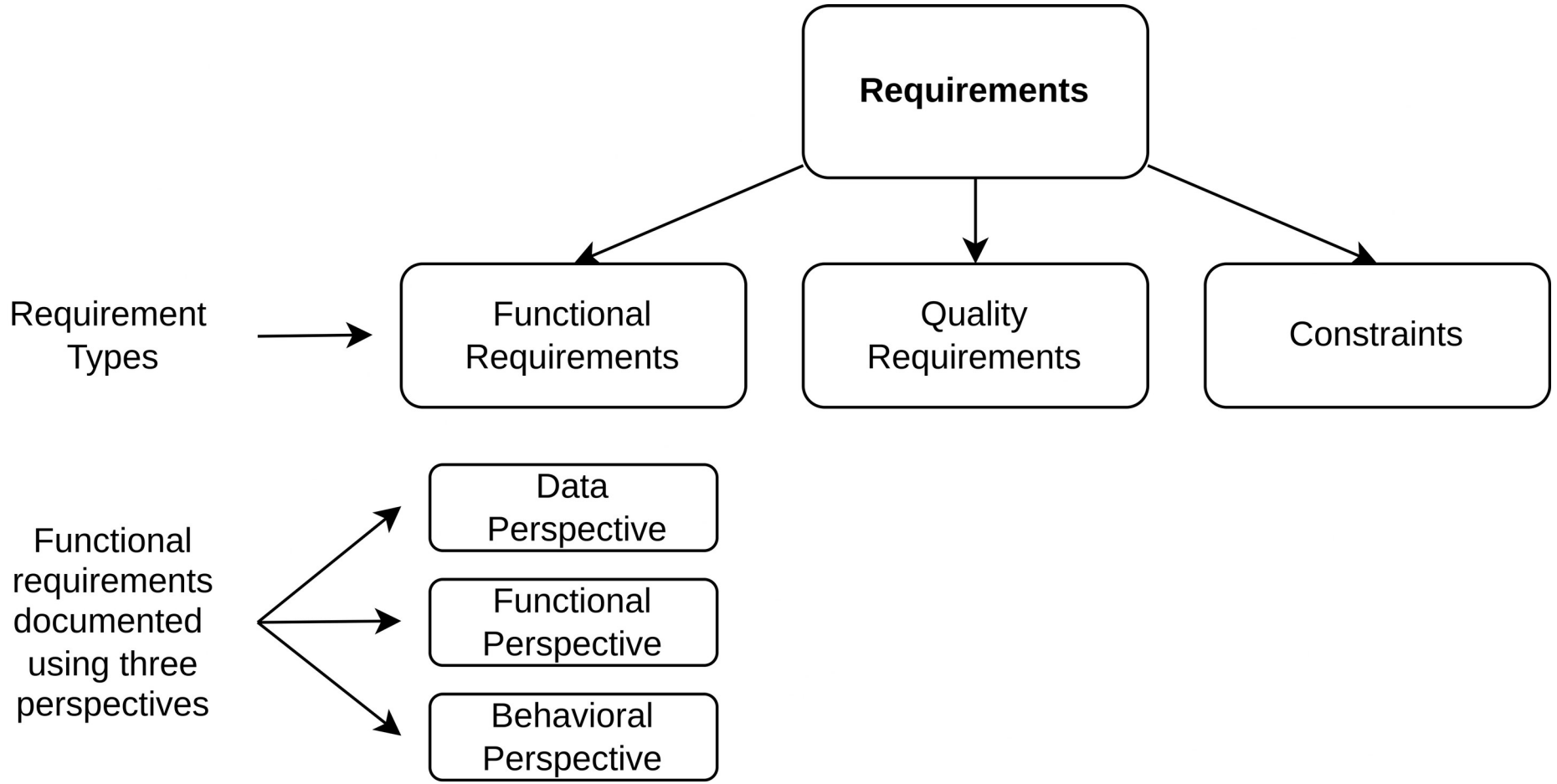
Motivation

- Different types of requirements must be documented for a complete requirements documentation.

- These requirements types differ with respect to:
  - adequate specification techniques
  - their importance for different types of systems

Types of Requirements

Functional Requirements



Types of Requirements / Functional Requirements

Data Perspective

- All systems need to deal with data
  - Data on customers, articles, etc.
  - Multimedia, e.g., videos, songs, etc.
  - ...

- Information must be adequately structured and represented:
  - Which information / data items are relevant to the system?
  - Which information / data items are at the boundary of the system?

Types of Requirements / Functional Requirements

Data Perspective – Representing Data

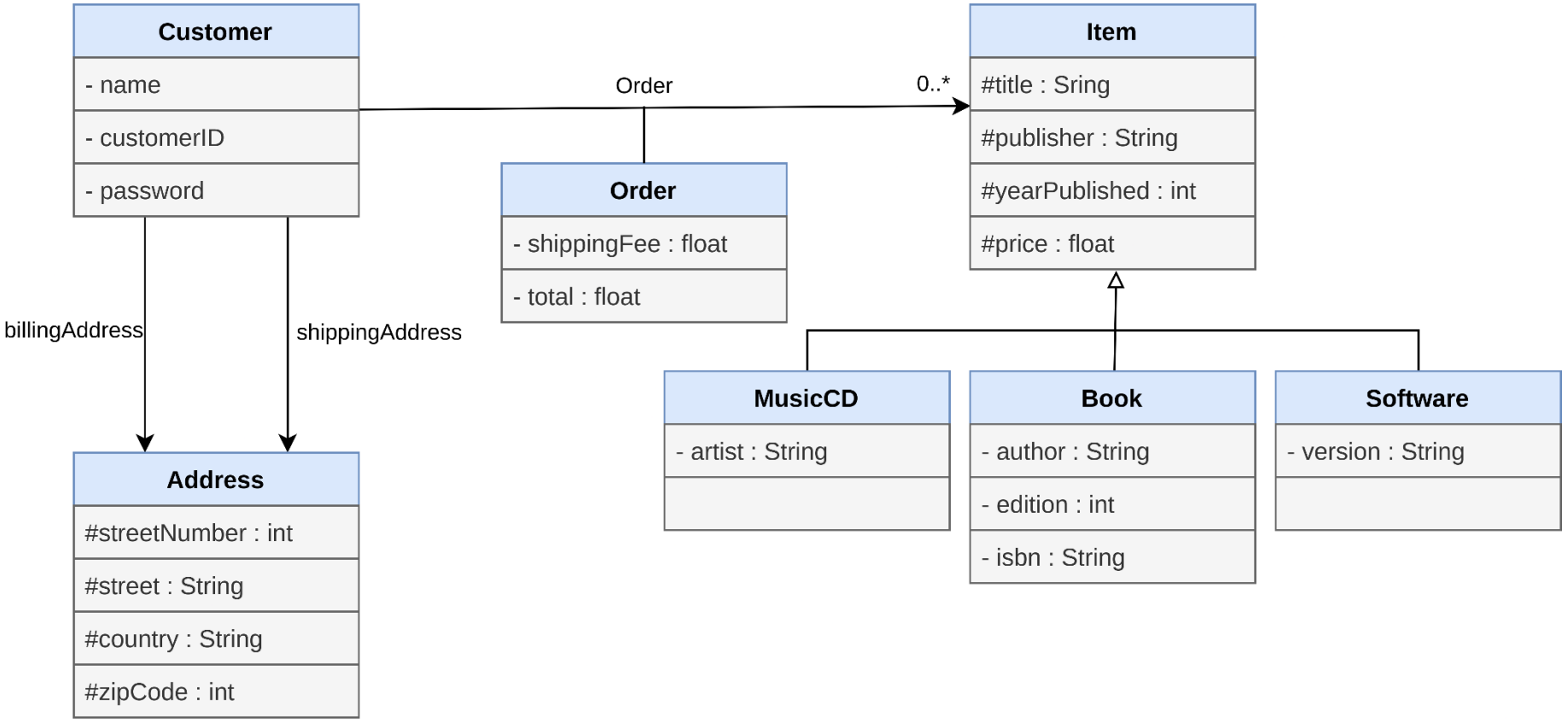
- UML class diagrams
- Context diagrams
- Data dictionary

Note:

- the data specified in the requirements need not be directly related to the implementation
  - same information, but different structure possible
  - e.g. attributes versus classes may change strongly
- In information systems understanding the data is a key driver !

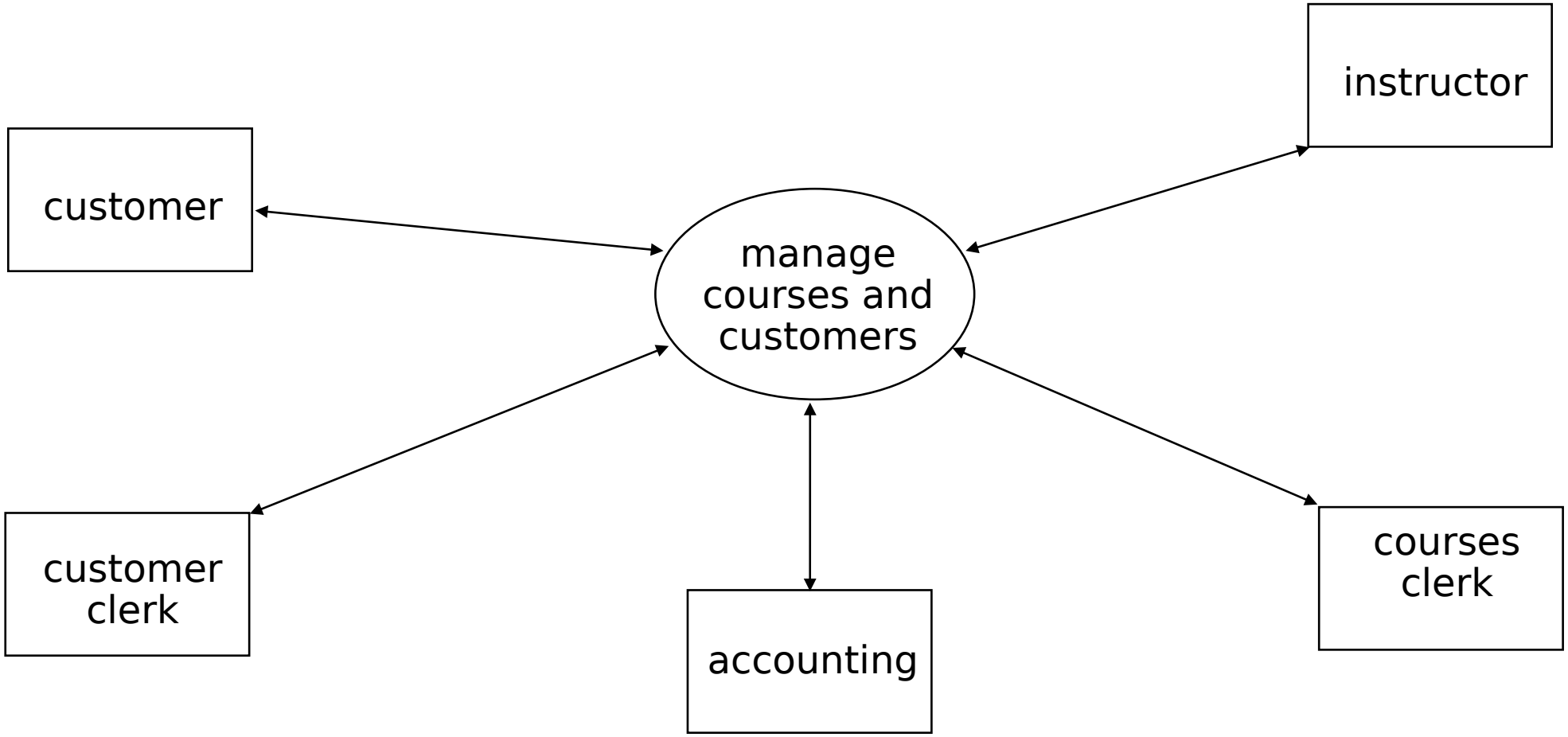
Types of Requirements / Functional Requirements

Data Perspective – Book Shop UML Example



Types of Requirements / Functional Requirements

Data Perspective – Context Diagram Example



Types of Requirements / Functional Requirements

Behavioral Perspective

Behavioral requirements describe what a system will do (with the data):

- how input information is transformed into state information and output information
- sequences of interaction of the software system with its environment (people, software, hardware)

System behavior is important on various levels:

- **Business processes** - describe the fundamental flow of activities in an enterprise
- **Task level** - describe the interaction of people with a software system on a coarse grained level (e.g., define new customer)
- **Stimulus / response** - describe interactions

Types of Requirements / Functional Requirements

Behavioral Perspective – Specification Techniques

Many different techniques were developed for specifying this:

- Textual Use Cases
- Business Process Modeling Languages
- Scenario-Based Modeling Approaches
- Event-Based Modeling Techniques

The techniques can be categorized along the following dimensions:

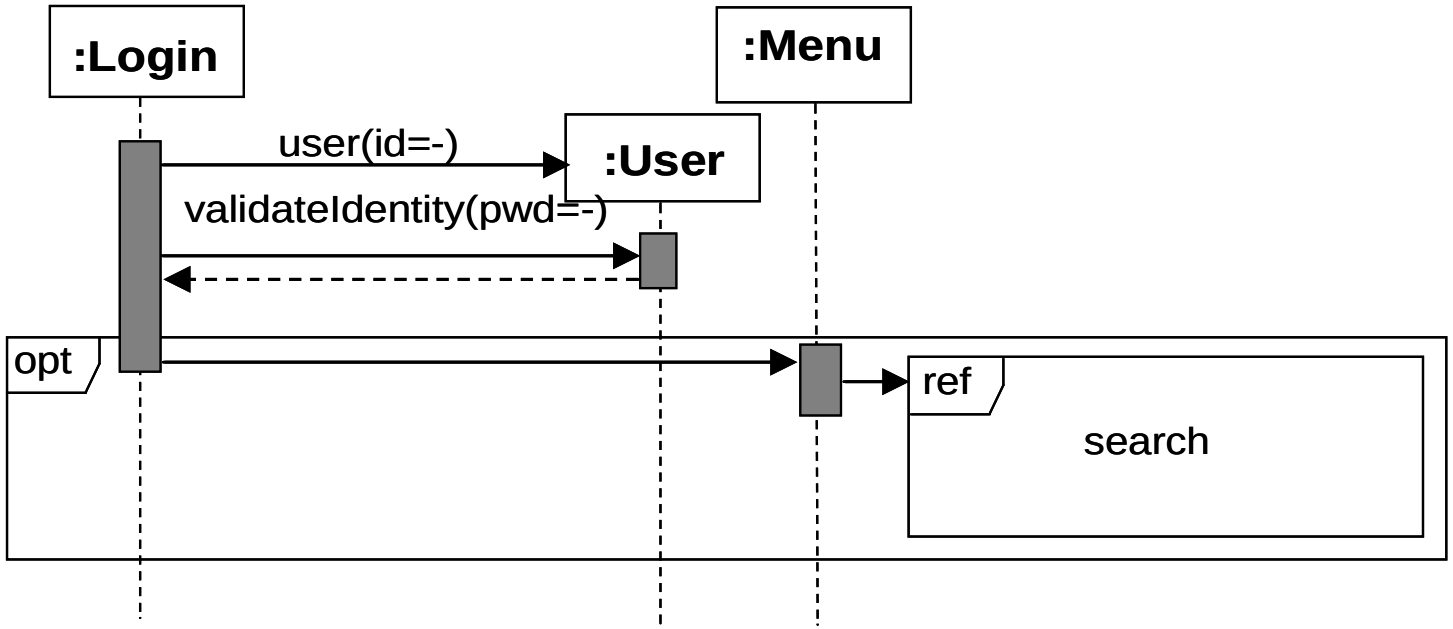
- **data-flow** (-transformation) vs. **stimulus-/response**
- **complete** description vs. **prototypical** description

Types of Requirements / Functional Requirements

Behavioral Perspective – Specification Techniques

The UML provides different approaches:

- Use Case Diagram
- State Machine Diagram
- Activity Diagram
-



Types of Requirements / Functional Requirements

Interface Requirements

- The interface takes apart interior and exterior
- Interfaces are defined by the project context
- Different types of interfaces:
  - User interfaces (Human-Machine Interface)
  - Software Interfaces
  - Hardware-related interfaces

Types of Requirements / Functional Requirements

Interface Requirements – User Interface

- Interface Description must describe
  - Layout
  - Look & Feel
  - Category of Interface (WIMP, ASCII-based, tactile, …)
  - Interaction sequences
  - ..

- Usability aspects are specific to this type of interface
  - Person (e.g., impairements, knowledge)
  - Situation
  - Task
  - …

Types of Requirements / Functional Requirements

Interface Requirements – Software Interface

- Interfaces to other software interfaces are defined based on
  - Identification of service, i.e., how to find it
  - The protocol (how to interact)
  - The data format(s), e.g., how to exchange data

- Typically use of standard protocols, like
  - Web-Service
  - HTTP
  - ...

But also possible → Data file is written to a specific location and read by another program

Types of Requirements / Functional Requirements

Interface Requirements – Hardware Interface

Hardware interfaces are often:

- Time critical
  - Protocol specification must include timing information
- Specified close to hardware (e.g. addressing)
  - Hardware-based → service identification may be given in bits and bytes
- Other than that, usually hardware interfaces are like a software interface
- Mapping software information to the physical world is done by hardware!

Types of Requirements

Non-Functional & Quality Requirements – Definition

1 Quality requirements **define qualitative attributes** of the whole system, a single function or a group of functions, i.e. how good a system shall do the things it is supposed to do.

2 Non-functional requirements are used to **encompass all kinds of *not* functional requirements** for a system:

- quality requirements should be related to the functional requirement or group of requirements they are relevant to
- development constraints should be captured separately
- project aspects should be clearly separated from product aspects

Types of Requirements

Non-Functional & Quality Requirements – Definition

1 Quality requirements **define qualitative attributes** of the whole system, a single function or a group of functions, i.e. how good a system shall do the things it is supposed to do.

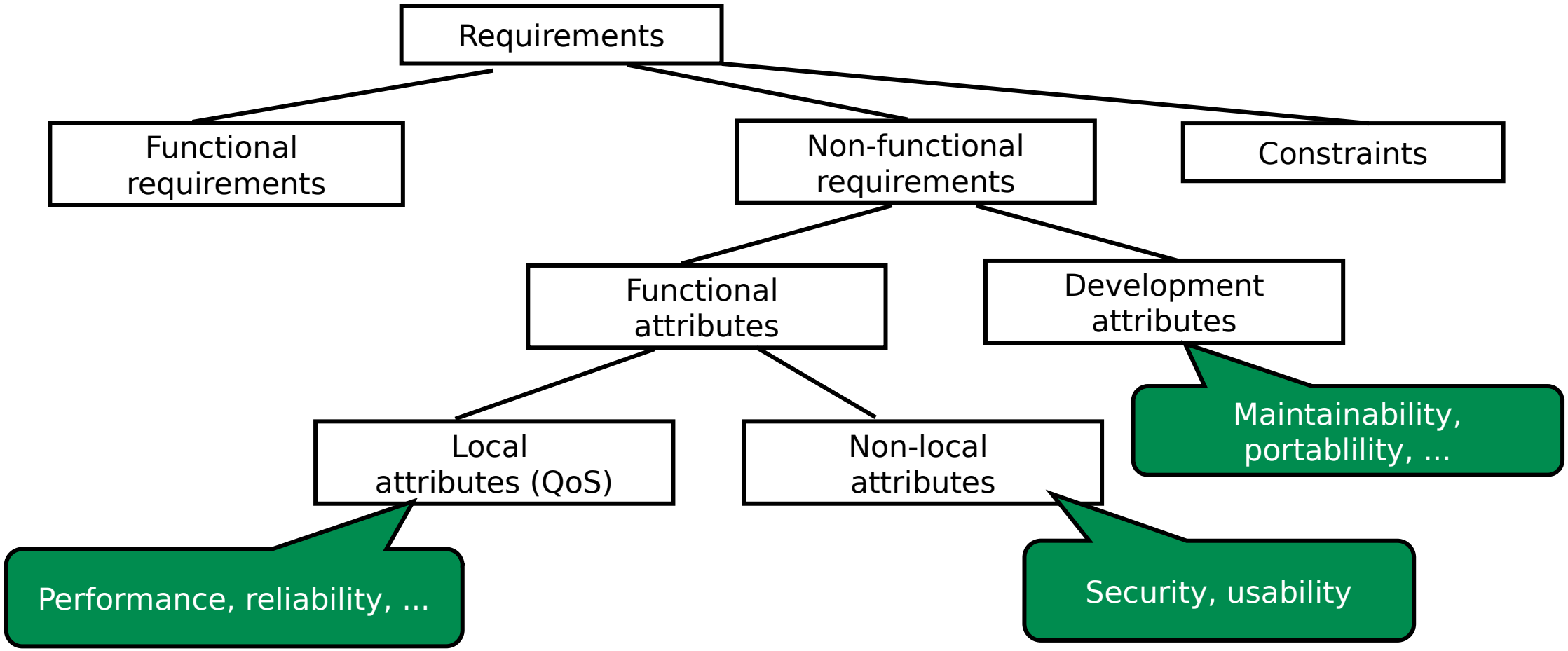
2 Non-functional requirements are used to **encompass all kinds of *not* functional requirements** for a system:

- quality requirements should be related to the functional requirement or group of requirements they are relevant to
- development constraints should be captured separately
- project aspects should be clearly separated from product aspects

The term non-functional requirements is depreciated (according to IEEE)

Types of Requirements

Non-Functional & Quality Requirements – Kinds of Requirements



Types of Requirements
Non-Functional & Quality Requirements – Product Quality (ISO 9126 / DIN 66272)

- Functionality
  - Adequacy
  - Security
  - Precision of calculation
  - Interoperability
  - Conformity with standards
- Reliability
  - Maturation
  - Fault tolerance
  - Recovery
- Usability
  - Comprehensibility
  - Learnability
  - Operability

Types of Requirements
Non-Functional & Quality Requirements – Product Quality (ISO 9126 / DIN 66272)

- Efficiency
  - Time response
  - Resource Consumption
- Changeability
  - Analyzability
  - Modifiability
  - Stability
  - Verifiability
- Portability
  - Adaptivity
  - Installability
  - Conformity with standards
  - Replaceability

Types of Requirements

Non-Functional & Quality Requirements – Example „*Performance*“

- User level → The user can create accounts with only **two** interactions
- System task level → The creation of an account (pressing of the „system availability“ button) takes max. 0.5 seconds

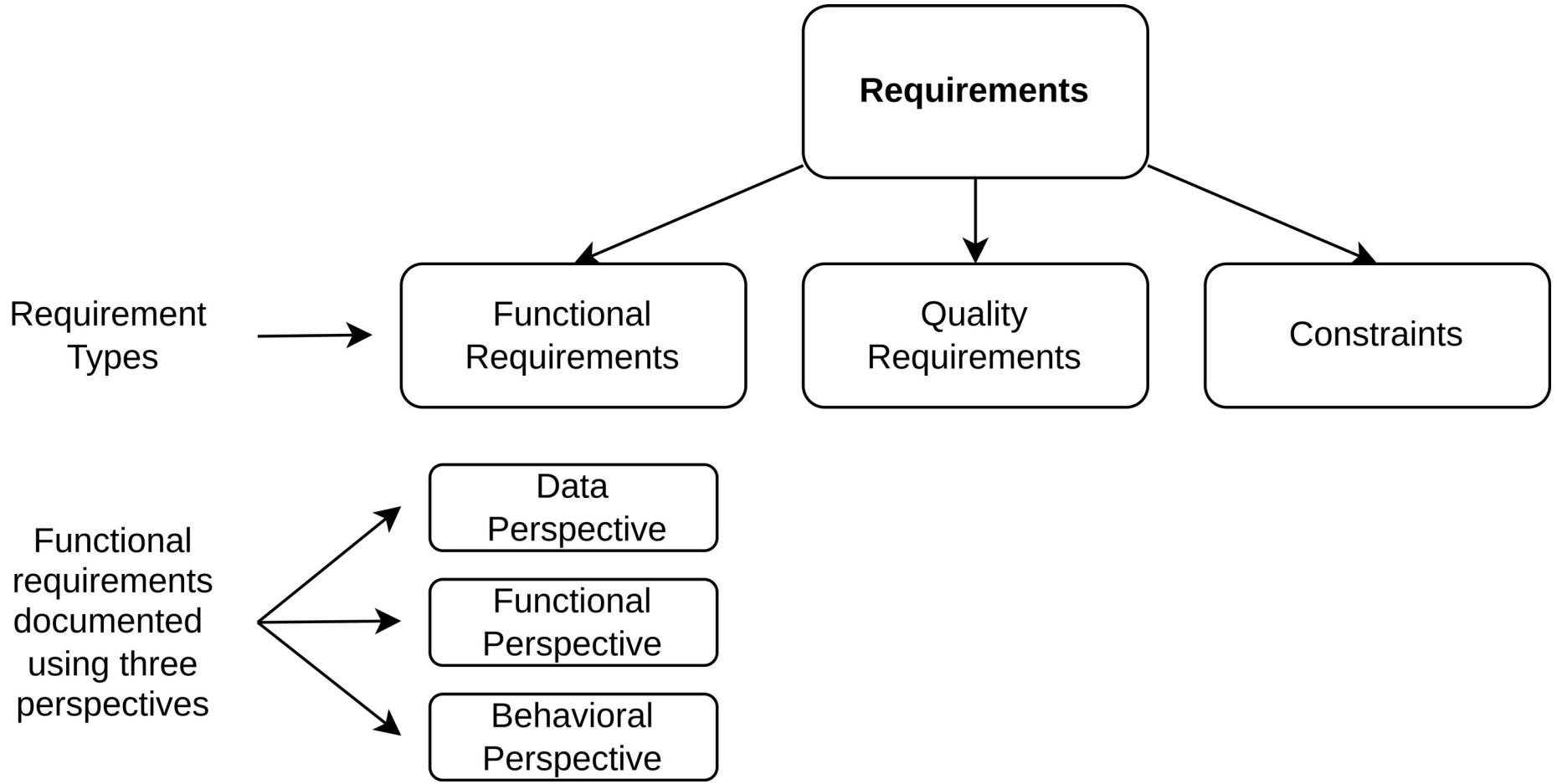
→ Derived non-functional requirement result from the interplay between both levels.

SUMMARY

Summary

- The system context defines what is relevant for the system and what can be ignored
  - Stakeholder, data sources and sinks, standards, ...
- The system boundary defines the scope of the system
  - E.g., which functionalities are provided by the system in comparison to what is provided by the system context
- A wrong system context leads to erroneous requirements
  - Possibly fatal for a project

Summary


Requirements
Requirement Types
Functional Requirements
Quality Requirements
Constraints
Functional requirements documented using three perspectives
Data Perspective
Functional Perspective
Behavioral Perspective

Questions?