Software Engineering

Prof. Dr. Joakim von Kistowski

TH Aschaffenburg
university of applied sciences

Check-In

Why is it important to deal with requirements?

Introduction: Requirements

Requirements Engineering

Context Analysis

Requirements specification / functional spec.

Use-Cases

TH Aschaffenburg
university of applied sciences

Disciplines in software engineering

Focus module

Basic topics

Configuration management | Documentation |
Knowledge management | People in the SWE process and digital ethics | Tools

Development

Requirements
- Context analysis
- Requirements Engineering

Design
- Architecture
- Detailed design

Implementation

QualityMgt.

Quality assurance and testing
- Test, inspection, metrics

Processes and procedure models
- Improvement, process model, maturity levels

Evolution

- Roll-Out
- Operation
- Maintenance
- Further development
- Reuse
- Reengineering
- Change management

Management

- Strategy
- Economy
- Team
- Dates
- Risks
- Customer, client/contractor
- Innovation

Project Task

Part of **Readiness for Acceptance** is a requirements specification with the following contents:

- Product vision and product goals
- Roles and personas
- User stories
- Glossary of terms
- Quantity structure
- Use cases

- Introduction: Requirements
- Requirements Engineering
- Context Analysis
- Requirements specification / functional spec.
- Use-Cases

Learning Objectives

- You can explain **why** it is essential to deal with requirements.
- You can explain what requirements engineering is and which sub-disciplines are involved.
- You can use examples to explain why RE is essential for the **success** of a project.
- Can you explain the **challenges** that can arise when dealing with requirements?

TH Aschaffenburg
university of applied sciences

Importance of the requirements recognized very early on

This is not always reflected in practice

> *The hardest single part of building a software system is deciding **precisely what to build**. No other part of the conceptual work is as difficult as establishing the detailed technical requirements ... No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later.*
>
> Fred Brooks, 1987

The software requirements are the most important information in software development.

Why? What happens if the requirements are unclear?

An entire field, requirements engineering, deals with the topic of "requirements"

According to IREB (International Requirements Engineering Board)

Requirements engineering is a

- systematic, disciplined and quantifiable process for the
  - Determination
  - Analysis
  - Specification and
  - Management of requirements
- with the goals
  - understand the wishes and needs of stakeholders and users and
  - minimize the risk of delivering a product that does not meet these wishes and needs.

What is involved?

- Requirements elicitation (extraction & elicitation)
  - What does the customer/user need/want? Collecting (functional and non-functional) requirements, e.g. through interviews/user discussions, market observation, design thinking workshops, etc.
- Requirements analysis (analysis, design, validation)
  - Classification, evaluation, prioritization, comparison and testing (e.g. according to cost/benefit aspects, consistency, completeness, etc.), e.g. through prototyping and iterative testing
- Requirement description/specification (Specification)
  - Description in a standardized form (e.g. naturally linguistic, sentence template, models, prototypes, etc.)
- Requirements management
  - Re-examination/modification of requirements

What is involved? Using the example of house building

- Extraction and elicitation:
  - Architect surveys and determines "wishes" = requirements of the client
- Requirements analysis (analysis, design, validation)
  - Architect classifies, evaluates, prioritizes (together with the client) the requirements, performs cost/benefit analysis, creates models (prototypes)
- Requirement description/specification (Specification)
  - Architect documents the requirements, plans, decisions, etc.
- Requirements management
  - Dealing with requirements in the course of the construction project, e.g. status, some requirements are discarded (because too expensive, for example), some requirements are added (e.g. if the ground conditions require a certain construction method)

What is involved? Using the example of house building

- Extraction and
  - Architect su
    ent
- Requirements
  - Architect cla
    ne requirements, makes
    cost/benefit
- Requirement c
  - Architect:in
- Requiremen
  - Dealing with
    e.g. status, some
    requirement
    ome requirements are
    added (e.g.
    ethod)

Requirements engineering is not an end in itself!

→ Results determine the quality and efficiency of the construction project

→ Results are required in the further process

→ The key to customer satisfaction

What is involved?

- **Requirements elicitation (extraction & elicitation)**
- **Requirements analysis (analysis, design, validation)**

Analysis

- **Requirement description/specification (Specification)**

Specification

- Requirements management

What is a requirement?

→ A Requirement solves a real problem or fulfills a real user need

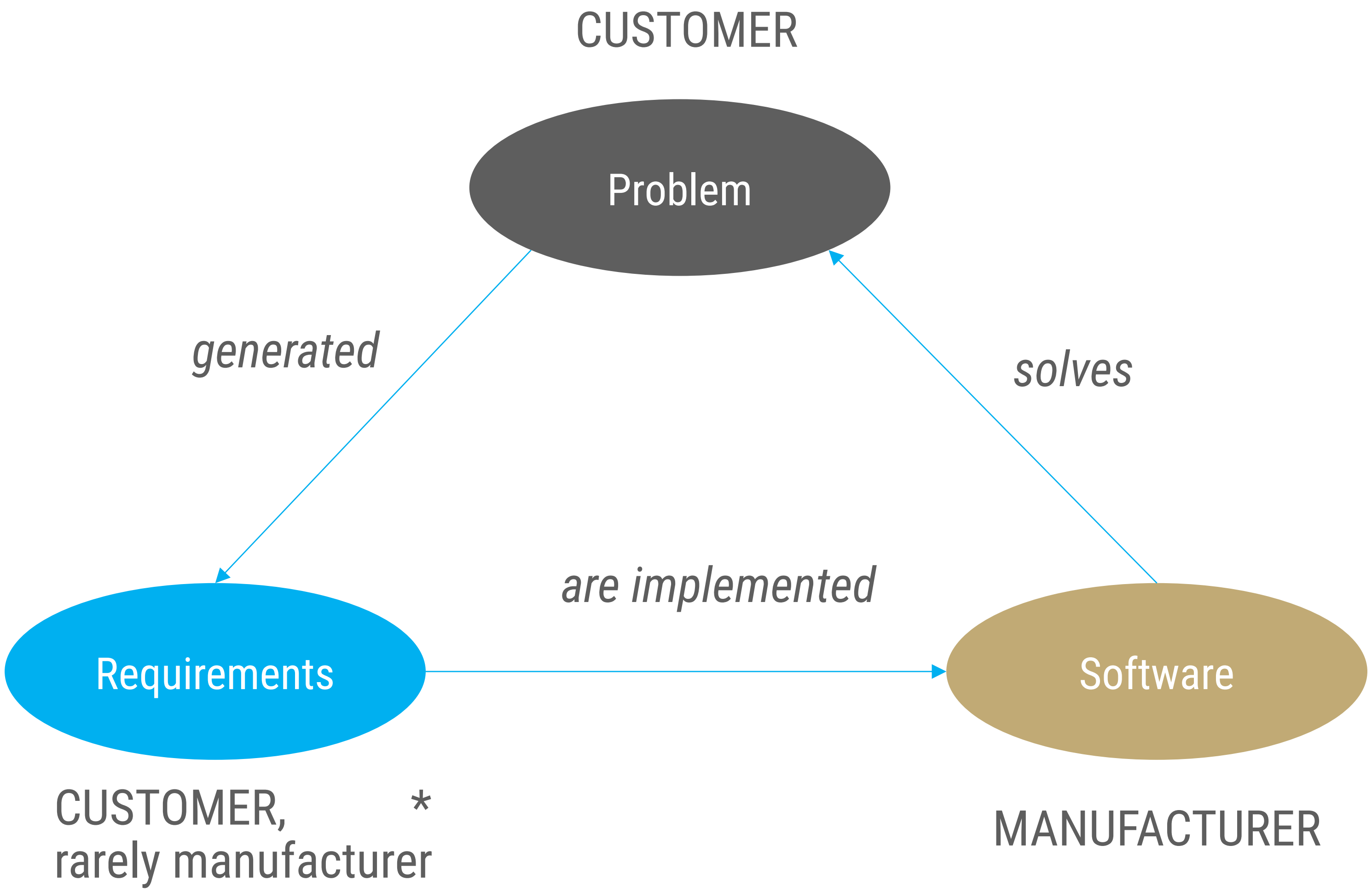
requirement - *(1) A condition or capability needed by a **user** to **solve** a problem or achieve an objective.*
(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
(3) A documented representation of a condition or capability as in (1) or (2).

IEEE Std 610.12-1990

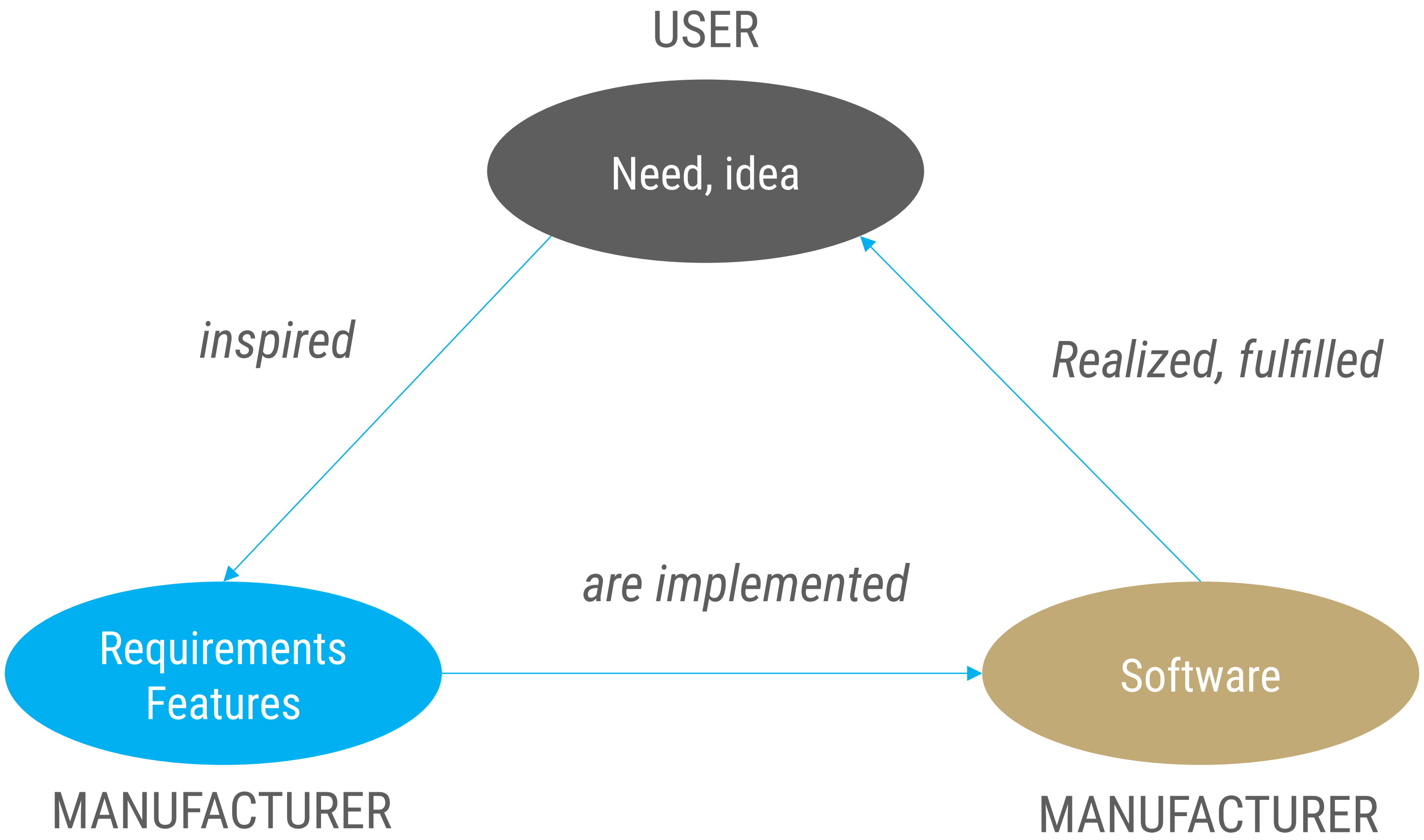
Requirement - *(1) A condition or capability needed by a person to solve a problem or achieve a goal. (2) A condition or capability that software must meet or possess in order to fulfill a contract, standard, or (3) other formally specified document. (IEEE 610.12-1990)*

IEEE Std 610.12-1990

Where do requirements come from?
Project-based software engineering



Where do requirements come from?
Product-based software engineering



Where do requirements come from?

Motivation for software (further) development

1. Business and **consumer** needs
2. **Dissatisfaction** with existing solutions
3. **Technological** changes
4. …

What are sources of requirements?

1. System context
2. Stakeholders (customers, users, but also operations, development, etc.)
3. Systems (legacy system, competition, etc.)
4. Documents

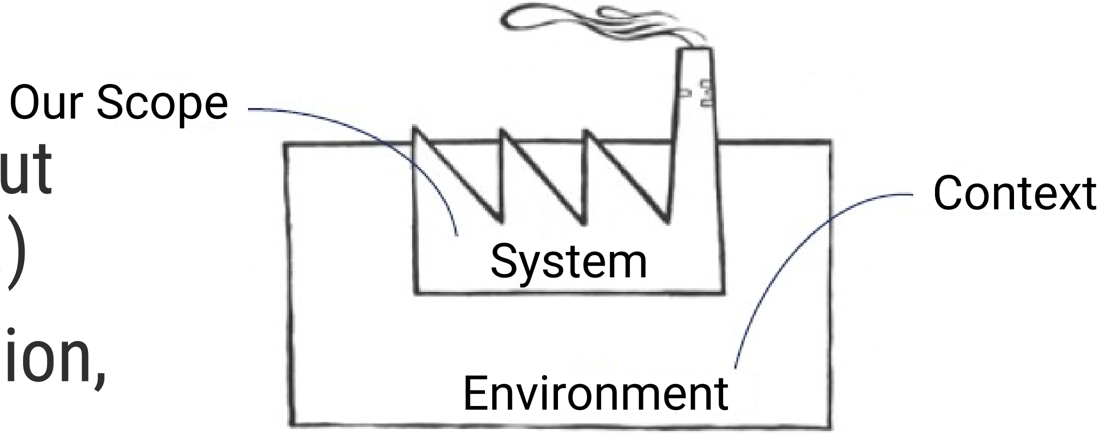


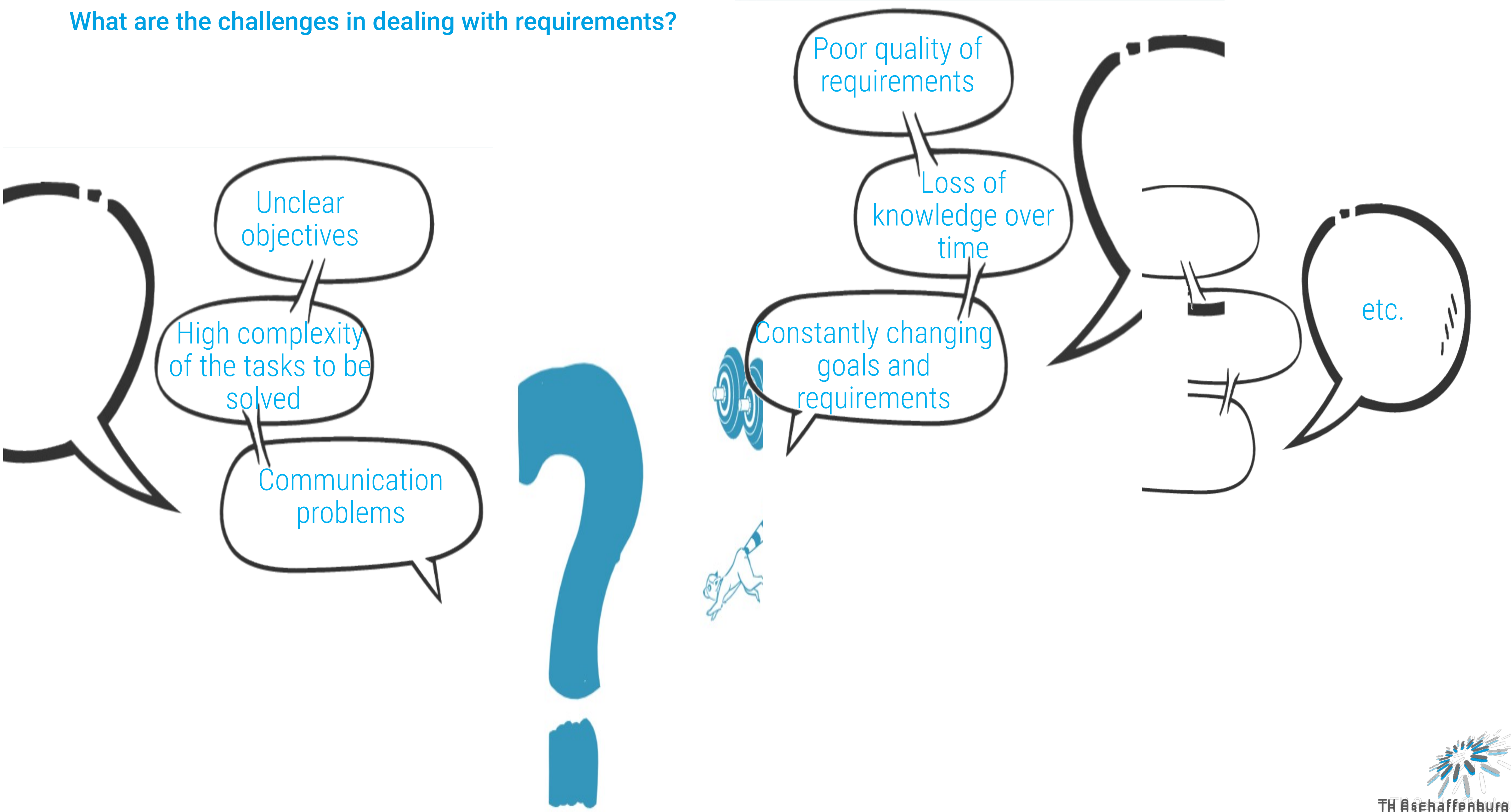
Abbildung 6.6: Scope versus Kontext

[Rupp, 2021]

Question

What are the challenges in dealing with requirements?

What are the challenges in dealing with requirements?



Unclear objectives

High complexity of the tasks to be solved

Communication problems

Poor quality of requirements

Loss of knowledge over time

Constantly changing goals and requirements

etc.

Introduction: Requirements

Requirements Engineering

Context Analysis

Requirements specification / functional spec.

Use-Cases

Importance of Requirements Engineering

> *„If I had asked people what they wanted, they would have said faster horses.“*
>
> Henry Ford

Learning Objectives (1/2)

- You can explain why the analysis of the current state should not be skipped.
- You can apply and compare methods for requirement elicitation.

Learning Objectives (2/2)

- You know different classification options for requirements and can give examples.
- You can give examples of **non-functional requirements**.

Analysis of current state, analysis wanted/target state

→ Also actual survey, target survey

- In order to be able to describe the **requirements for the software** well, an **analysis of the current state** is necessary, i.e. an understanding of the current (working) environment of the users. We also call the (working) environment the context.
- As far as possible, this should be **independent** of the considerations regarding the new software so that the focus is not immediately on the technical **solution**.
- Based on this, the **analysis of the wanted/target state** is carried out, i.e. the **wishes** for the new software **are determined** and the requirements are specified.

Difficulties part 1

- Actually, it shouldn't be a problem to ask customers/users about their wishes, ... OR?
  - Customers/users focus on what they currently don't like.
  - Implicitly, customers/users expect that everything that has worked or was good so far will stay that way or get better.
  - The requirements for the new system therefore only consist to a very small extent of requests for change; the vast majority of requirements are for continuity.
- Tips:
  - Create an understanding of the user's environment (What do they see, hear, do?)
  - Collect all wishes and pain points (Pains & Gains), do not criticize for the time being "Wish list"
  - Disclose contradictions objectively, discuss alternatives (most people are not even aware of them)

Difficulties part 2

- **Too many** requirements
- **Unrealistic** requirements
- Customer has requirements that he **does not** want **to say**
- Customer has requirements that are so **self-evident** that he/she does not say them
- Customer has requirements which he **cannot formulate** (knows the problem but cannot derive the requirement)
- Etc.

Methods of requirements elicitation/survey

Method	Current State Analysis	Target State Analysis
Evaluation of existing data, documents **Systems archaeology**	✓	(✓)
Observation	✓	(✓)
Survey (open, structured, closed questions | interview (individual vs. group) vs. questionnaire)	✓	✓
Creativity techniques, e.g. mind mapping, headstand, 635 method, etc.		✓
Prototyping, **experiments**		✓
Design Thinking, Living Lab, CrowdRE		✓

Co-Creation

System archaeology, document analysis

- "**Asynchronous**" method independent of the availability of stakeholders
- Increases the chance of not forgetting any requirements from the old system
- Can be very time-consuming and depend on the quality of the documents
- Reuse, further development is based on the results of system archaeology
- Must be combined with other techniques to ensure that documents are up to date

TH Aschaffenburg
university of applied sciences

Observation

- Field observation
  - Stakeholders are monitored (users, future users)

    when using the IST system in the system context

    at work (with or without a system → to record work processes, time dependencies, etc.)
- Apprenticing
  - Like field observation, but interruptions are allowed for interim questions from the requirements engineer.
- Contextual Inquiry
  - Requirements Engineer "learns/executes" the activities of the users in order to derive requirements for the new/changed system

Survey → Interview

→ Tips for good interviews

- **Prepare for** the interview! (open vs. closed questions)
- Active listening (absorbing and **repeating** information)
- **Record** everything (note-taker should not be the interviewer at the same time)
- **Prepare** the interviews. Clean up the notes **promptly!!!** while the knowledge/experience is still fresh.

Survey → Interview

→ Questioning techniques based on [Rupp, 2021]

- **Deep drilling**: Can you describe this in more detail? Which aspects are important to you? What else is involved?

Survey → Interview

→ Questioning techniques based on [Rupp, 2021]

- **Deep drilling**: Can you describe this in more detail? What aspects are important to you? What else is involved?
- **Active listening:** So could you describe the situation as follows? Did I understand you correctly when I said ...?

Survey → Interview

→ Questioning techniques based on [Rupp, 2021]

- **Deep drilling**: Can you describe this in more detail? What aspects are important to you? What else is involved?
- **Active listening:** So could you describe the situation as follows? Did I understand you correctly when I said ...?
- **Synthesis:** Let me summarize once again ... In your opinion, the most important points are 1. ... 2. ... 3. ...

Survey → Interview

→ Questioning techniques based on [Rupp, 2021]

- **Deep drilling**: Can you describe this in more detail? Which aspects are important to you? What else is involved?
- **Active listening:** So could you describe the situation as follows? Did I understand you correctly when I said ...?
- **Synthesis:** Let me summarize once again ... In your opinion, the most important points are 1. ... 2. ... 3. ...
- **Filter:** Separate the important from the unimportant. What is more important, A or B? How often does << case x occur >>

TH Aschaffenburg
university of applied sciences

Survey → Interview

→ Questioning techniques based on [Rupp, 2021]

- **Deep drilling**: Can you describe this in more detail? What aspects are important to you? What else is involved?
- **Active listening:** So could you describe the situation as follows? Did I understand you correctly when I said ...?
- **Synthesis:** Let me summarize once again ... In your opinion, the most important points are 1. ... 2. ... 3. ...
- **Filter:** Separate the important from the unimportant. What is more important, A or B? How often does << case x occur >>
- **Parking:** If unclear, clarify the parking issue and further procedure afterwards.

Survey → Questionnaire

- Advantages
  - High number of stakeholders can be surveyed efficiently
  - Time and location-independent form
  - Easy to evaluate
- Disadvantages
  - Implicit knowledge difficult to determine
  - Follow-up questions/further questions difficult
  - Suggestion
- The questionnaire can offer a good alternative as a basis for further methods

Classification of requirements

→ What types of requirements are there?

- Functional vs. non-functional requirements
- Open/latent requirements
- MUST vs. CAN requirements
- ...

Functional vs. non-functional requirements

→ "What" and "How well" a requirement must be implemented

- Non-functional requirements
  - describe **how well** a service must be fulfilled.
  - **Can increase production costs many times over.**
  - Must be determined, validated and documented just as systematically as functional requirements
  - Sometimes they are also referred to as "quality requirements".

Non-functional requirements

"How well" a requirement must be implemented

Non-functional requirements

Requirements for the technology

Requirements for the user interface

Legal and contractual requirements

Quality requirements →

Requirements for other delivery components

Requirements for activities to be carried out

Quality requirements

- A quality requirement includes specifications for the quality of the **product** or **process**, e.g. requirements for quality of use, product quality, data quality, etc.
- Different **standards** for describing individual quality features
- Company-specific specifications necessary

Quality requirements → Characteristics of usability

→ According to Ludewig, 2013

Correctness:	is high if the specification is correct and the rest of the software is correct in relation to the specification.
Reliability:	is high if the software rarely fails to perform the expected function.
Accuracy:	is high if the results deviate only slightly from the mathematically correct result.
Efficiency:	is high if the software requires hardly any more computing time than the minimum required.
Frugality:	is high if the software requires hardly any more storage space and other resources than would be minimally required.
Performance completeness:	is high if the software actually provides all the required services.
Manual completeness:	is high if the manuals provide exhaustive information on all useful user questions.
Consistency:	is high if the software behaves similarly towards the user in similar situations. This applies to operation, error messages, data formats, etc.
Comprehensibility:	is high if the user quickly understands how to use the software.
Simplicity:	is high if the software appears conceptually simple to the user.

Quality requirements → Maintainability characteristics

→ According to Ludewig, 2013

Specification completeness:	is high if the specification fully states the actual requirements and only states these completely.
Locality of the software:	is high if remote effects in the software (effects beyond the boundaries of the software components) are avoided.
Testability of the software:	is high if the programs can be executed under defined conditions and the relevant results can be recorded in full. The execution is therefore reproducible.
Structuredness:	is high if the software is divided into logically self-contained units with high cohesion and low coupling.
Simplicity:	is high if the software only contains a few designs that are difficult to understand.
Scarcity of the software:	is high if its scope has been kept low by avoiding redundancy of any kind.
Readability of the software:	is high if a (foreign) reader is able to grasp the content correctly with minimal effort.
Device independence:	is high if features of special devices play a minor role in it.
Seclusion:	is high if the software provides a well-defined service and therefore has hardly any interfaces to other systems.

Requirements for delivery components

- all products to be supplied that serve the development, operation and use of the system.
  - Development artifacts, e.g. prototypes, test documentation
  - Operating artifacts, e.g. operating manual, safety concept, hardware documentation
  - Application artifacts, e.g. user manual, online tutorials
  - Installation tools

Other non-functional requirements

- Legal and contractual requirements
  - Contractual requirements (e.g. maintenance contracts)
  - Requirements for the subcontractor (e.g. only after approval by the client)
  - Offer and costs
  - Legal requirements & compliance
- Requirements for activities to be carried out
  - Development activities or activities for and operation of the product.
    - Product development activities, e.g. reviews, quality management
    - Activities during the product life cycle, e.g. installation, maintenance, etc.

Other non-functional requirements

- Technology requirements
  - Restrict the solution space for implementation
  - Specify solutions, properties of the system or its interfaces or its operating environment
- Requirements for the user interface
  - Specifications regarding the user interface (optics, acoustics, haptics) and the operation of the system (e.g. design, dialogue guidance, personalization, internationalization, accessibility, etc.)

MUST vs. CAN

→ Legally binding

- MUST: Mandatory requirements, acceptance of the system can be refused if a MANDATORY requirement is not fulfilled
- CAN: Optional requirements that increase stakeholder satisfaction when fulfilled.

Open vs. latent

→ Legally binding

- OPEN: Requirements formulated by the customer, user
  - Known, explicit
- LATENT: Unconscious, unformulated requirements
  - Unknown or unconscious, implicit
  - Reasons: Requirements are assumed and are not explicitly stated or they are not known

Introduction: Requirements

Requirements Engineering

Context Analysis

Requirements specification / functional spec.

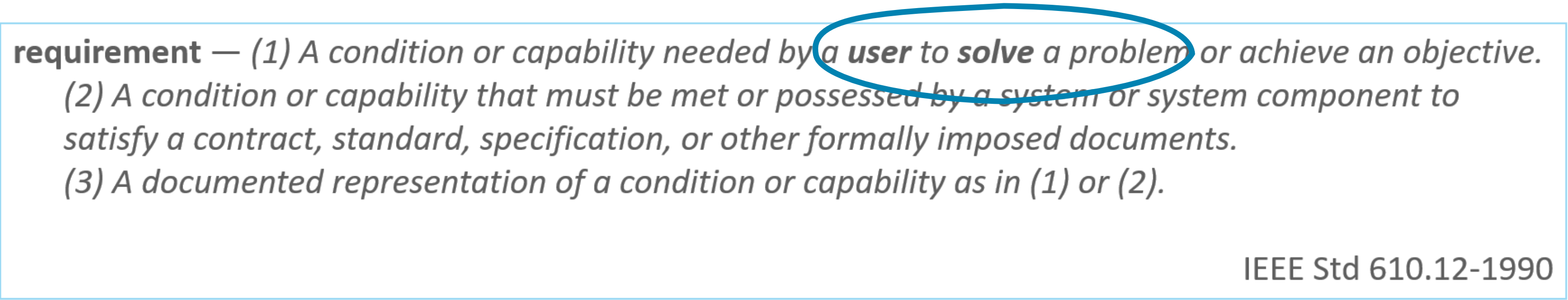
Use-Cases

Learning Objectives

- You can explain why it is important to understand the context of use and carry out a context analysis.
- You can apply different methods of context analysis and specification.
- You can explain why a dictionary of terms should be maintained.

Why is it important to understand the context of use?

- Because we develop software FOR users. That's why it's important to understand what needs and challenges need to be solved.
- Never start building software without understanding for whom and what problem is to be solved or what need is to be fulfilled!



requirement — *(1) A condition or capability needed by a **user** to **solve** a problem or achieve an objective.*

(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.

(3) A documented representation of a condition or capability as in (1) or (2).

IEEE Std 610.12-1990

Before we get into context analysis, it is important to know that there are ...

- different **levels of abstraction and views of** a system, and that there are
- There are different methods for different levels of abstraction and views.

Different levels of abstraction

Course granular

Context level	Describes the context of use. Who are the users? What challenges do they face, etc.?	The requirements for a house for a family of five are different from those for an apartment for a retired couple.
Domain level	Describes the area of application, e.g. house building, banking, gaming, e-mobility, etc.	Here it is important to understand what makes the "domain" tick, what vocabulary is important, what rules there are, etc.
Interaction level	Describes the way in which the system is interacted with.	What interaction patterns are there? Speech recognition? Mouse input, touch, etc.
System level	What **features and system functions** should the system have?	Which user requirements are mapped to which functions/features?

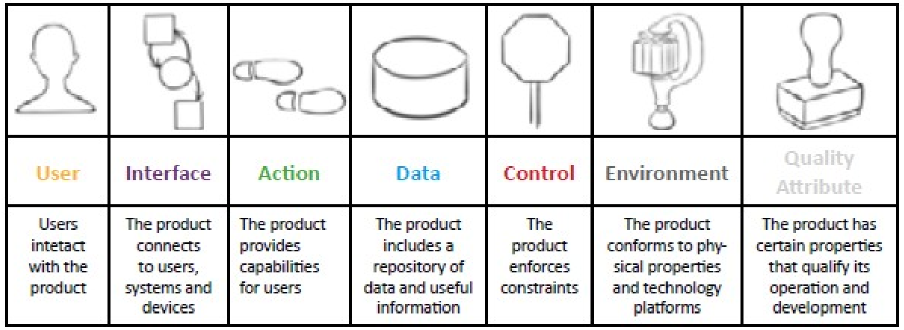
Fine granular

Different levels of abstraction

Context level	Describes the context of use. Who are the users? What challenges do they face, etc.?		Shopping
Domain level	Describes the area of application, e.g. house building, banking, gaming, e-mobility, etc.		Banking
Interaction level	Describes the way in which the system is interacted with.		Interactio n with the system
System level	What **features and system functions** should the system have?	Read card, authenticate, carry out transaction, cancel, ...	System modeling

Different **views** of a system

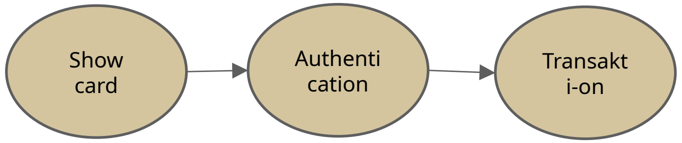
Users *Interaction/ Interface* *Procedure* *Data* *Non-functional Requirements* *Surroundings* *Constraints/ Restrictions*



User	Interface	Action	Data	Control	Environment	Quality Attribute
Users intetact with the product	The product connects to users, systems and devices	The product provides capabilities for users	The product includes a repository of data and useful information	The product enforces constraints	The product conforms to physical properties and technology platforms	The product has certain properties that qualify its operation and development

According to: Gorman, 2021 - Discover to Deliver: Agile Product Planning and Analysis

Different **views** of a system



Users

Interaction/ Interface

Procedure

Data

Non-functional Requirements

Surroundings

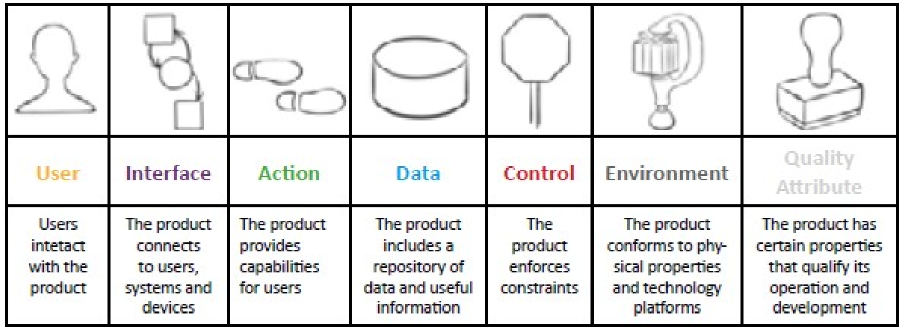
Constraints/ Restrictions

Bank details: Customer, account, etc

Secure, performant, usable

Platform, etc.

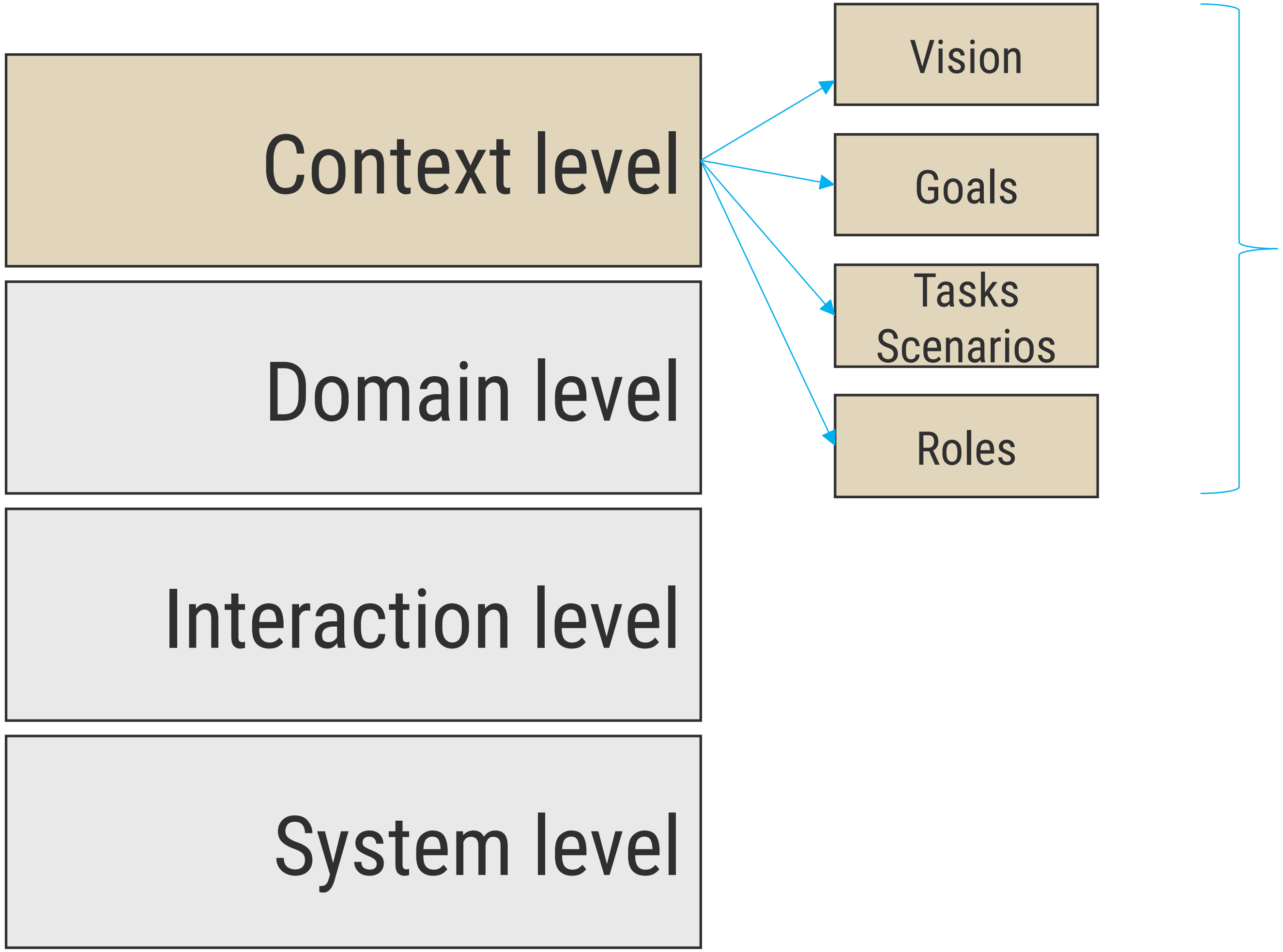
3 times invalid PIN possible, etc.



User	Interface	Action	Data	Control	Environment	Quality Attribute
Users intetact with the product	The product connects to users, systems and devices	The product provides capabilities for users	The product includes a repository of data and useful information	The product enforces constraints	The product conforms to physical properties and technology platforms	The product has certain properties that qualify its operation and development

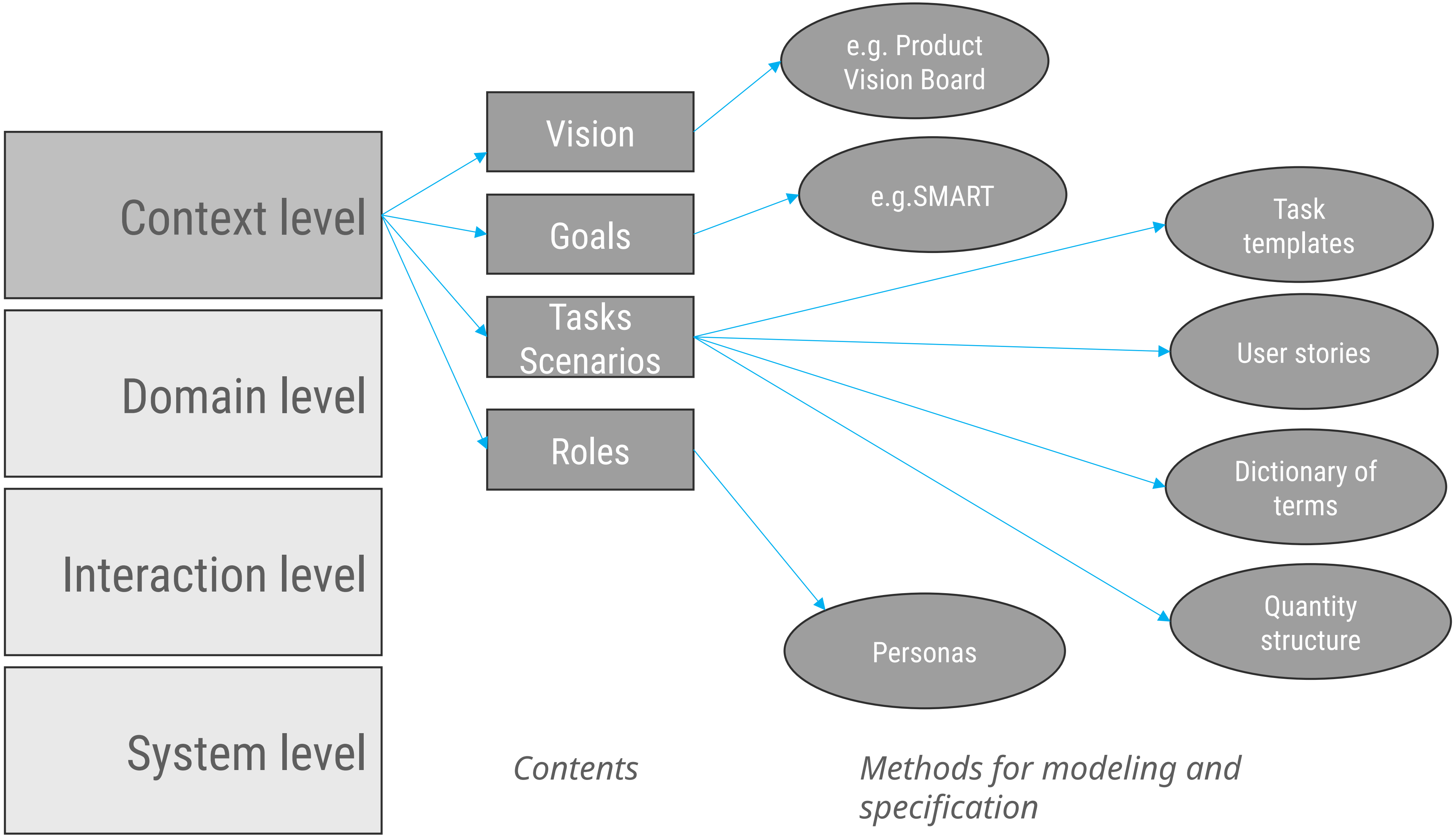
According to: Gorman, 2021 - Discover to Deliver: Agile Product Planning and Analysis

Requirements exist at different **levels of abstraction**



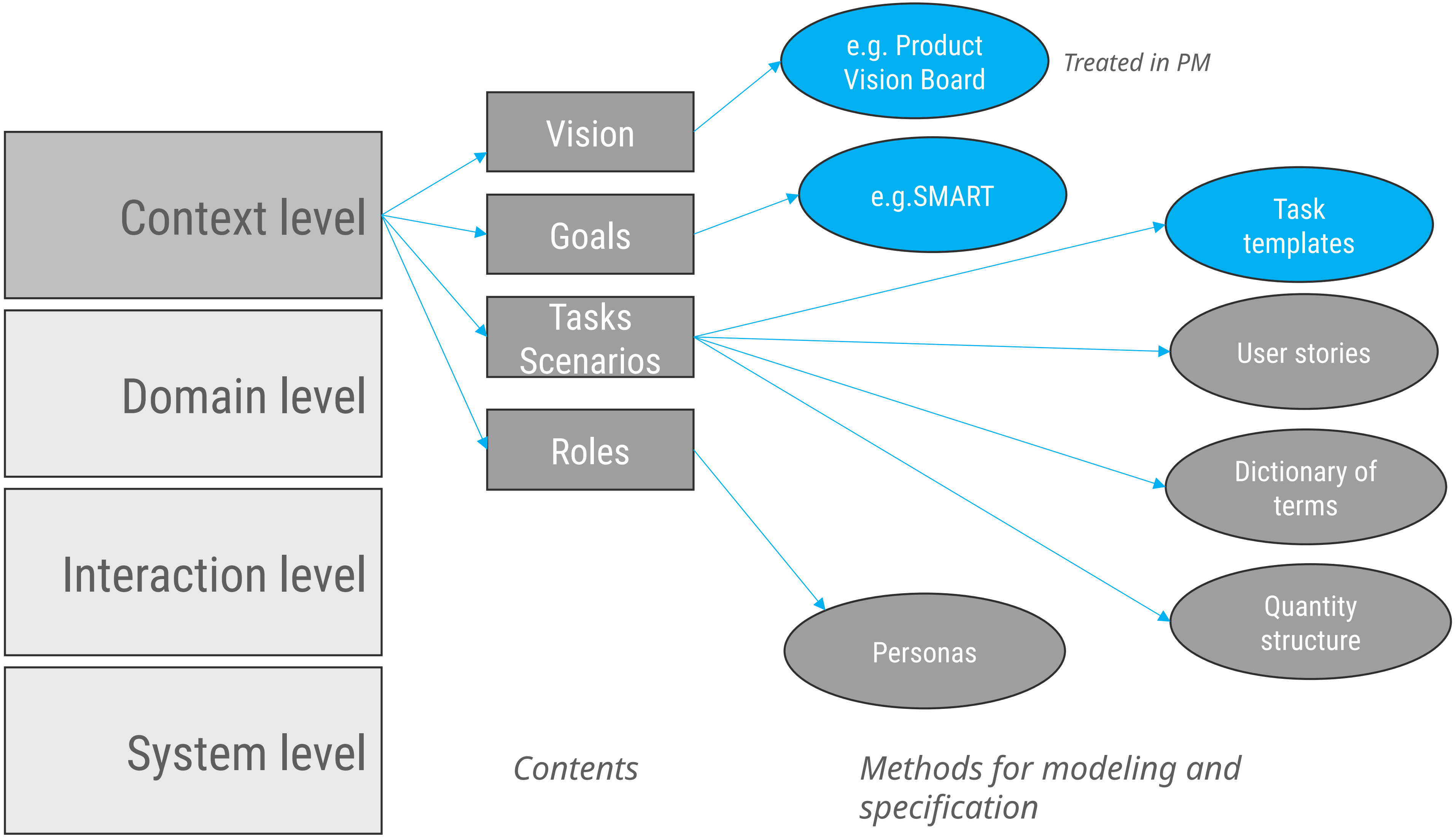
Describes the **context of use of** the software to be (further) developed.

Content and methods at the context level



** The glossary of terms is created from the outset and refined and supplemented as the product life cycle progresses.*

Content and methods at the context level



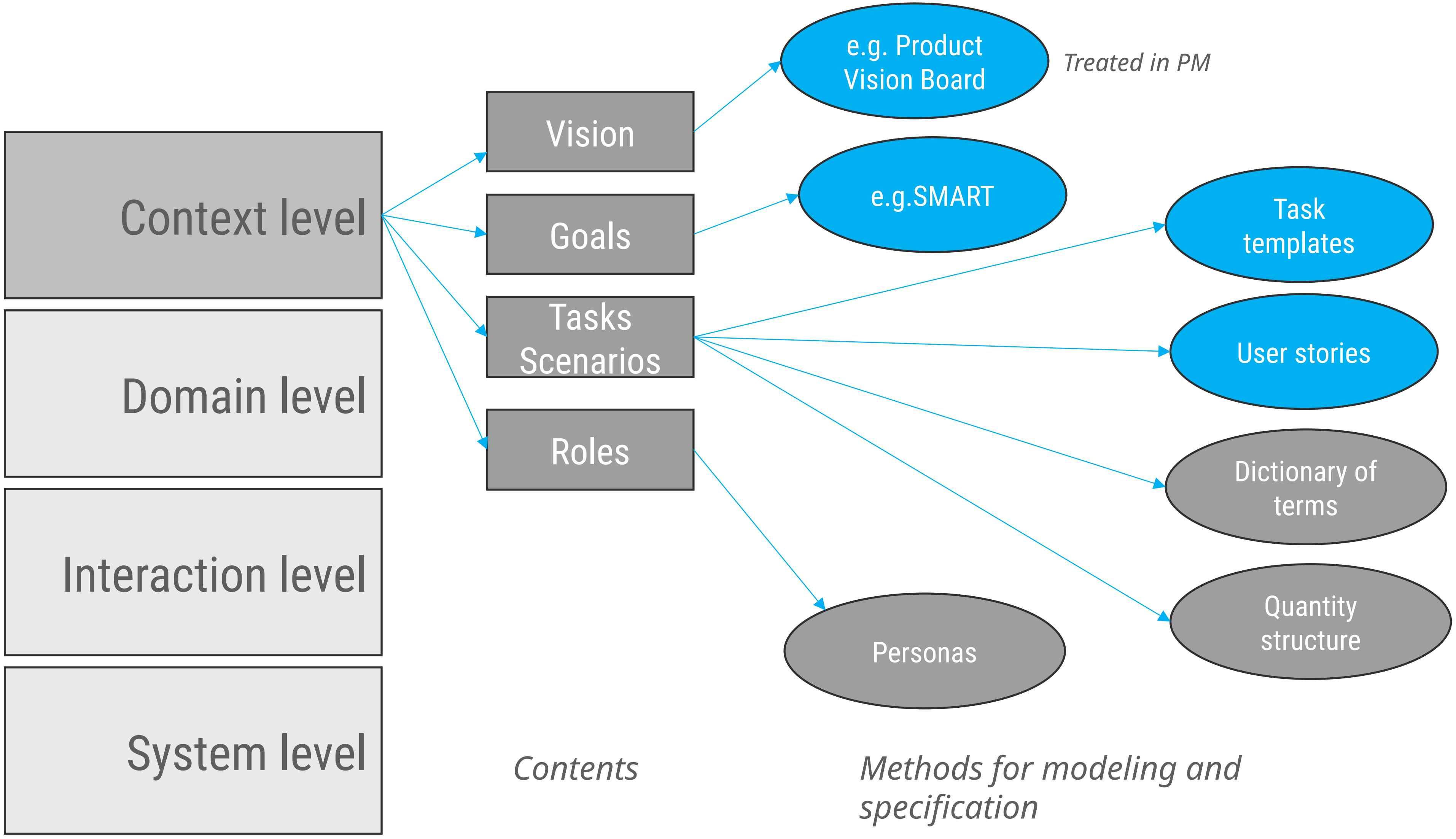
Treated in PM

** The glossary of terms is created from the outset and refined and supplemented as the product life cycle progresses.*

Contents

Methods for modeling and specification

Content and methods at the context level

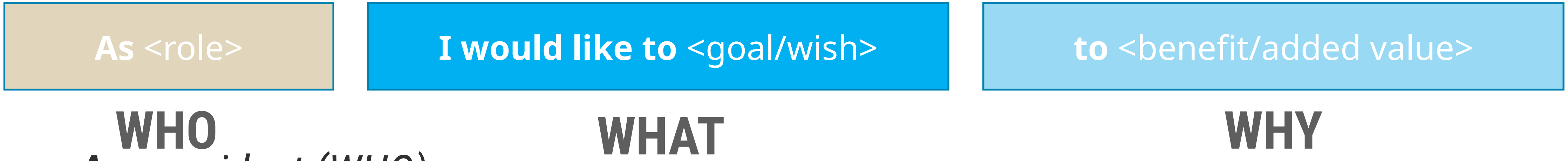


** The glossary of terms is created from the outset and refined and supplemented as the product life cycle progresses.*

TH Aschaffenburg
university of applied sciences

User stories

- Describe desired functionalities or properties of a system from the perspective of the person who needs the functionality or property. The function or feature has added value for the person.
- A typical template for formulating good user stories:



As <role>

I would like to <goal/wish>

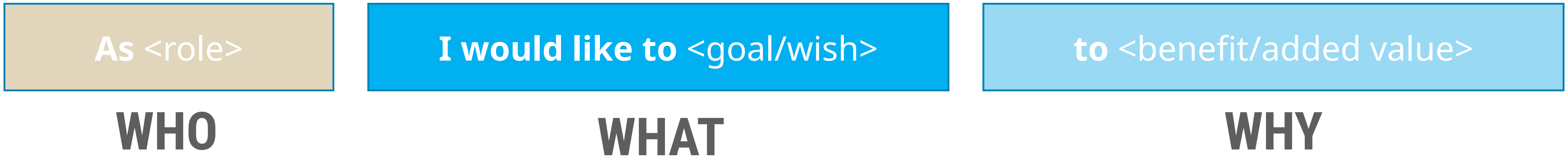
to <benefit/added value>

WHO

WHAT

WHY

  - ***As a** resident (WHO)*
  - ***I would like** keyless access to the building, (WHAT)*
  - ***to** be able to enter the building with your hands full (WHY).*



As <role>

WHO

I would like to <goal/wish>

WHAT

to <benefit/added value>

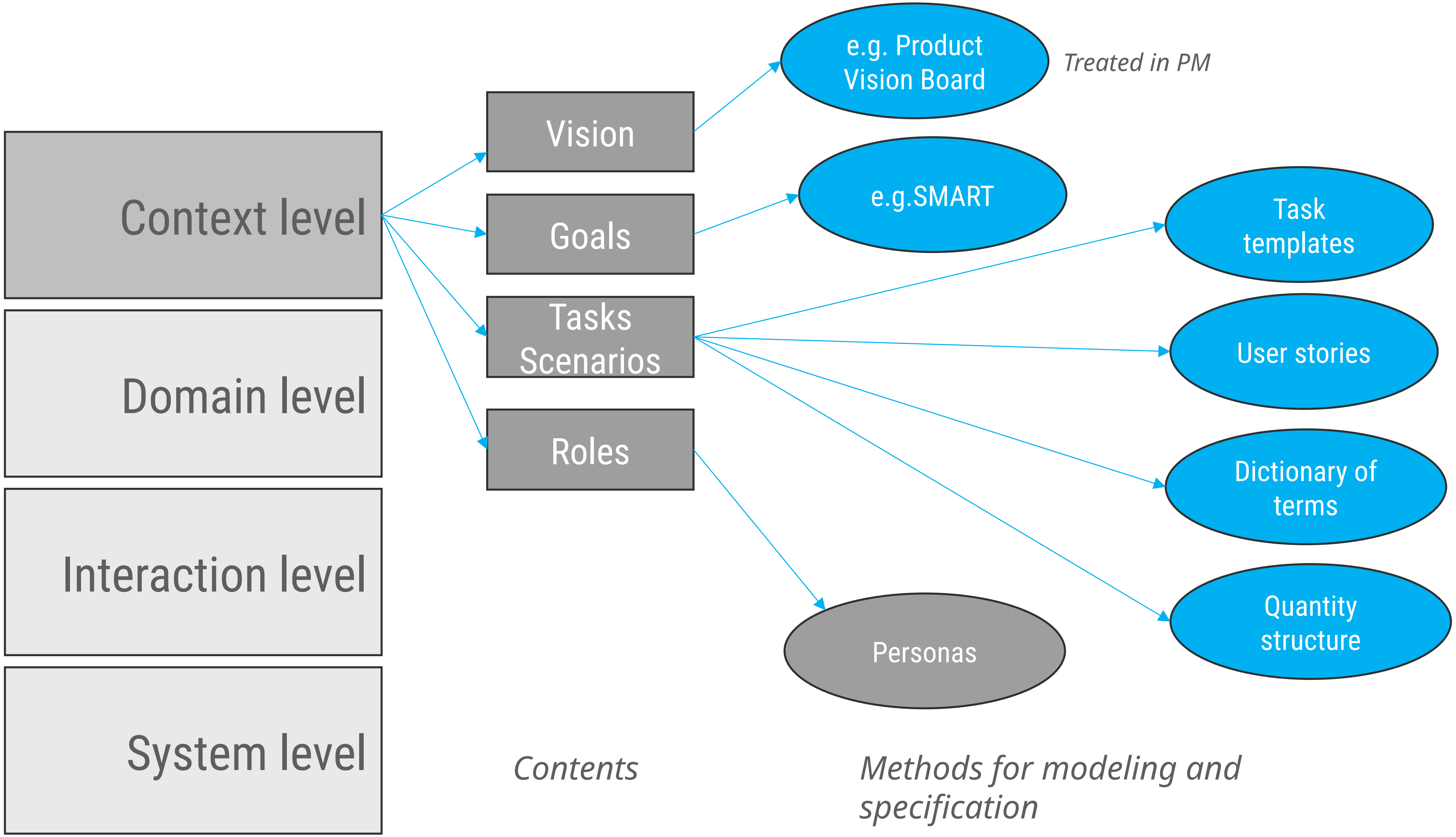
WHY

User stories

- Formulate user stories independently of the solution
  - ***As a*** *customer in the store (WHO)*
  - ***I*** *simply* ***want to*** *pay for my purchases, (WHAT) → Solution is not provided*
  - ***to*** *be able to take the goods (WHY).*

- vs.
  - ***As a*** *customer in the store (WHO)*
  - ***I would like to*** *withdraw money, (WHAT) → Solution is given, solution space extremely limited*
  - ***to*** *be able to take the goods with you (WHY).*

Content and methods at the context level



Treated in PM

Contents

Methods for modeling and specification

** The glossary of terms is created from the outset and refined and supplemented as the product life cycle progresses.*

Dictionary of terms right from the start

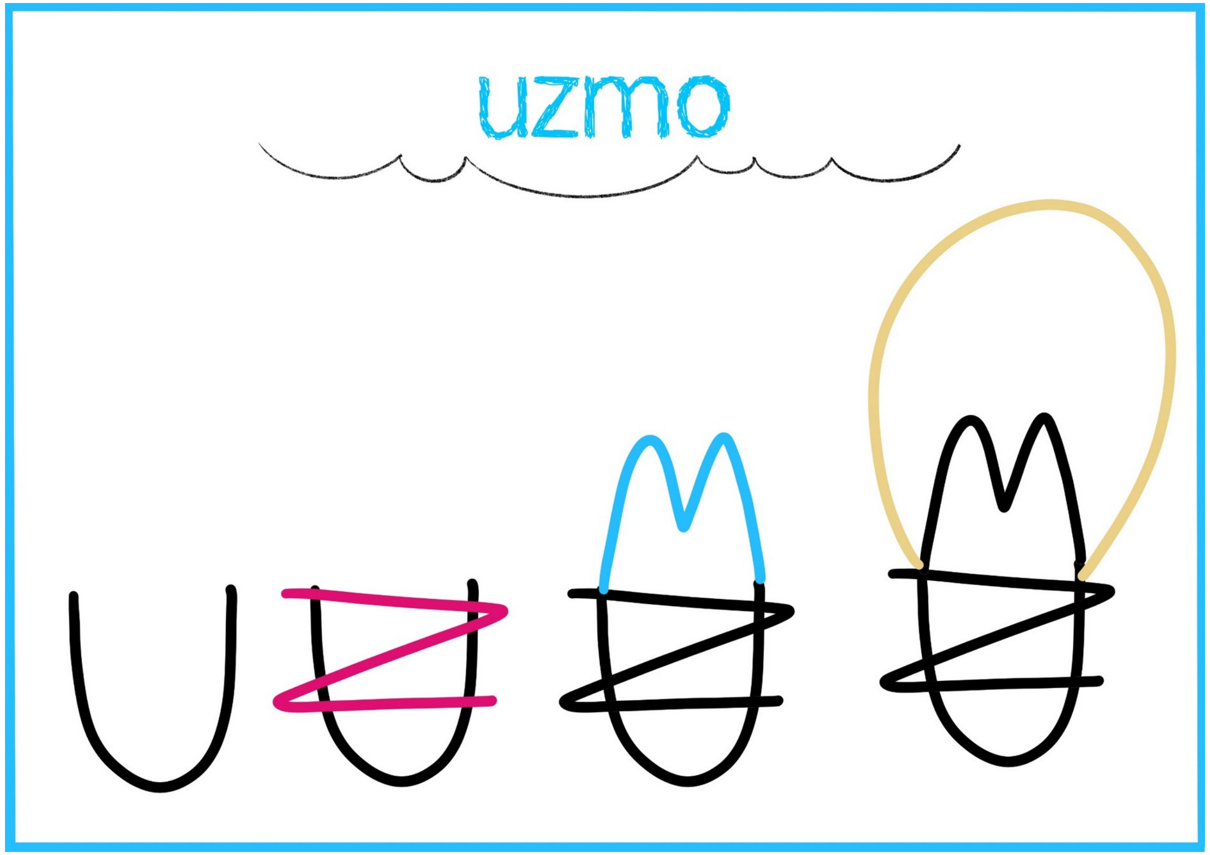
→Helps to keep your bearings in the jungle of terms

- UZMO

Main problems in requirements engineering

→ Communication Problems

- UZMO



Even if you think you have understood everything, you may be in for a nasty surprise.

Dictionary of terms

- Create a glossary of terms right from the start!
- Continue to develop the document.
- The dictionary of terms contains terms that
  - are **important** and could **be interpreted** differently by different people.
  - This often includes **terms that appear to be completely clear at first glance** → See UZMO

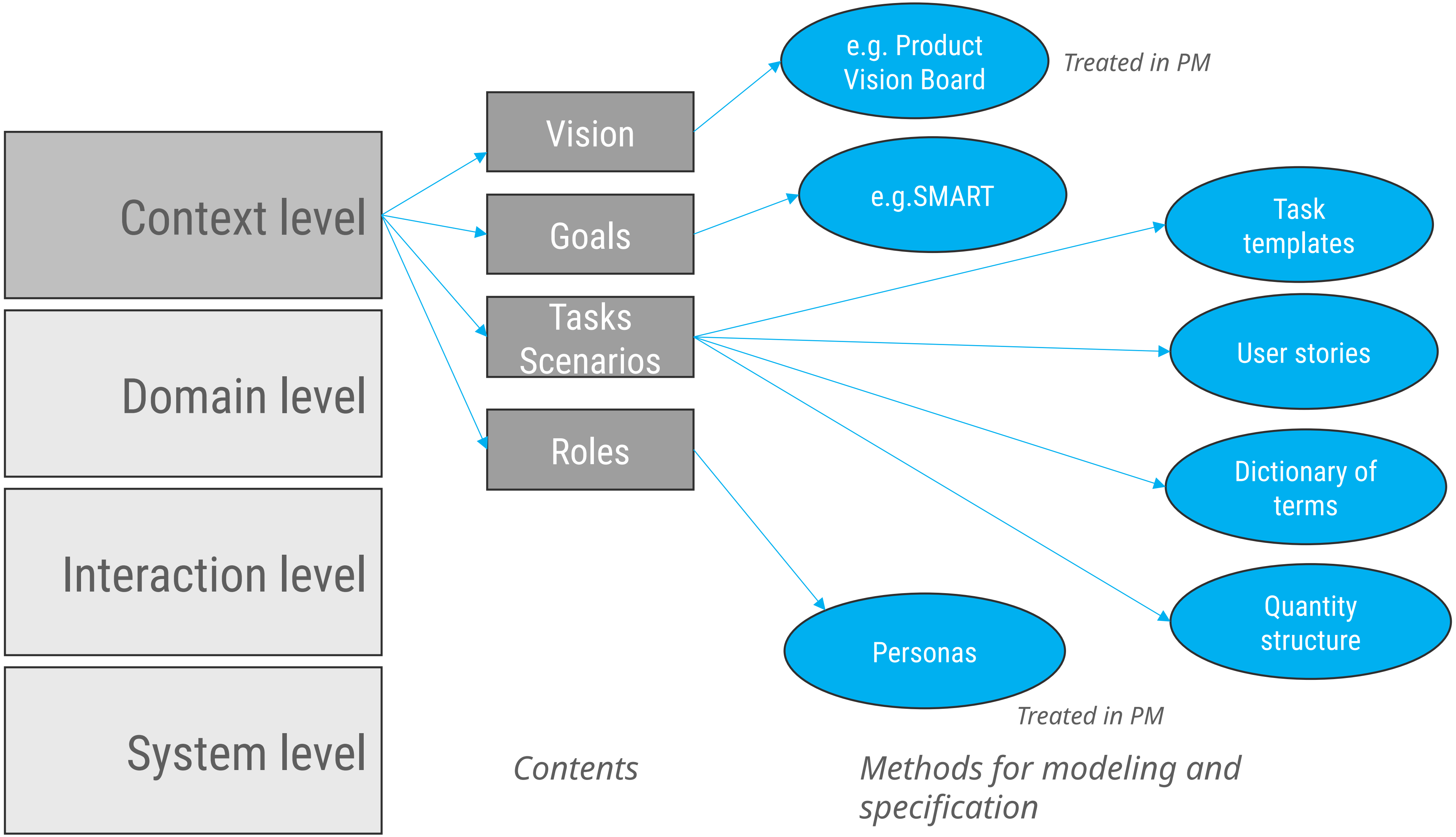
Dictionary of terms

a) Term and synonyms (as defined in the specification)
b) Abbreviation
c) Origin (conversation, document, etc.)
d) Meaning (definition, explanation)
e) Scope (where is this term not applicable?)
f) Validity (temporal, spatial, other)
g) Questions of designation, unambiguity, etc.
h) Uncertainties that have not yet been resolved
i) Related terms (cross-references)

Quantity structure

- Information on the problem size.
- Should be an integral part of the specification.
- All specifications define the **upper limits, lower limits** or typical values of significance.
  - Length PIN
  - Number of transactions/per time unit
  - Rate of change of certain states
  - Etc.

Content and methods at the context level



Treated in PM

Treated in PM

Contents

Methods for modeling and specification

** The glossary of terms is created from the outset and refined and supplemented as the product life cycle progresses.*

Roles and personas

→ See PM lecture

- Describe **actors** of the current and/or target system
- **Roles** describe **user groups**
- **Personas** describe specific representatives of a user group

Description of personas

→ Different templates, all have the following categories of information in common

- **Personalization**: personal data of the individual, incl. picture!
- **Occupational factors**: Details of the individual's working/living environment
- **Relevance**: Details about the interest in the product
- **Education**: Details on education and experience

- Personas contain all the information that is important to understand users in order to be able to put yourself in their shoes.

- Introduction: Requirements
- Requirements Engineering
- Context Analysis
- Requirements specification / functional spec.
- Use-Cases

Disciplines in software engineering

Focus today

Basic topics

Configuration management | Documentation |
Knowledge management | People in the SWE process and digital ethics | Tools

Development

Requirements

- Context analysis
- Requirements Engineering

Design

- Architecture
- Detailed design

Implementation

QualityMgt.

Quality assurance and testing

- Test, inspection, metrics

Processes and procedure models

- Improvement, process model, maturity levels

Evolution

- Roll-Out
- Operation
- Maintenance
- Further development
- Reuse
- Reengineering
- Change management

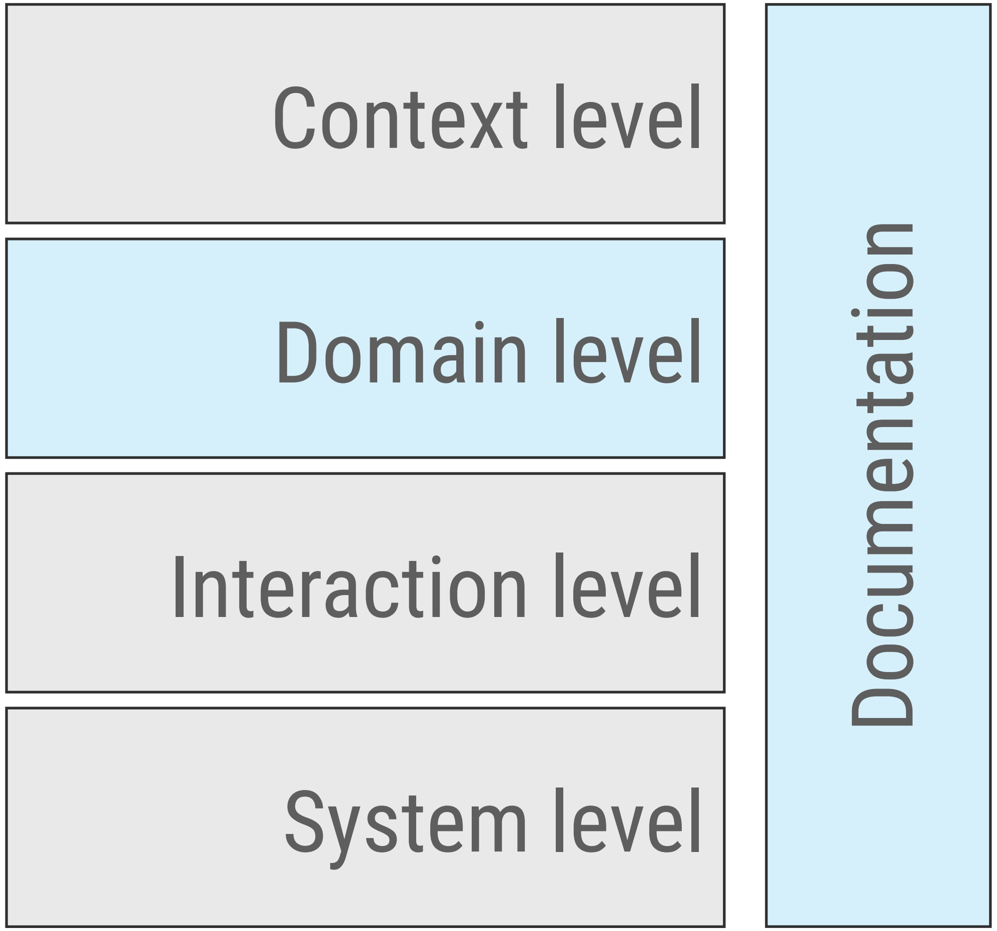
Management

- Strategy
- Economy
- Team
- Dates
- Risks
- Customer, client/contractor
- Innovation

Learning Objectives

- You can explain why **documentation** is generally important.
- You can explain the benefits of a **requirements specification** in the SWE process.
- You can give examples of typical contents of a **technical document**.
- You will be able to explain the terms **requirements specification** and **functional specification**.
- You can name typical contents of a **requirements specification**.
- You can discuss how detailed a document should be.

Documentation is important at all levels



Question

Why is documentation important in general?

Why is documentation generally important?

→ Documentation also as a means of communication

- Direct communication is extremely important, but there are some good reasons why documentation is important:
- "Writing down" forces us to **formulate** certain things **precisely**
- **Common understanding** (not only in the "heads")
- Documentation particularly important
  - **Longevity** of the content
  - **Legal** relevance
  - **Complexity** of the content
  - **Accessibility** for many stakeholders

Question

Why is the specification of requirements important?

Benefits of the specification

Ludewig, 2013

Specification is the basis for various activities in the software development process

- coordination with the customer and marketing,
- the design and implementation,
- the user manual,
- the test preparation,
- the acceptance,
- reuse,
- the clarification of later objections, recourse claims, etc,
- a later re-implementation.

There is no specification in the head! [Ludewig, 2013]

Question

What belongs in a technical document?

What belongs in a technical document?

1. Introduction

1.1 Purpose

- Who craeated the document and how?
- Who should read this document?
- Who is bound by this document (scope of use)?

1.2 Summary

- Scope and stakeholders

1.3 Definitions and Abbreviations

- Definition of terms and abbreviations (incl. Glossary)

Source [Paech, 2021]

What belongs in a technical document?

1.4 References, Standards, and Rules

List of referenced documents

1.5 Overview

Content and structure of this document

2. – X. Core Content

X+1. Summary

X+2. Appendix

X+3. Index (incl. table of contents)

Source [Paech, 2021]

Question

What are typical documents that are created during the requirements analysis?

Requirements Specification (German: Lastenheft)

→ Basis for tendering and contract design in a formal, classic project setting

- Core: functional and non-functional requirements that the system to be developed MUST or CAN fulfill.
- Also: Requirements, delivery conditions and acceptance criteria that are relevant for the subsequent acceptance of the system.
- Particularly relevant for tenders and when drafting contracts.

Requirements Specification (German: Lastenheft)

→ Definitions

Lastenheft - *compilation of all the* ***client****'s requirements with regard to the scope of delivery and services. The requirements specification contains the requirements from the user's point of view, including all environmental and boundary conditions.*
These should be quantifiable and verifiable. The specifications define WHAT is to be solved and WHY. The specifications are drawn up by or on behalf of the client. It serves as the basis for tenders, offers and/or contracts.

After VDI2519, P. 2

Lastenheft: The entirety of the requirements specified by the client for the deliveries and services of a contractor within an order.

After DIN69905, P. 3

Functional Specification (German: Pflichtenheft)

→ Definitions

Functional specification: *Description of the implementation of all requirements of the requirements specification. The functional specification contains the requirements specification. User specifications are detailed and the implementation requirements are described. The functional specification defines HOW and WITH WHAT the requirements are to be implemented. As a rule, the functional specification is drawn up by the* ***contractor*** *after the order has been placed, if necessary with the cooperation of the client. When drawing up the functional specification, the contractor checks that the requirements specified in the requirements specification are consistent and can be implemented.*

VDI2519, P. 2

Functional specification: *Implementation specifications drawn up by the contractor based on the implementation of the requirements specifications provided by the client.*

DIN69905, P. 3

Question

Consider the following scenario:

While creating the functional specification, we find out that that the project is going to be at least 3 times more expensive to implement compared to our initial expectation.

How do we deal with this?

Full requirements and functional specification

→ Combine requirements specification and functional specification

- Pragmatic approach to avoid inconsistencies between two redundant documents.
- Different scenarios can be found in practice, depending on how formal a project is set up and how agile the development is
  - Rather rare: Requirements and specifications are managed in parallel
  - In the initialization phase, the requirements specification serves as the basis for tenders and contractual arrangements, later agreement on a joint document (typically: requirements specification)
  - A joint document is created and further developed from the outset.
  - Client is the product owner and is responsible for creating and maintaining the documents and the backlog.

High,
classic PM

Formalization degree

Low,
agile PM

Rather
project-based
SWE

Rather
product-based
SWE

Example of a specification

2. **Description of the product context**

   2.1 Purpose of the product

   Product vision, business goals to be achieved

   2.2 Users and target audience

   Roles, personae

   2.3 Processes in context:

   Current status: Task description in which the Product required (incl. tasks, roles)

   Target state: tasks, roles, user stories

Source [Paech, 2021]

Example of a specification

3. **System requirements**

   3.1 Overview of the system

   Course granular summary of the system, e.g. in a Use case diagram

   3.2 Functional requirements

   Process and behavior description (user stories, use cases, system functions, data specification, UI structure, depending on the context, behavioral and structural digrams)

   3.3 Non-functional requirements

Source [Paech, 2021]

Example of a requirements specification

4. Project requirements

4.1 Overview of procedure and process

4.2 Assumptions and dependencies

4.3 Acceptance

Source [Paech, 2021]

Question

What determines how detailed a document should be?

What determines how detailed a document should be?

Scope and level of detail of a document

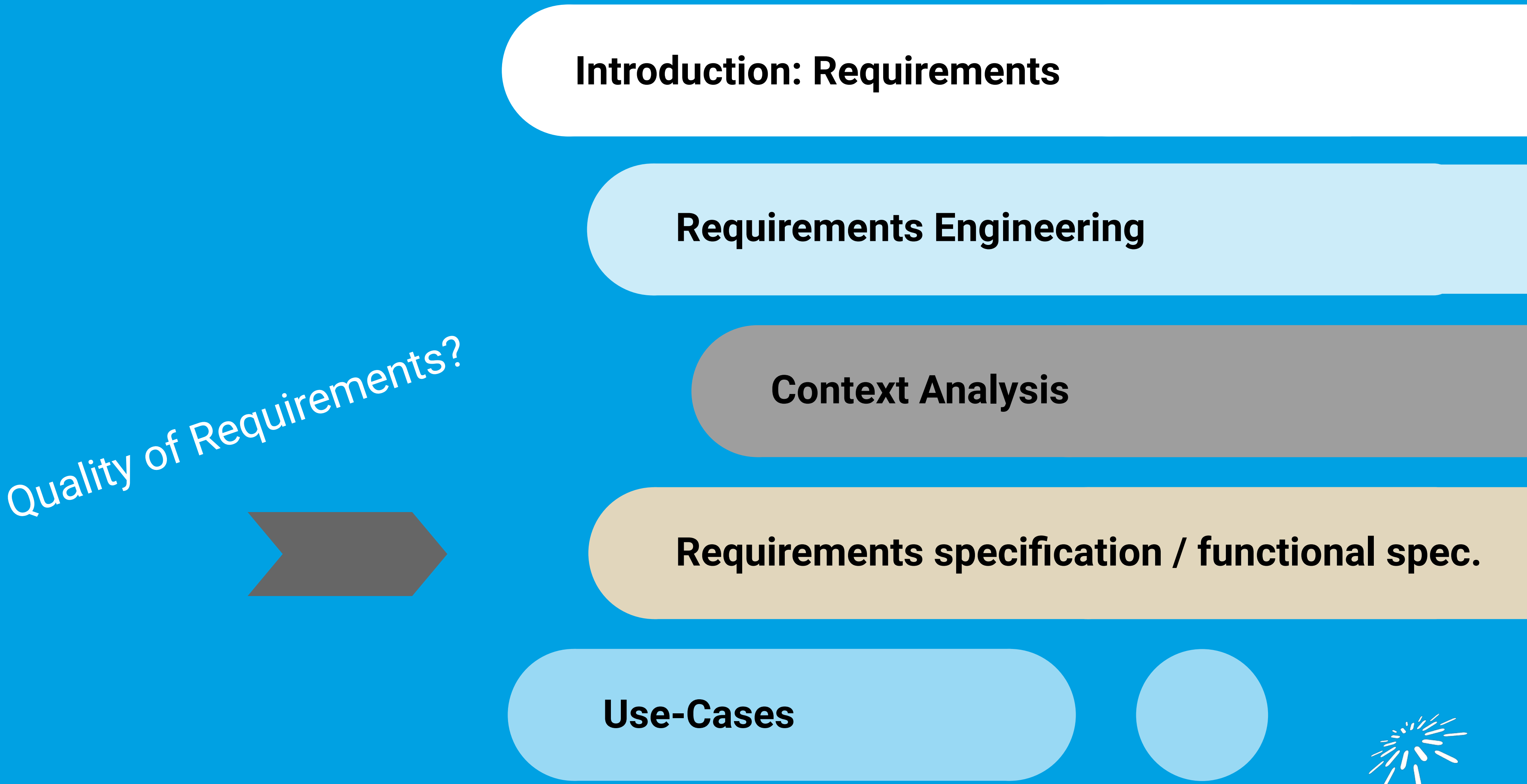
- are dependent on the **risk** that would be a result of the required information not being written down and unable to be found
- Depends on **how often** and **how many** readers need the information to clarify questions
  - The more people need a piece of information or the more often a piece of information is needed, the more important it is to write it down
- High **risk of errors occurring** due to missing or misleading information
- High **probability of change**
- Etc.

Requirements specification, functional specification

→ Example templates

- VModel - XT (structure):
  - Lastenheft
  - Pflichtenheft

- SOPHISTS:
  - Product Specification Template
  - NFR template



Introduction: Requirements

Requirements Engineering

Context Analysis

Quality of Requirements?

Requirements specification / functional spec.

Use-Cases

TH Aschaffenburg
university of applied sciences

Learning Objectives

- You can use examples to explain what the **criteria for good requirements** are.
- You can formulate linguistically clear requirements by applying the "little etiquette for requirements".

Question

When is a requirement good?

Criteria for good requirements documents

According to ISO/IEC/IEEE 29148:2018

- Complete
  - All relevant requirements are fully documented (including errors and exceptions)
  - Formal completeness (graphics, tables, list of sources and abbreviations, etc.)
- Consistent
  - Free of contradictions, consistent terminology
- Realizable
  - Realizable on time and within budget
- Validatable
  - We can check that needs are met

Criteria for good requirements documents

According to ISO/IEC/IEEE 29148:2018

- Understandable → For the readers of the document!!!
  - Depending on the target audience, a document may or may not be comprehensible.
  - Stakeholder: correct, necessary, appropriate: Is my order/task accurately reflected? Am I getting what I asked for?
  - Developers: Are the requirements feasible?
  - Testers: Are the requirements testable? Can test cases be derived?
  - Project manager: Are the requirements feasible?
  - Etc.

Language problem: fuzzy requirements

So you open the lock at the main entrance in the morning?

Yes, I told you.

Every morning?

Of course.

Even at the weekend?

No, the entrance is closed at weekends.

And during the company vacations?

Of course it stays closed.

And if you are ill or on vacation?

Then Mr. X does it.

And what if Mr. X also drops out?

Then at some point a customer knocks on the window because he can't get in.

What does "morning" mean? ...

Source [Ludewig, 2013]

Introduction: Requirements

Requirements Engineering

Context Analysis

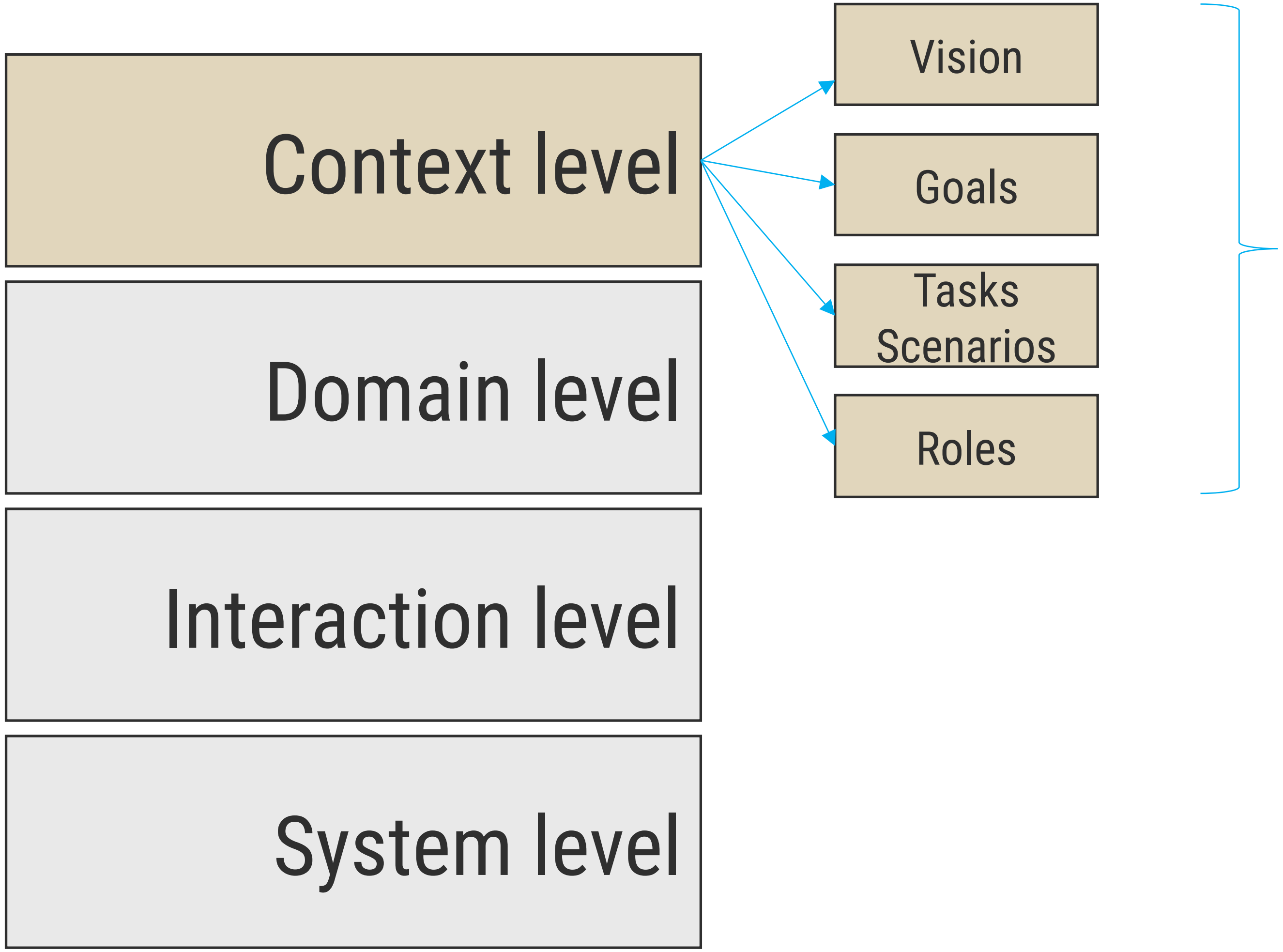
Requirements specification / functional spec.

Use-Cases

Learning Objectives

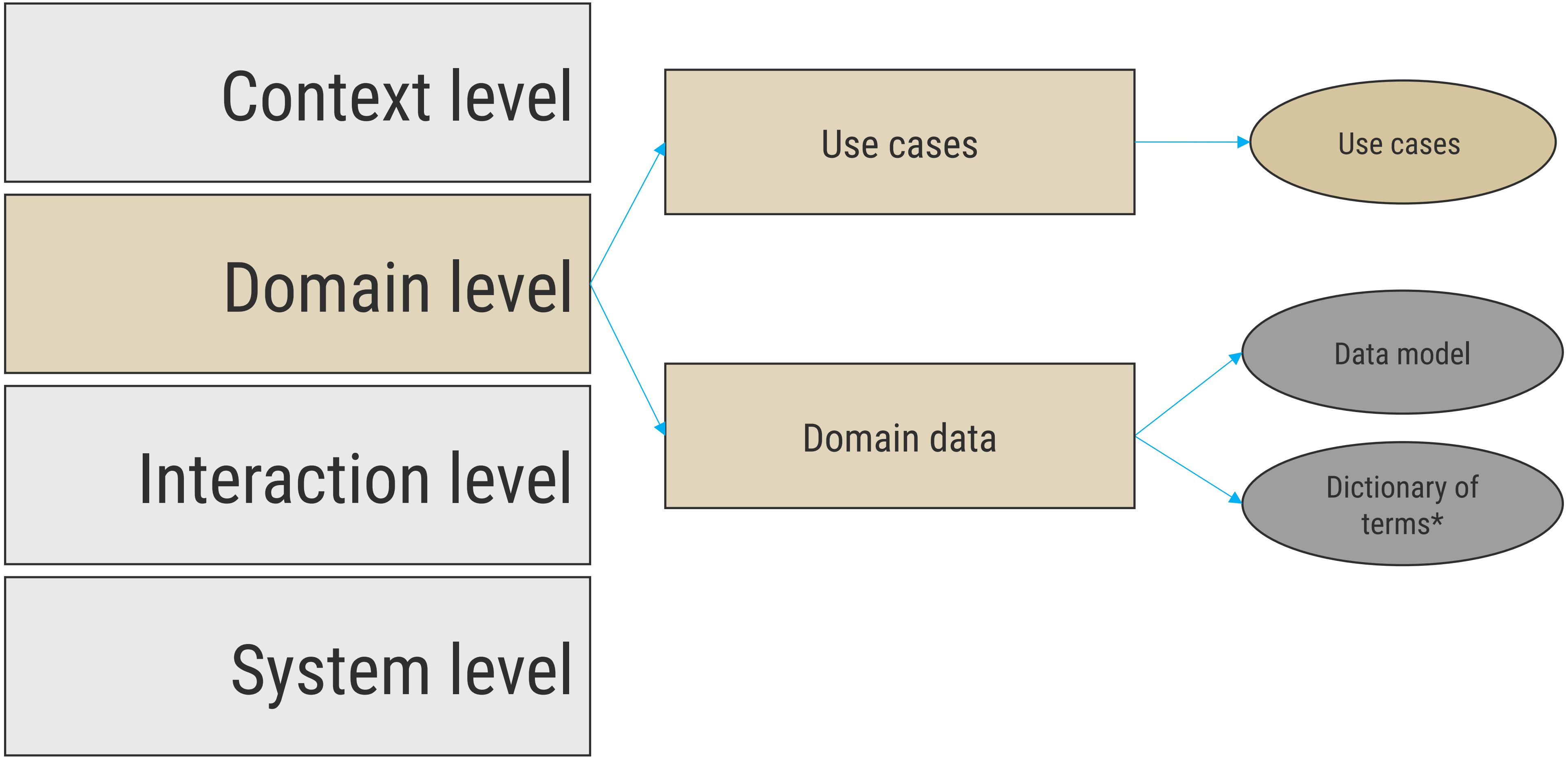
- You can identify and specify **use cases**.

Modeling levels



Describes the **context of use of** the software to be (further) developed.

Requirements exist at different **levels of abstraction**



** The glossary of terms is created from the outset and refined and supplemented as the product life cycle progresses. Especially when the application domain, i.e. the "specialist area" for which the software is to be (further) developed, becomes detailed, it is essential to keep the BL up-to-date, consistent and complete for the clear definition of important terms.*

Use cases

In general

- German: Anwendungsfälle
- A use case specifies a function of the system that provides a **business value** for at least one actor from the system context.
- Actors
  - **People** in a **specific** role
  - **Neighboring systems** that interact with the system when the use case is executed.
- The system is a **black box**, i.e. only the externally visible behavior is described.
- **Completed** system processes

Black box vs. white box

Use cases

Where do the use cases come from?

- Derived **from context analysis**
  - Which task or user story should be supported by a system?
  - Which system function does the system use to support the respective step?
  - Is there already a system function that supports the step? Does this need to be added?
  - Which system functions require the data you identified in the context analysis?
- Use cases/use case diagrams can **detail user stories** and relate them to each other.

Use cases

How do you describe use cases?

Graphical & textual

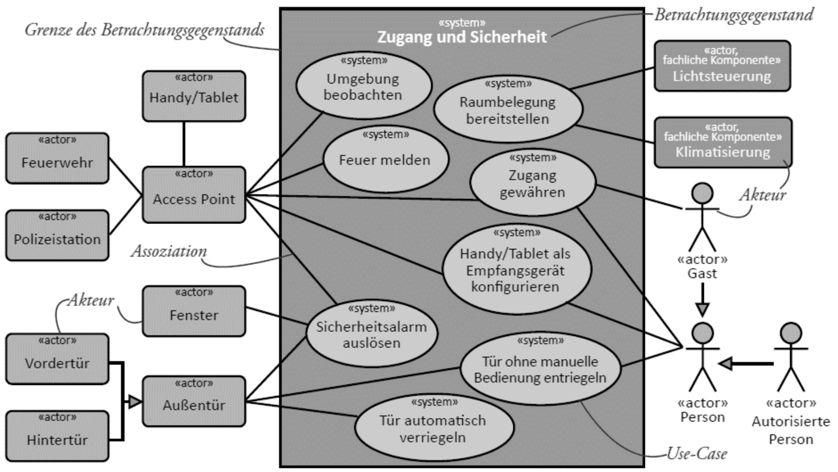
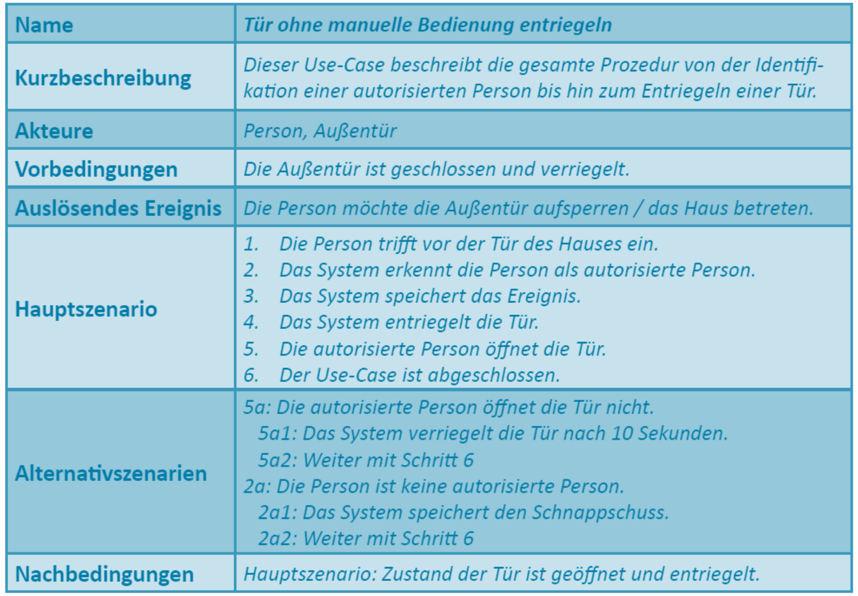


Abbildung 18.4: Use-Cases der fachlichen Komponente Zugang und Sicherheit

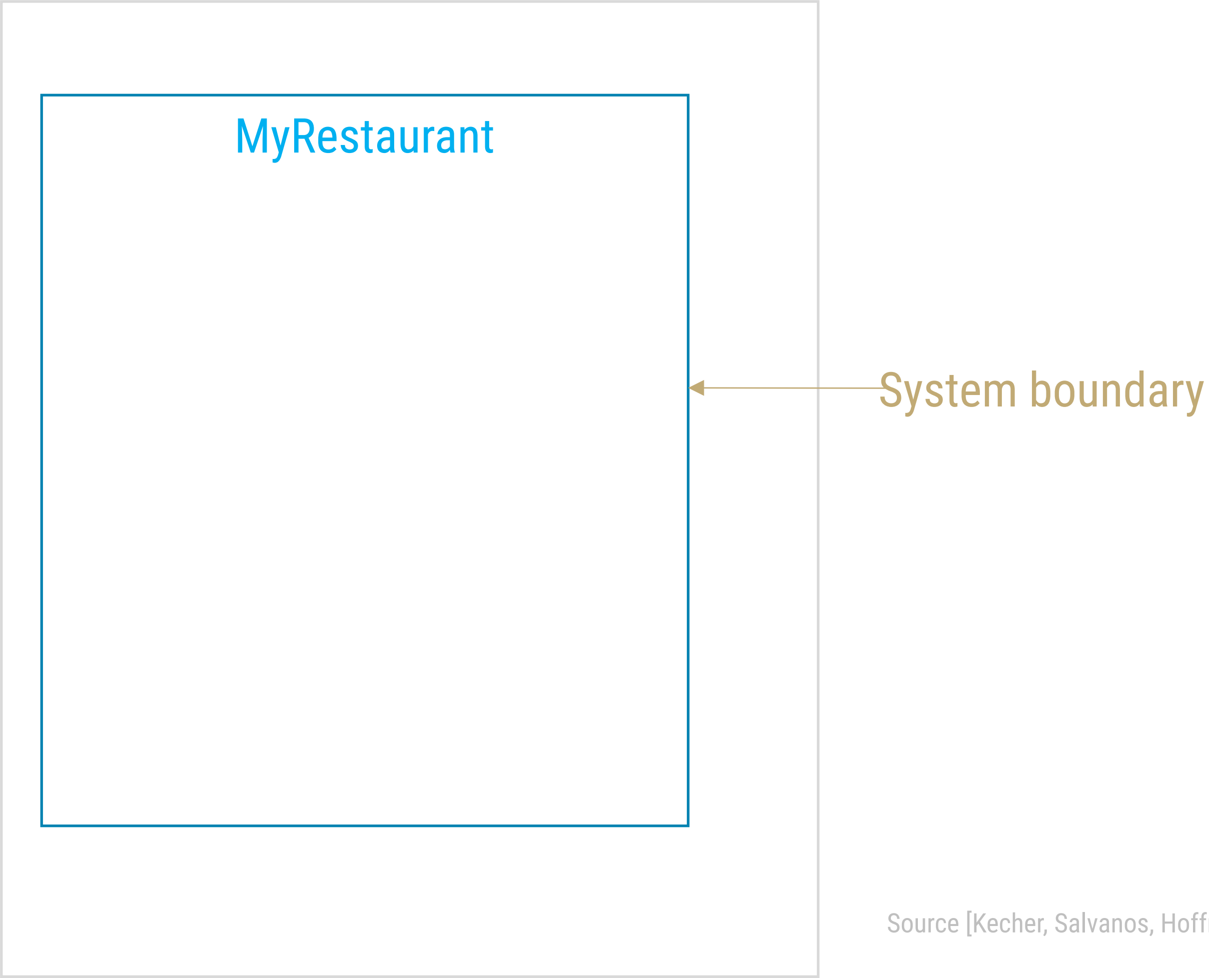


Name	***Tür ohne manuelle Bedienung entriegeln***
Kurzbeschreibung	*Dieser Use-Case beschreibt die gesamte Prozedur von der Identifikation einer autorisierten Person bis hin zum Entriegeln einer Tür.*
Akteure	*Person, Außentür*
Vorbedingungen	*Die Außentür ist geschlossen und verriegelt.*
Auslösendes Ereignis	*Die Person möchte die Außentür aufsperren / das Haus betreten.*
Hauptszenario	*1. Die Person trifft vor der Tür des Hauses ein.* *2. Das System erkennt die Person als autorisierte Person.* *3. Das System speichert das Ereignis.* *4. Das System entriegelt die Tür.* *5. Die autorisierte Person öffnet die Tür.* *6. Der Use-Case ist abgeschlossen.*
Alternativszenarien	*5a: Die autorisierte Person öffnet die Tür nicht.* *5a1: Das System verriegelt die Tür nach 10 Sekunden.* *5a2: Weiter mit Schritt 6* *2a: Die Person ist keine autorisierte Person.* *2a1: Das System speichert den Schnappschuss.* *2a2: Weiter mit Schritt 6*
Nachbedingungen	*Hauptszenario: Zustand der Tür ist geöffnet und entriegelt.*

Abbildung 18.5: Use-Case-Beschreibung zu Tür ohne manuelle Bedienung entriegeln

Use cases → Notation elements

System boundary



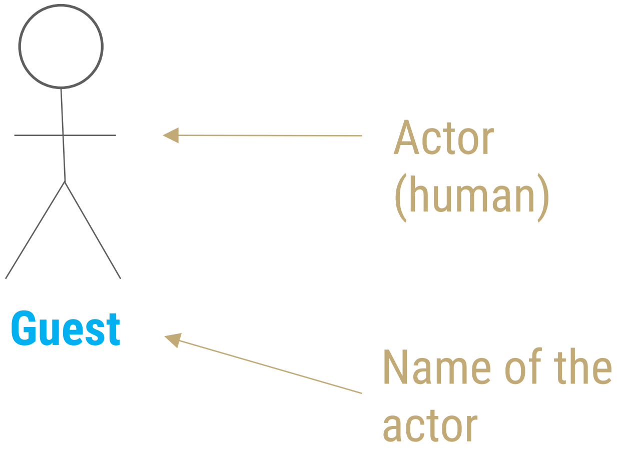
The system boundary refers to a system that contains the defined use cases that interact with the user.

Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

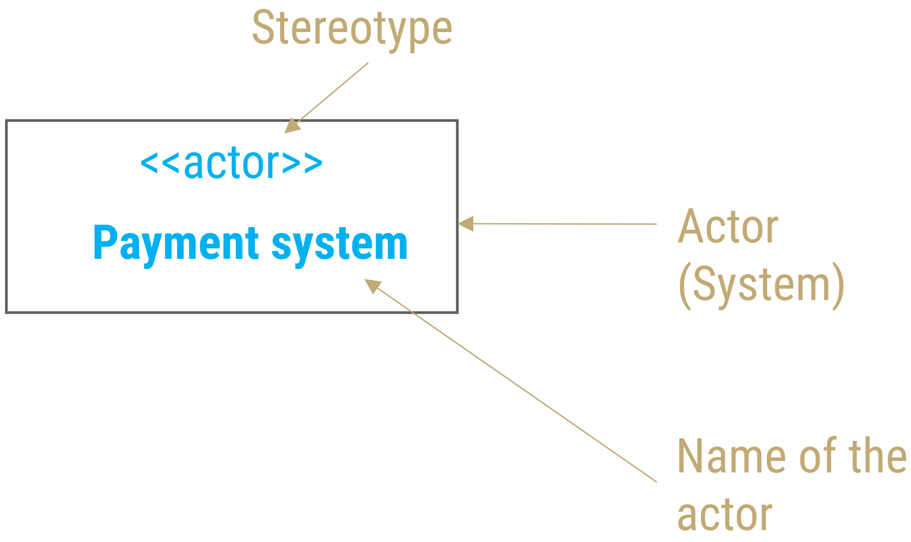
Use cases → Notation elements

Actors

Human actors



Systems

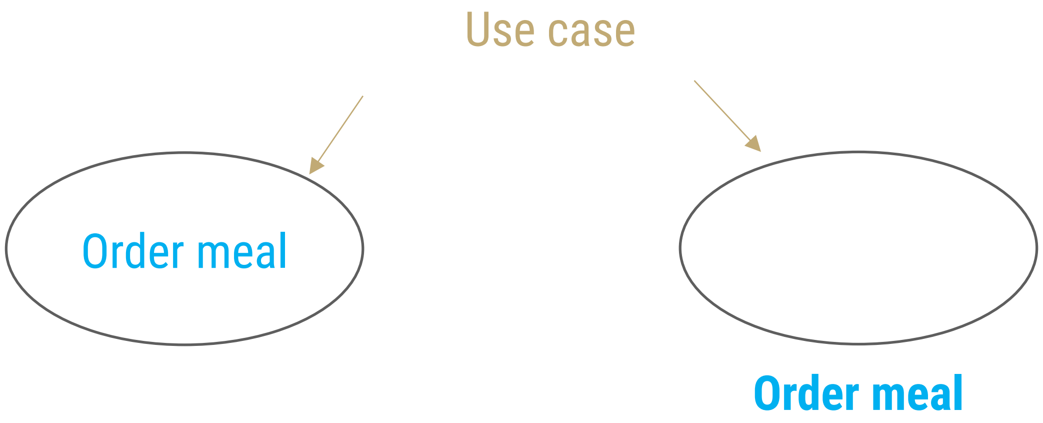


An actor *models a type or role that an external user or an external system assumes during interaction with a system.*

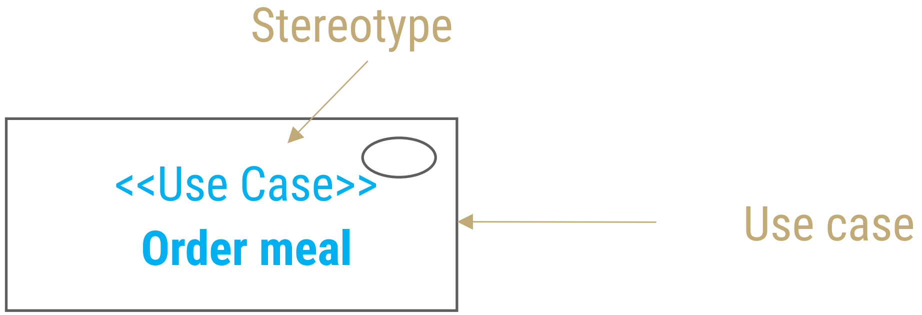
Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

Use cases → Notation elements

Use case



Alternative modeling (rather rare)

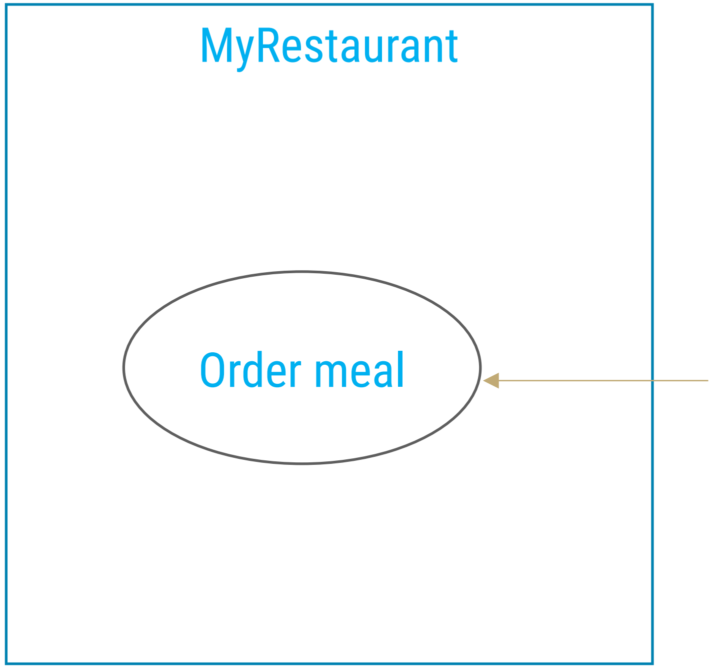


A use case *specifies a closed set of actions that are provided by a system and bring a recognizable benefit for one or more actors.*

Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

Use cases → Notation elements

Use case ("usual notation")



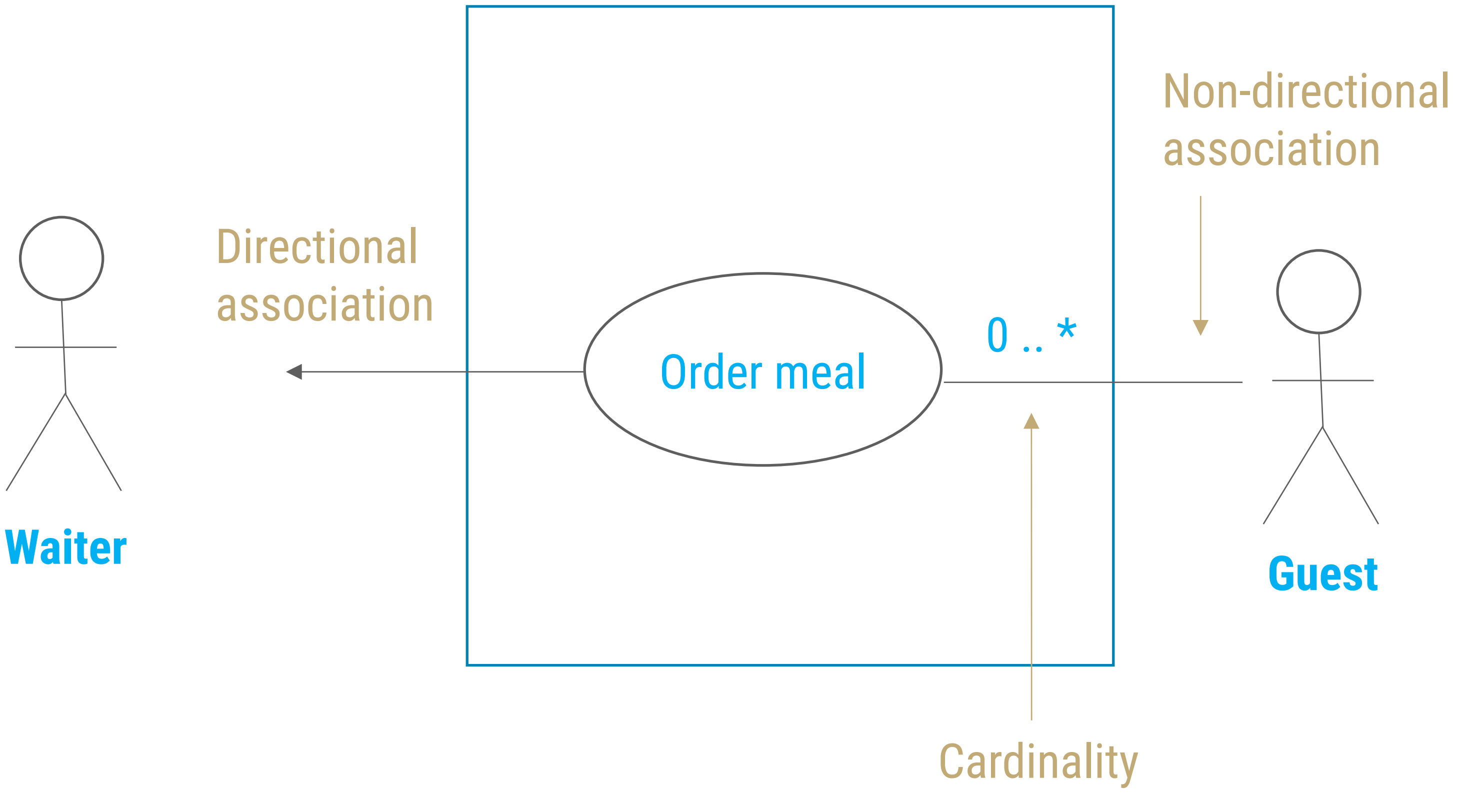
Use case within
the system boundary

***A use case** specifies a closed set of actions that are provided by a system and bring a recognizable benefit for one or more actors.*

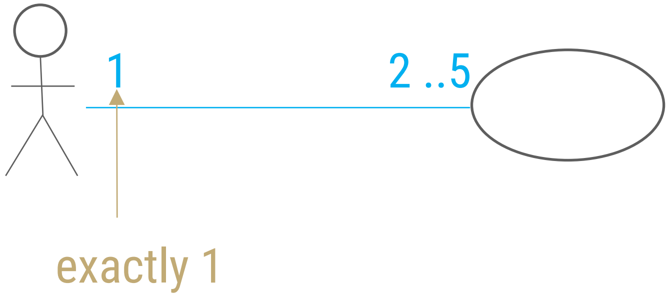
Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

Use cases → Notation elements

Association



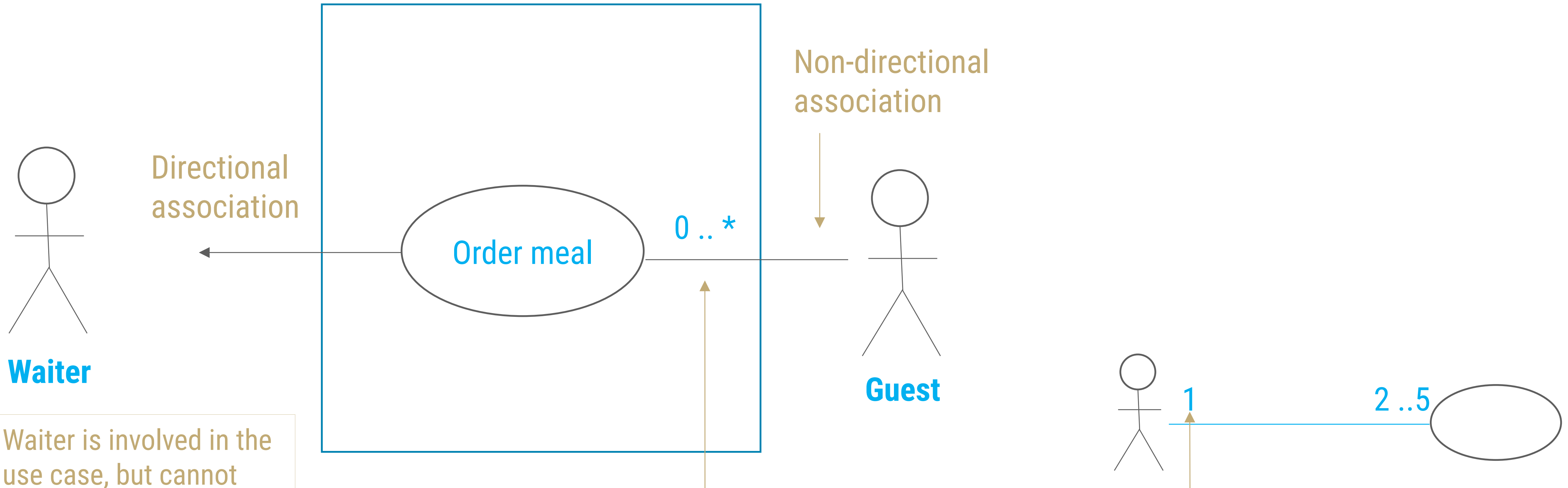
*Association models a relationship between actors and use cases in use case descriptions. Associations can be **non-directional** or **directional** and contain **cardinalities**.*



Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

Use cases → Notation elements

Association



Cardinality

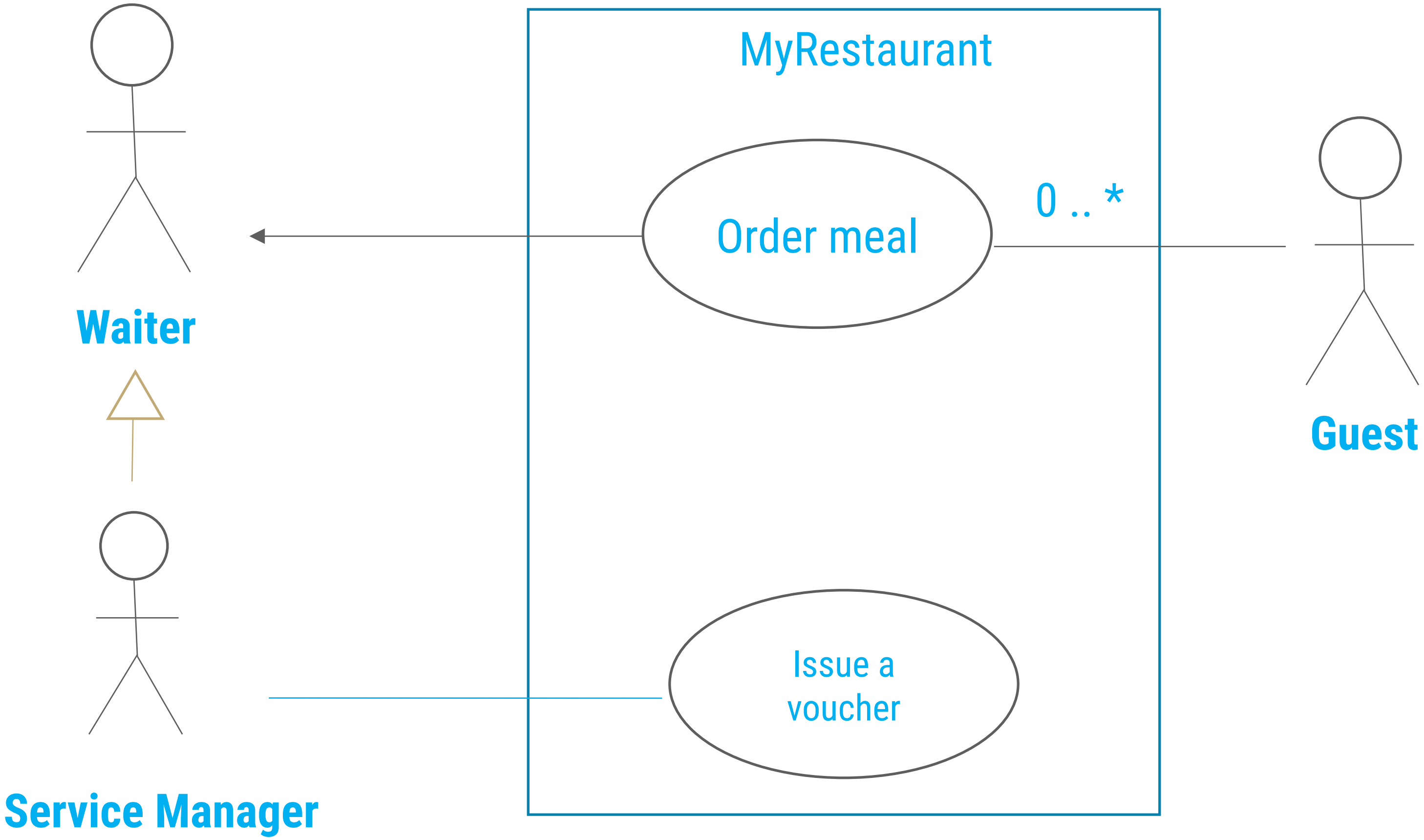
exactly 1

Waiter is involved in the use case, but cannot initiate it himself.
Only the guest can trigger the use case.

Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

Use cases → Notation elements

Generalization/specialization

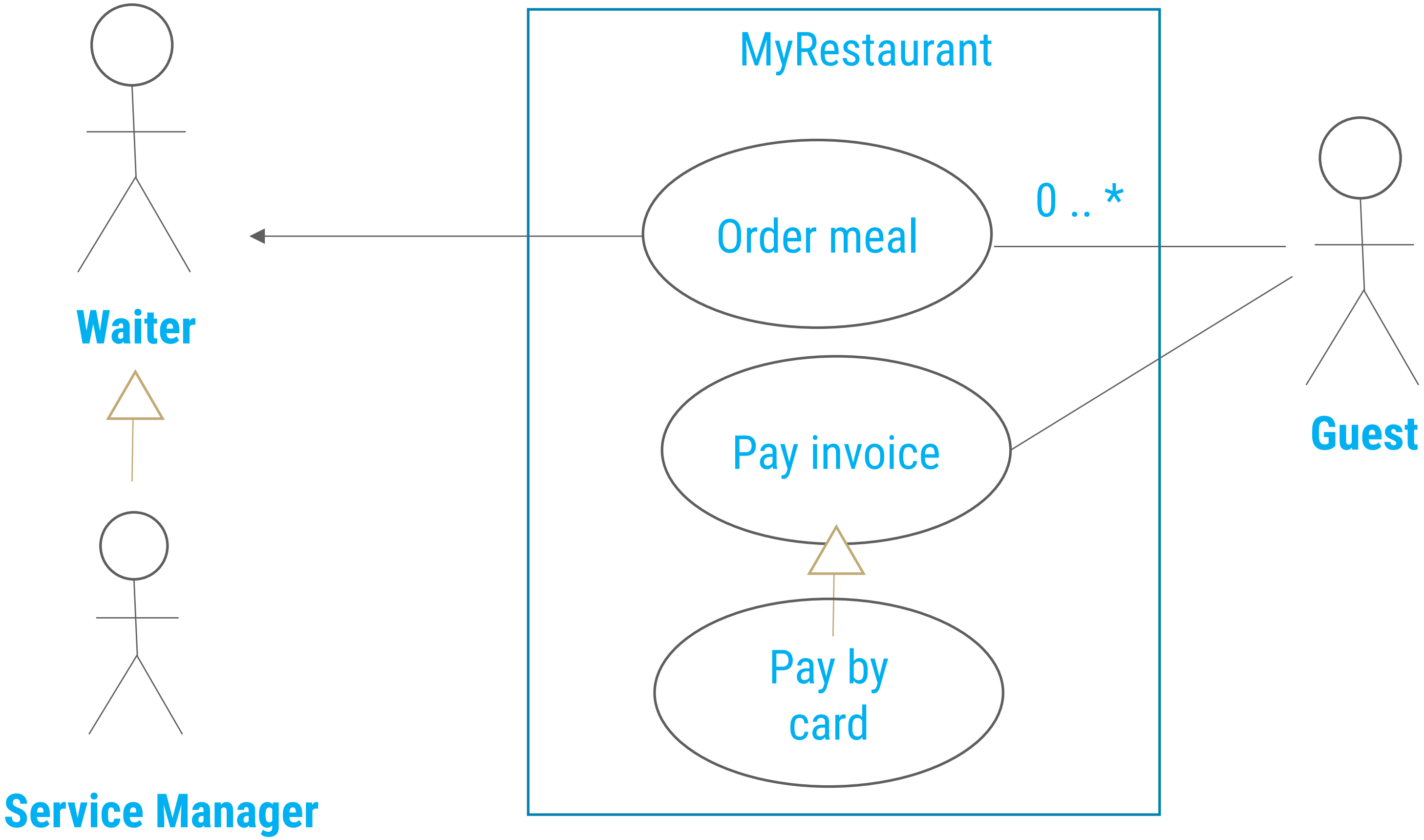


Generalization describes an inheritance relationship between actors or use cases.

Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

Use cases → Notation elements

Generalization/specialization

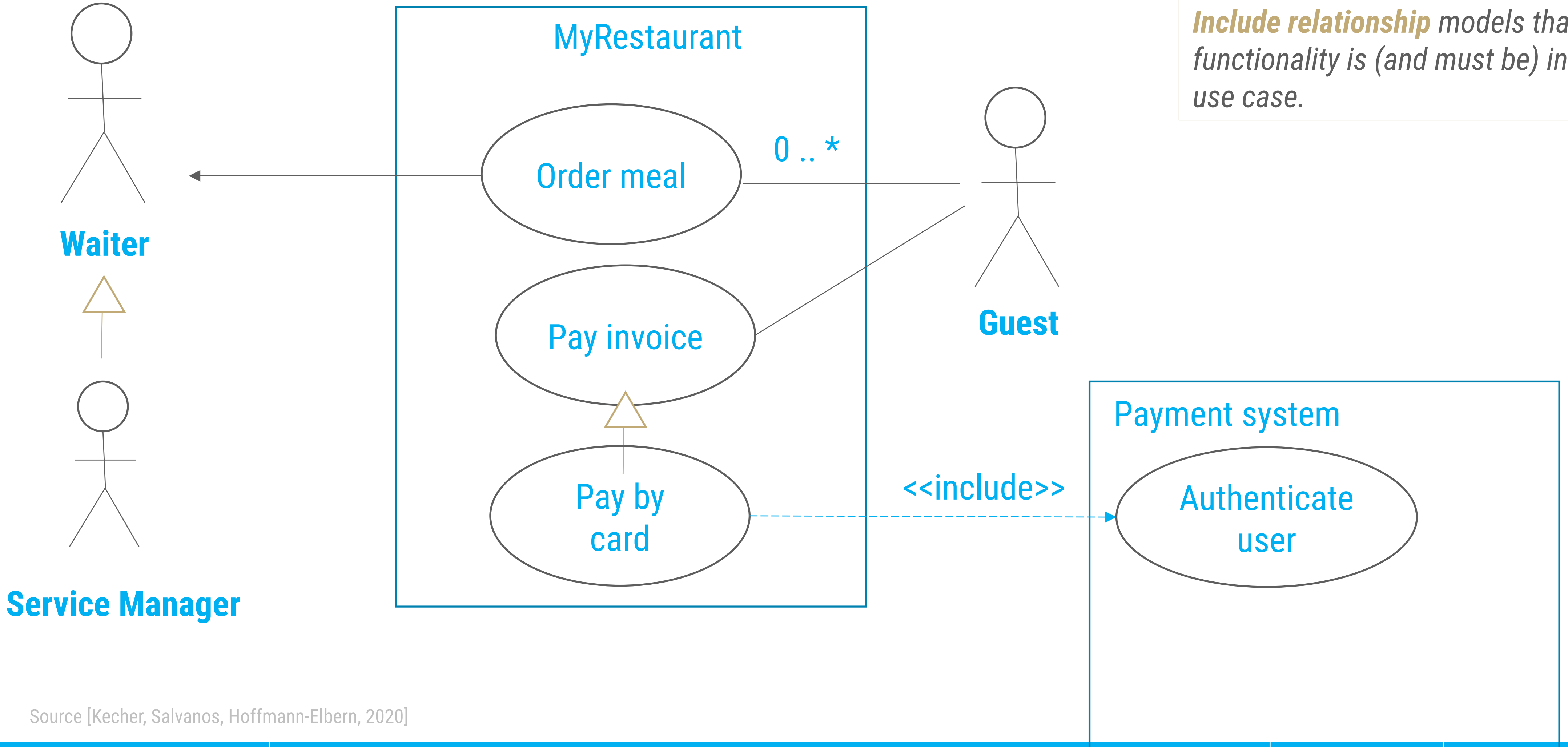


Generalization describes an inheritance relationship between actors or use cases.

Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

Use cases → Notation elements

Include relationship

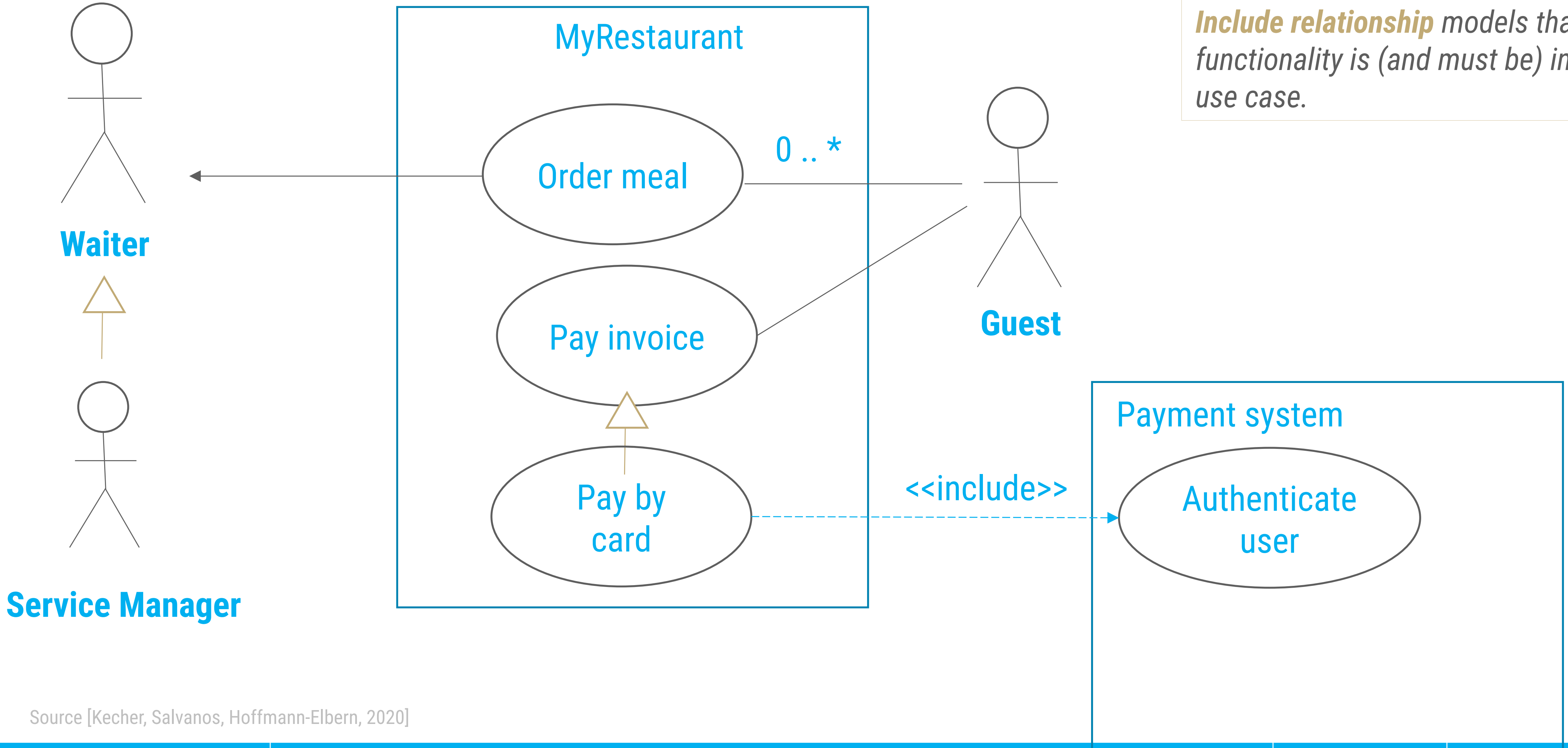


***Include relationship** models that a certain functionality is (and must be) included in a use case.*

Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

Use cases → Notation elements

Include relationship: MUST relationship

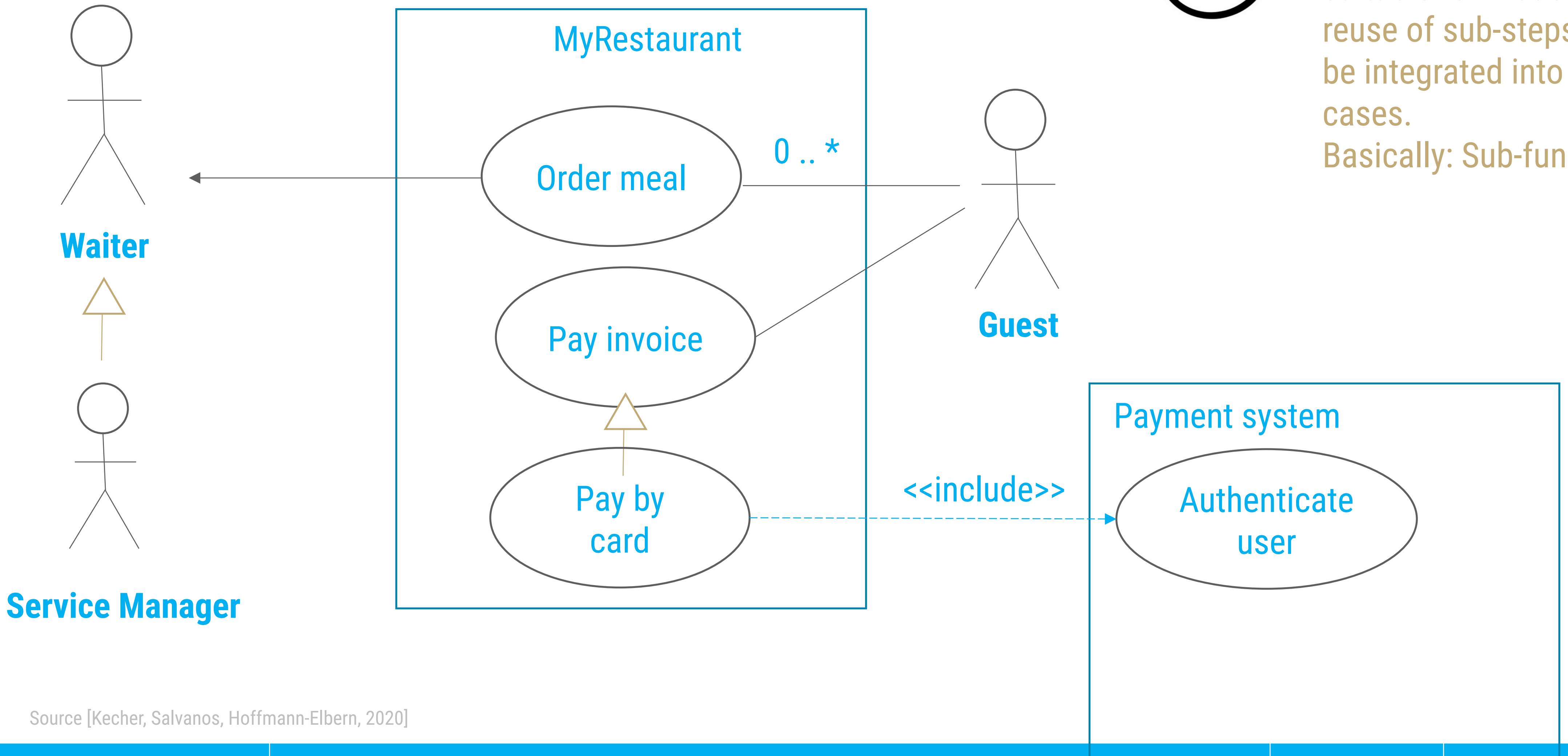


Include relationship models that a certain functionality is (and must be) included in a use case.

Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

Use cases → Notation elements

Include relationship: MUST relationship

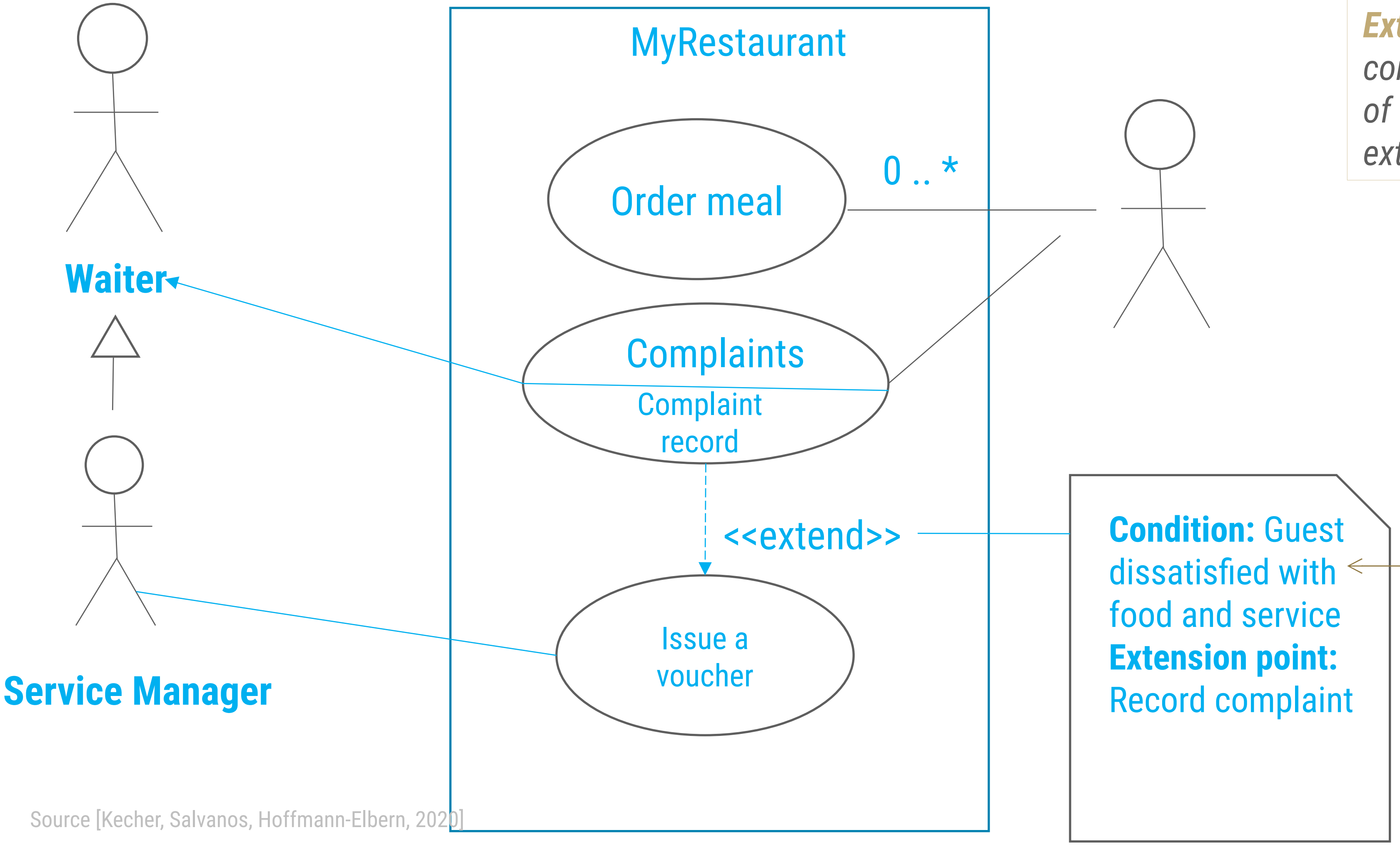


Tip: Include relationships are suitable for modeling the reuse of sub-steps that can be integrated into several use cases.
Basically: Sub-function call.

Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

Use cases → Notation elements

Extends relationship: CAN relationship



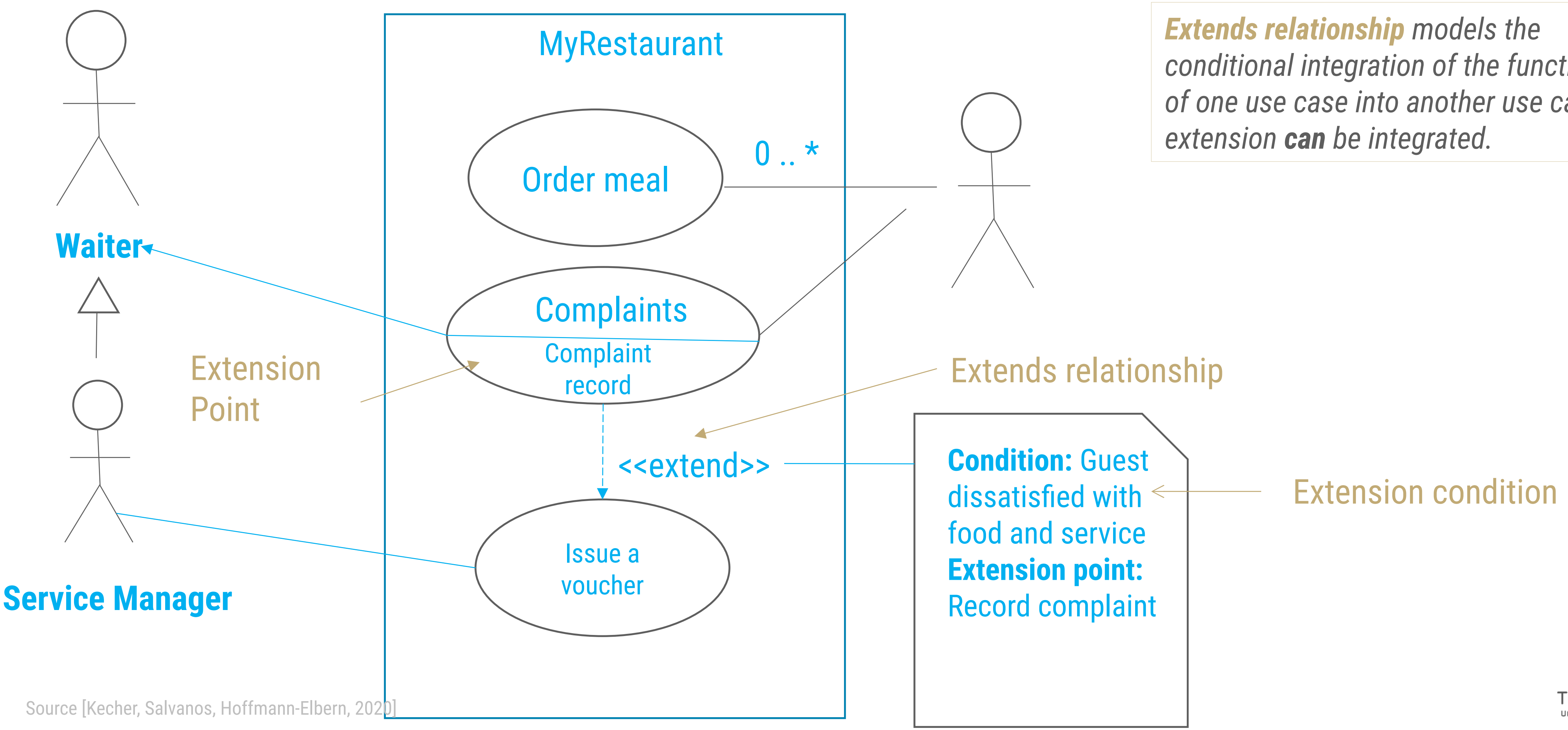
Extends relationship *models the conditional integration of the functionality of one use case into another use case. The extension* ***can*** *be integrated.*

Extension condition

Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

Use cases → Notation elements

Extends relationship: CAN relationship

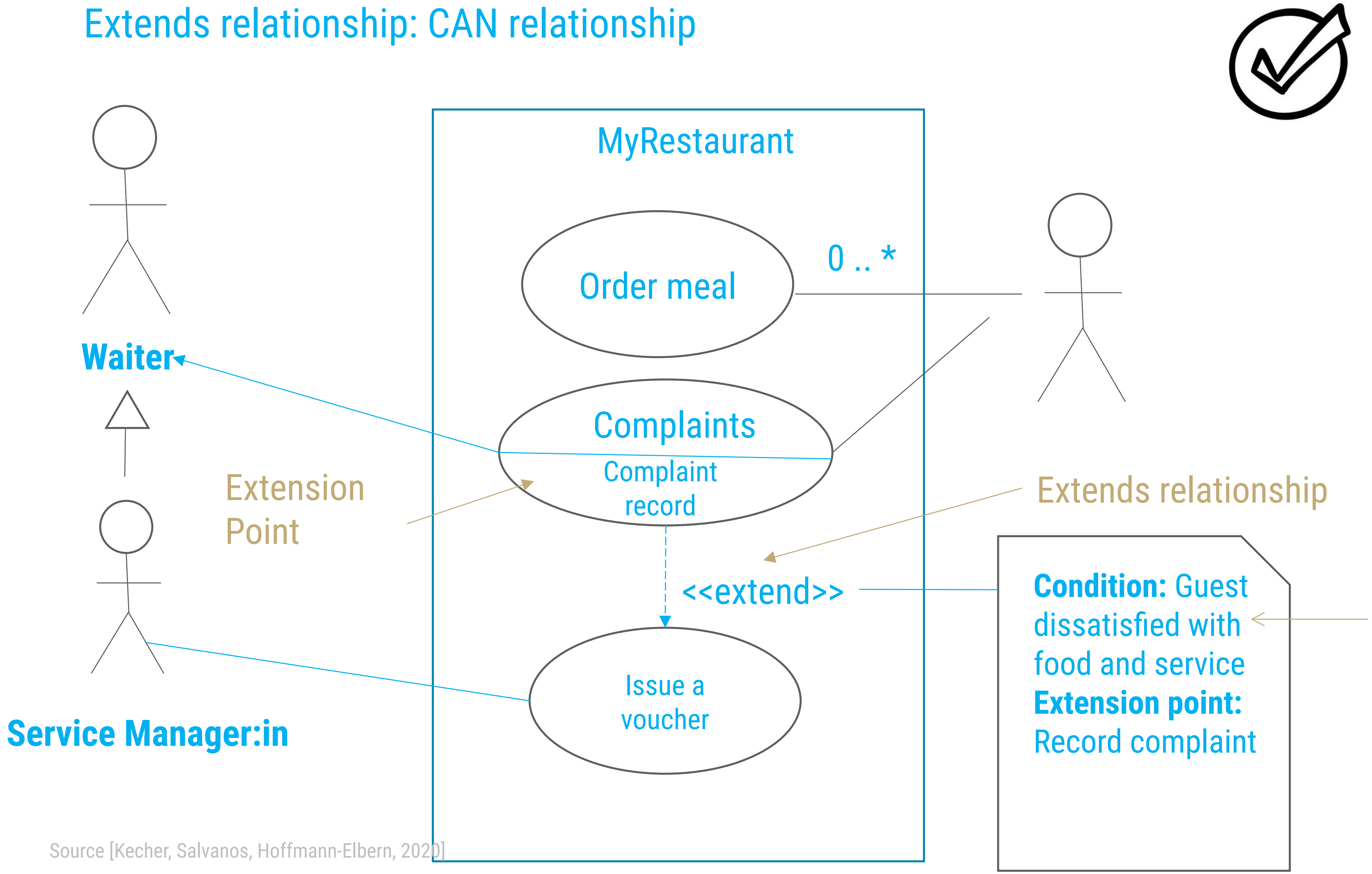


Extends relationship *models the conditional integration of the functionality of one use case into another use case. The extension* ***can*** *be integrated.*

Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

Use cases → Notation elements

Extends relationship: CAN relationship



Condition: Guest dissatisfied with food and service
Extension point: Record complaint

Tip: The Extends relationship also allows partial functionality to be outsourced.

The extending use case can typically also exist without the "caller".

Avoid cyclical dependencies.

Source [Kecher, Salvanos, Hoffmann-Elbern, 2020]

Use cases

How do you describe use cases? → Textual description

Name	Unlock door without manual intervention
Short description	This use case describes the entire procedure from identification of an authorized person to unlocking of a door
Actos	Person, outwards facing door
Pre-condition	The outwards facing door is closed and locked.
Trigger	The person wants to unlock the door / enter the house.
Main Scenario	1. The person arrives in front of the house door 2. The system detects the person as authorized 3. The system saves the result 4. The system unlocks the door 5. The authorized person opens the door 6. Use case completed
Alternative Scenarios	5a: The authorized person does not open the door. 5a1: The system locks the door after 10 seconds 5a2: Continue with step 6 2a: The person is not authorized. 2a1: The system saves the snapshot. 2a2: Continue with step 6
Post-conditions	Main Scenario: Door state is open and unlocked.

Source: [Rupp, 2021]

Main scenario,
Happy Path

Typically as a "dialog" between system and user

Use cases

→ Procedure [Ludewig, 2013]

Identify **actors** (people and systems)	The task analysis provides important information for this step. If systems interact with the system to be modeled, they are also actors.
Set **system boundaries**	The system boundaries define what belongs to the system and what does not.
Identify use cases of the **main functions**	The use cases of the identified **main functions** are roughly formulated. It must be ensured that the use cases fully cover the functionality.
Structure use cases and the actors	The relationships between actors and use cases as well as between the use cases themselves (include/extends) are identified.

Use cases

→ Procedure [Ludewig, 2013]

Activity	Notes
Outline the **normal sequence** of use cases.	The normal process, i.e. the path to the desired goal of the main actor, should be formulated first.
Describe **special cases** and **alternative processes**	It must be clarified which special cases and which alternative processes can occur and how these deviate from the normal process.
Check for and extract identical **interaction sequences**	Identical interaction sequences can be defined as **basic functions** and used in several places. With the help of the generalization relationship, sequences can be specified generically, then specialized and reused.
Validate use cases	The use cases are checked in reviews and released once all corrections and improvements have been incorporated.

Project Task

Part of **Readiness for Acceptance** is a requirements specification with the following contents:

- Product vision and product goals
- Roles and personas
- User stories
- Glossary of terms
- Quantity structure
- Use cases

Literature

- **[Ludewig 2013]** Ludewig, Lichter: Software Engineering. Grundlagen, Menschen, Prozesse, Techniken, dpunkt.verlag.
- **[Paech 2021]** Barbara Paech: Vorlesung Software Engineering, Uni-Heidelberg.
- **[Balzert, 2009]** Helmut Balzert: Lehrbuch der Softwaretechnik, Basiskonzepte und Requirements Engineering, 3. Auflage, 2009, Springer.
- **[Sommerville 2018]** Ian Sommerville: Software Engineering, Pearson.
- **[Rupp, 2021]** Rupp & die SOPHISTen: Requirements Engineering und – Management, Hanser Verlag, 7. Auflage, Hanser Verlag, 2021.

Thank you for your attention!

Software Engineering

Prof. Dr. J. v. Kistowski

TH Aschaffenburg
university of applied sciences