Technology
Arts Sciences
TH Köln

Requirements Engineering

Master Digital Science
Stefan Bente, Fabian Krampe

Reference

Pre-Version 0.1, translated by Deepl Pro from German, NOT YET POST-EDITED

This quick reference is intended to serve as a "cheat sheet" in everyday requirements engineering work. The module content of the Master course “Requirements Engineering” is summarized here in compact form.

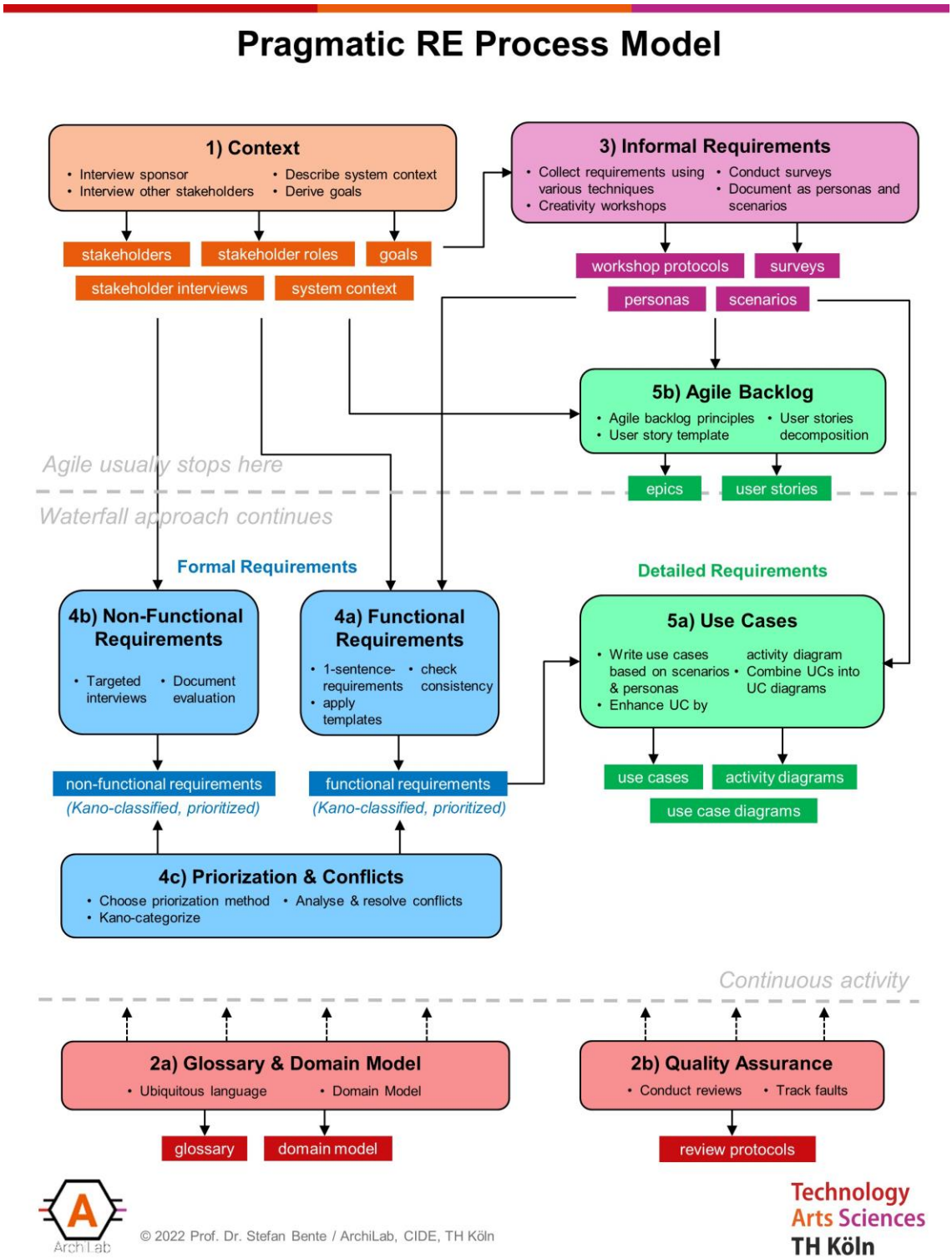
The individual chapters are structured according to a similar scheme throughout:

- Glossary / clarification of terms
- Templates
- Common procedures, techniques, and methods (with recommendations)
- Reference to literature for details that have no place in the quick reference guide

Content

0 Introduction and overview

The following pragmatic process model is the base of this course. Use it as a guideline and adapt it to the circumstances or your particular project.



0.1 Glossary of terms for basic documents

You will encounter the following documents during the requirements gathering and implementation phases of a project. With the content of this course, you will be able to make contributions to these documents.

Term	Definition	Do you need for ...
Statement of work Functional Specification *(German: **Lastenheft**)*	Created by the **client**. Contains the totality of requirements for deliveries and services of a contractor (system vision, system goals, functions and qualities, context aspects).	Summarizes all **requirements for** an IT system. With the methods in this document, you cover the essential part of the specification creation. The specifications form the basis for a **tender**.
Scope Statement Technical Specification *(German: **Pflichtenheft**)*	Specification prepared by the **contractor.** Describes the implementation of the specifications given by the client, the architecture and planned details of the realization.	Forms the contractor's response to the statement of work.

0.1.1 Proposal for structure of a statement of work

- Introduction
  - Problem description
  - System context
  - Stakeholder overview
  - System goals
- Product requirements
  - Personas
  - Scenarios
  - Functional requirements
- Use Cases
- Professional system context
  - Business processes, incidents
  - Business properties
  - Further technical framework
- Technical system context
  - HW and SW configuration
  - Interfaces to other systems
  - Other technical conditions
- Development framework
  - Development process model
  - Development tools
- Time and cost frame
  - Milestones
- Directories
  - Reference documents

0.1.2 Further reading

- Pohl, 2008, pp. 232-236 and 251-257.
- Rupp, 2014, p. 36-39
- Schienmann, 200, pp. 141-148
- Weit e.V. (2006), esp. pp. 1-61

1 Identification of goals and system context

The *system context* describes aspects of the environment that have a relevant relationship to the system. The system context includes requirements sources, such as documents, operative systems, and *stakeholders* that can be used for requirements elicitation. Stakeholder *goals* serve as an initial description of intended usage practices of the system. They can be used to verify the relevance and completeness of the requirements.

1.1 Glossary / Clarification of terms

Term	Definition	Do you need for ...
System context	Part of the environment of a system that is relevant to the definition and understanding of the requirements of the system under consideration.	Identify sources of requirements and avoid misleading requirements.
Stakeholder	Person or organization that has direct influence on the requirements of the system under consideration, and/or an interest in the system.	Important source of system context, goals, and requirements.
Goal	Intentional description of a characteristic feature of the system to be developed or the associated development process.	Creates common system understanding. Requirements serve to achieve this goal.

1.2 General approach

- Identify and document *stakeholders* (stakeholder template)
  - Classify stakeholders (influence/motivation matrix, see chap. 1.3.1)
  - Derive measures for dealing with individual stakeholders
  - Survey stakeholders (interviews)
- Survey the actual situation in a non-judgmental way, note problems, no solutions
  - Evaluate actual situation, identify causes
- Derive *goals* and assign them to the stakeholders
  - Refine and document goals (goal template, 7 rules)
- Define system context

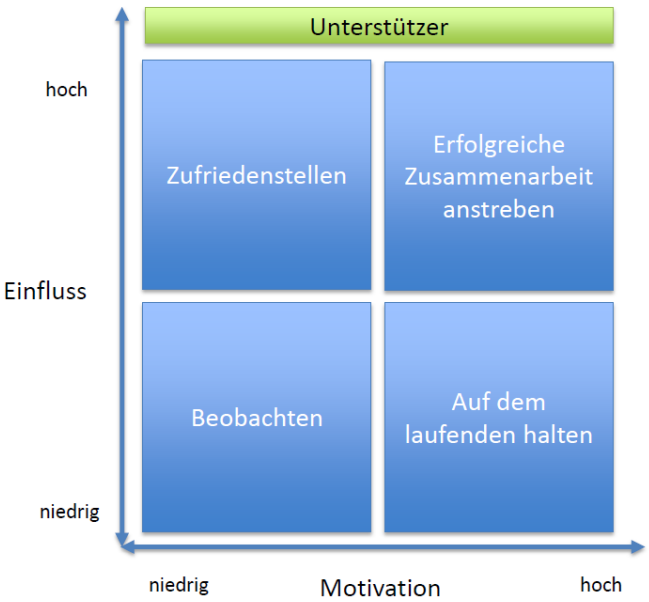
1.2.1 Challenges and pitfalls

- The **system context** usually cannot be *completely specified* at the beginning. The system boundary needs be clearly defined only at the end of the requirements elicitation process. Therefore, do not invest too much effort in the initial system context boundaries.
- **Forgotten stakeholders** result in missing requirements. Therefore, pay *special attention* to identifying stakeholders.
- **Conflicting goals** between stakeholders are often not immediately visible.

1.3 Stakeholder

1.3.1 Templates and stencils

Stakeholderschablone		
Nr.	**Abschnitt**	**Inhalt**
1	Funktion (Rolle)	
2	Name	
3	Kontakt	
4	Verfügbarkeit	
5	Wissen	
6	Interessen & Ziele	
7	Relevanz	
optional 8	Entscheidungsbefugnis	
optional 9	Beziehungen und Abgrenzung verschiedener Stakeholder	
optional 10	Grund, warum diese Person als Repräsentant für die Stakeholderrolle ausgewählt wurde	
optional 11	Grad der Beteiligung während der Analyse	
optional 12	Art der Kommunikation (per Telefon, E-Mail, Datenbank)	
optional 13	Mitwirkung während der Qualitätssicherung und Freigabe der Anforderungen	



Test questions for stakeholder identification

The following questions will help you compile your stakeholder lists.

- Who are the users of the system?
- Who is the customer of the system?
- Who will maintain the new system?
- Who else is affected by the outputs of the system?
- Do other people or organizations exist that are interested in the system?
- Who will evaluate / approve if / when the system will be delivered or deployed?

Stakeholder categorization

1.3.2 Investigation techniques / methods: stakeholder interview

Planning:

- As a basis, e.g., Kaiser (2014).
- Develop interview objective and gather rough information about the stakeholder
- Create interview guide
- Distribute roles (interviewer, protocolist,...)

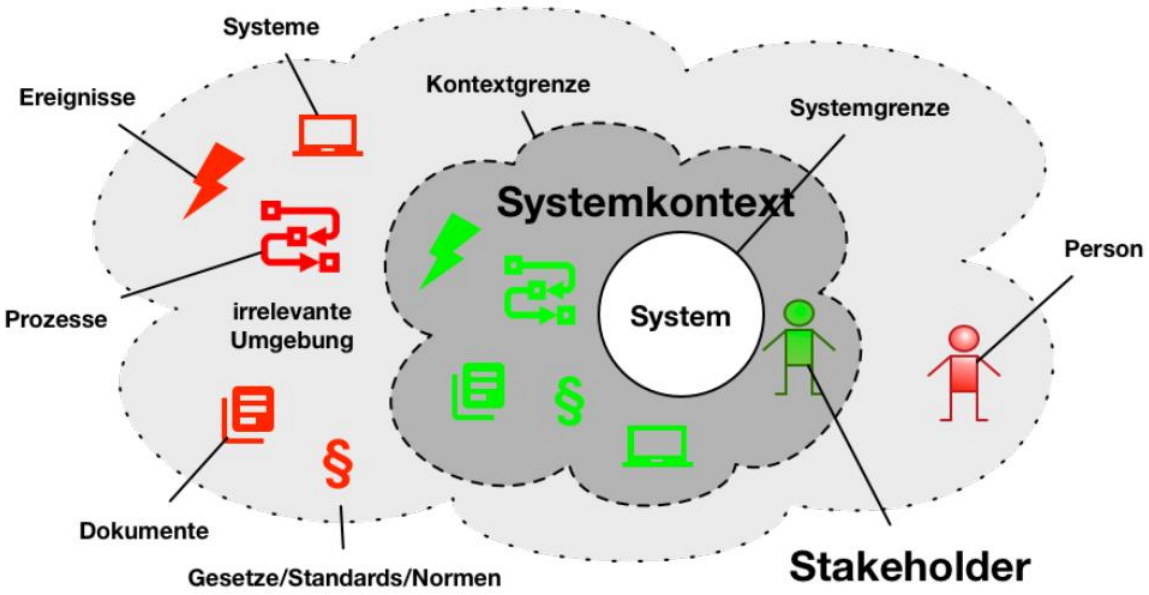
Interview Guide:

- Introduction:
  - Introduce yourself or the team
  - State the goals of the interview
  - Motivate the stakeholder to speak and respond freely.
- Main part:
  - What is your role related to the product?
  - What are your personal expectations for this project?
  - Do you have any concerns / worries related to the project? If yes, what are they?
  - What should this product or service be/provide?
  - What is (not) to be achieved with the new system?
  - How would you personally define the success of this project?
  - Which processes are executed, and how?
  - What problems occur during the execution of processes?
  - Are there any rules and regulations that need to be taken into account?
  - How should the system be used?
  - What tools / methods are used to perform specific tasks?
  - Is there any technology that should be applied / incorporated?
- Conclusion:
  - Ask about other possible stakeholders
  - Clarify further collaboration (e.g., "How would you like to be involved in the project and what is the best way to reach you?")

Follow-up:

- Provide the stakeholder with the transcript of the interview
- Get confirmation to resolve any misunderstandings (review).

1.4 System context



1.5 Targets

1.5.1 Seven rules for target definition (after Pohl, 2008)

1. Formulate goals **briefly and concisely**, no filler words, no nested sentences
2. Use **active formulations, as the** actor must be named **here.**
   - *Not: The* duration of the reporting process is to be halved compared to the current situation.
   - *But: Clerks* can create reports in half the time as before.
3. Formulate **verifiable** goals.
4. If your goal is not verifiable, break it down into verifiable **subgoals**.
5. Formulate the **added value of** the goal (what benefit will it achieve?).
6. Provide a **rationale for** the goal.
7. Avoid **solution approaches**.

1.6 Further reading

- Broadbent & Kitzis, 2004, pp. 51-55.
- Kaiser, Robert, 2014, Qualitative Expert Interviews: Conceptual foundations and practical implementation. Springer-Verlag[1] .
- Leffingwell, 2003, pp. 43-57.
- Pohl, 2008, pp. 89-108
- Pohl, 2011, pp. 21-32
- Rupp, 2014, pp. 62-77

[1] Available as an electronic resource in the university library

2 Survey techniques, personas, scenarios

2.1 Survey methods

2.1.1 Glossary / Clarification of terms / Advantages and disadvantages

Techniques	Definition	Do you need for ...	Advantages	Disadvantages
Interviewing techniques	Determining the wishes and needs of the stakeholders by asking specific questions	Initial survey of "obvious" requirements, recognition of **performance factors** (see Kano model in chap. 6.2.2, S. 29)	• Targeted questioning possible • Requirements determination can be structured well, results can be compared well	• Works only with stakeholders who are aware of the requirements and can verbalize thoughts • Quality of answers strongly depend on the quality of the questions
Document-centered / artifact-based techniques	Requirements analysis based on an existing system, documents or other artifacts	Identify or supplement requirements when stakeholders are not available to be interviewed	• Overview of functionalities of the old system => completeness of the new system • Reuse solutions and experience => save time and reduce costs	• Not suitable with many potential changes • Good documentation / structure of the legacy system is a prerequisite • Under certain circumstances, errors are also transferred
Creativity techniques	Techniques for using imagination and creativity (in determining requirements).	Breaking up rigid thought patterns, recognizing **enthusiasm factors** (see Kano model)	• New, innovative ideas • Helps break down thought patterns • Sometimes gives unusual ideas	• Leads to undetailed / unstructured requirements • Stakeholder collaboration / acceptance central to success • Does not fit into every corporate culture
Observation techniques	Identify and analyze requirements by observing / documenting the steps of stakeholders and potential users.	Documentation of requirements that often fall by the wayside during written elaboration because they are taken for granted (**basic factors**, see Kano)	• Recognition of inefficient processes through external view • Suitable if stakeholders cannot express their know-how linguistically	• Stakeholders might feel monitored and therefore distort the process • Not applicable for new developments

2.1.2 Interviewing techniques

- **Interview**: Guided interview based on a question guide. Useful for questioning individual experts specifically and in depth. It is essential to secure the results.
  - Prepare list of questions, but allow flexibility. See guidance for stakeholder interviews (chap. 1.3.2 on p. 2).
  - Respect the opinion or knowledge of the interview partner!
- **(Online) survey** Through a survey, many potential users can be reached with little effort. In addition, the respondents do not feel pressured by a real person, as is the case with an interview.
  - Careful preparation with test runs necessary.
  - Important: Free text fields for naming aspects not previously considered.
- **Self-writing:** Selected stakeholders document their fields of activity as well as their requirements, suggestions for change and optimization.
- **On-Site Customer:** A stakeholder is available to the development team in an advisory capacity during the concept and implementation phases.

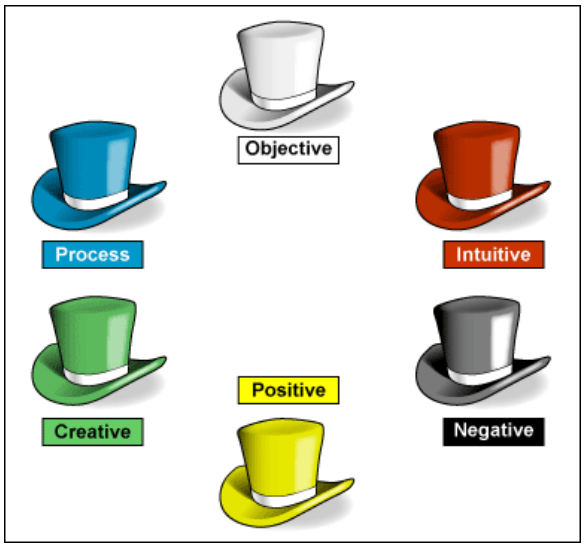
2.1.3 Document-centric / artifact-based techniques.

- **System archaeology:** Functional requirements are derived from legacy systems, for example by analyzing user documentation, analyzing user interfaces, analyzing source code etc.
- **Perspective-based reading**: Existing specification documents are specifically evaluated from the perspective of a stakeholder / potential user to extract functional requirements.

2.1.4 Creativity techniques

- **Brainstorming**: In a group of 5-10 people, ideas are collected in a given time. Each idea is written down or posted for all to see, without comment from the creator or the other participants. The method works very well with a wide variety of stakeholders. A wide range of ideas is represented. However, care should be taken to ensure that the mood is positive and that there is no early criticism.
- **Brainstorming Paradox**: This involves collecting ideas that are **not** intended to be **achieved.** This is especially useful when the group is at a "knot" and there is a mood of lack of ideas, or when there is strong dissent in the group. The deliberately negative perspective makes it easier to focus on the desired solutions.
- **6-3-5 Method**: 6 people create 3 ideas each at the beginning and write them on a piece of paper. The slips of paper are then passed on to the next person (clockwise). They now write down an extension or a supplementary, new idea for each idea. The whole thing is repeated 5 times, so that each participant has considered each idea once. Each round has a given time frame. This method works very well with difficult group dynamics because the discussion is in writing. The method produces fewer ideas than brainstorming, but they are usually more accurate. (Depending on the size of the group, adjustments are possible, e.g. 4-2-3).

- **Walt Disney method**: The group moves one after the other into three different rooms, each representing competing views: 1) Visionary, 2) Realist, 3) Critic. There, all participants* should adopt the "spirit" of the room. The respective rooms should transport the view through their furnishings (e.g. realist = sober furnishings).
- **6-hats method:** With this method, different views of the problem / project are collected. Each stakeholder is assigned a view and must adopt it. These views cover a range of opinions from "critical" to "analytical" to "positive". The stakeholders have to get involved with this method, as they have to put their own opinions in the background.
    - **blue**: organizing, moderating thinking, overview, processes, Big Picture ("the blue sky")
    - **white**: analytical thinking, focus on facts and achievable ("the white sheet")
    - **red**: emotional thinking, feeling, concentration on feelings and opinions ("fire and warmth")
    - **black**: critical thinking, risk assessment, problems, skepticism, criticism and fears ("doom and gloom", Advocatus Diaboli)
    - **yellow**: unconditionally optimistic thinking, best-case scenario ("sunshine")
    - **green**: creative, associative thinking, new ideas ("green meadow")



2.1.5 Observation techniques

- **Field observation**: The requirements investigators play "fly on the wall" and observe / log the existing workflows of the potential users. By observing the users, their behavior in the respective context can be analyzed and understood particularly well. In addition, the users are usually highly available, as they can pursue their normal tasks. Furthermore, the observers may notice aspects that the users had not even thought of because they were obvious to them.
    - How do users react under time pressure?
    - What influence do external aspects such as light, noise, other users have?
    - How do users work when they are distracted?
- **Apprenticing**: The requirements investigator is "trained" over a limited period of time for the operational activities of the potential users (apprentice). Outside the apprenticing process, the procedures are then documented and analyzed.

2.1.6 Further reading

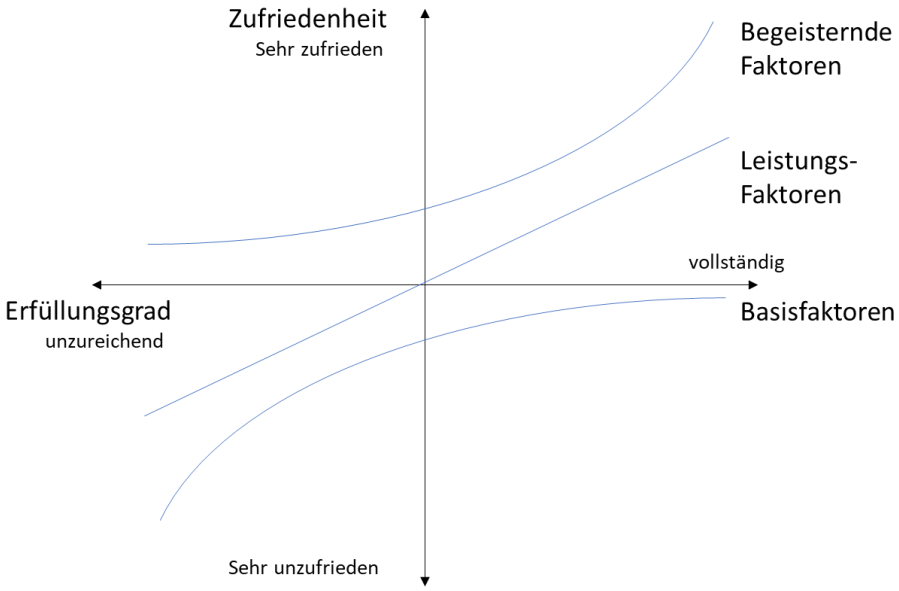
- Bono, E. de. (1989).
- Gürtler, Meyer, 2013
- Pohl, 2008, pp. 32-41

- Rupp, 2014, pp. 80-106
- Schienmann, 2001, pp. 203-213

2.2 Classification according to Kano method

The Kano model helps to classify requirements. The criterion used here is the satisfaction of stakeholders and potential users, or more precisely: how they would perceive and evaluate the respective property.

For this purpose, the requirements are divided into three characteristic categories (**basic**, **performance** and **enthusiasm factors**). This classification can then serve as the basis or input for a requirements prioritization.



The classification according to Kano factors becomes important later when functional requirements ("1-set requirements") are established (see chapter 4). You can then group them according to Kano and use them later for prioritization (see also Chapter XXX).

2.2.1 Glossary / Clarification of terms

Term	Definition	Do you need for ...
Base factor	System property that is taken for granted	Must be met to guarantee satisfaction. However, it is often not explicitly mentioned by stakeholders because they take it for granted. Therefore a potential source of misunderstanding and conflict during implementation.
Power factor	Consciously and explicitly required feature of the system	Performance factors usually form the main part of a requirement specification.
Enthusiasm factor	System property that is not expected, but is perceived as pleasant and useful when present.	Increases stakeholder acceptance of the system by providing unexpected features that are perceived as positive.

2.2.2 Which investigation techniques (see chap. 2.1 on p. 10) provide which Kano factors?

Base factors	Performance factors	Enthusiasm factors
Observation techniques • Field observation • Apprenticing **Document-centered / artifact-based techniques** • System Archaeology • Perspective-based learning	**Interviewing techniques** • Self-recording on the part of the stakeholder • On-Site Customer • Interviews • Workshops *(but also all other techniques)*	**Creativity techniques** • 6 hats method • 6-3-5 Method • Walt Disney method • ...

2.2.3 Help for Kano classification

Although the definition of the Kano factors is intuitively understandable, the assignment of a concrete requirement to a factor often causes problems. Here it helps to ask the stakeholder (or oneself, as a thought experiment) two questions and to classify the answers according to the following matrix.

Dysfunctional question: What if the feature is missing?

Functional question: What if the feature is present ?

Funktional \ Dysfunktional	Das würde mich sehr freuen	Das setze ich voraus	Das ist mir egal	Das nehme ich gerade noch hin	Das würde mich sehr stören
Das würde mich sehr freuen	-	-	Begeisterung	Begeisterung / Leistung	Leistung
Das setze ich voraus	-	-	-	-	Basis
Das ist mir egal	-	-	Unerheblich	-	-
Das nehme ich gerade noch hin	Rückweisung	-	-	-	-
Das würde mich sehr stören	Rückweisung	Rückweisung	(Rückweisung)	-	-

2.2.4 Further reading

- Gürtler, Meyer; 2013
- Pohl 2008, pp. 32-41
- Rupp 2014, pp. 80-106
- Schienmann 2001, pp. 203-2013

2.3 Personas

Personas are fictitious people who represent the **typical users** in a target group. The persona should represent the **important characteristics of** the target group. Personas are created in narrative form and are based on real user stakeholders.

2.3.1 Glossary / Clarification of terms

Term	Definition	Do you need for ...
Persona	Text-based archetypes of real users, represents characteristics of a stakeholder.	Identify possible functionalities for the system to be created. Putting yourself in the shoes of the user.
primary Persona	Persona in the focus of system design.	Prioritization by identifying the "important" stakeholders that feed into the persona. Opposite are *secondary* personas.

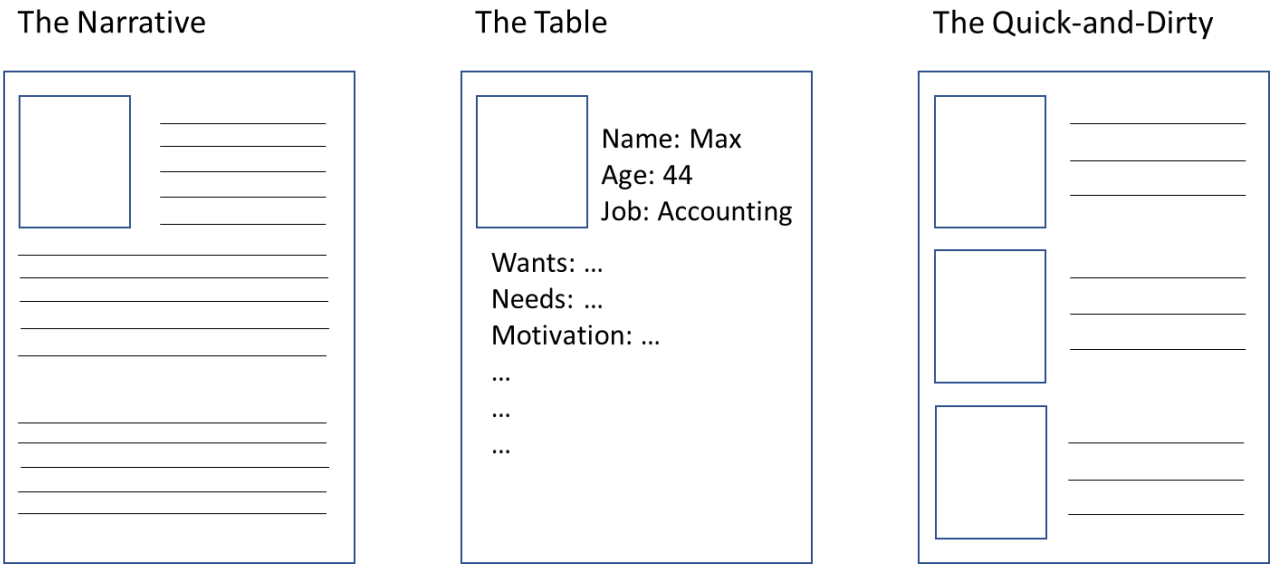
2.3.2 Templates: Description of personas

Checklist:

- Photo
- Name, age
- Education
- Job
- Family
- Disabilities
- Technology
- Motivation, goals, needs
- Expectations
- Behavior
- Working environment
- Skills

Max. 1 page, relevant properties vary depending on context.

With personas, there is a danger that creators are too strongly guided by their own ideas. It is therefore important that the designer or developer recognizes that they are not developing for themselves.



2.3.3 How do I create personas?

Depending on the context and the task at hand, various options are available. In some cases, a clever combination is the most profitable option. For example, in the case of demographic questions, market research in combination with an online survey is the best option in most cases.

Suitable techniques are interviewing and observation techniques (see also the description of survey techniques in chap. 2.1 on p. 10):

- Interview
  - Interviews are suitable to better understand requirements, behavior and needs of users and influences of the environment
  - Usually last 30-60 min
  - Hold interview, if possible, in person's work environment to gain insight into user's activities through observation
  - Respect the opinion or knowledge of the interview partner!
  - W-questions: Who, Where, What, When, How, Why, From where?
- Survey
- Field observation of users (in the real context of use).
- Market research
  - Through a market research, many parameters can be determined in advance with a low effort. Through a combination with another determination technique, advantages can be drawn from both.
  - A pure market research for the creation only makes sense if the personas are enriched in time by well-founded project-related data.

2.3.4 Further reading

Calabria, 2004
Cooper 1999, pp. 123-148
Pohl 2008, pp. 127-138
Rupp 2014, p. 210-211

2.4 Scenarios

Scenarios describe situations in which people interact with a system (story line). These situations have a clearly defined context. On the one hand, scenarios serve to informally record requirements for a system and, on the other hand, to evaluate a system with respect to formalized goals and requirements.

Scenarios are based on the previously created personas (see chap. 0) and can refer to the current situation (ACTUAL scenario) or to the future (improved) situation (TARGET scenario). Scenarios focus on the activities of people and less on concrete interactions. In addition, these include the usage environment of the application or system, the duration of use, the probability of interruption, and the use of other applications.

2.4.1 Glossary / Clarification of terms

Term	Definition	Do you need for ...
Scenario	Narrative, informal description of a situation in which personas interact in the system in the particular context of use.	Recording of typical usage sequences (e.g. as a result of survey techniques from chap. 2.1), without initially restricting oneself by too many formalisms
Context of use	Action space and environment of the user (tasks, physical and social environment, hardware and software, consumables, etc.)	Reduction to the most important details of the interaction between user and system

2.4.2 Further reading

- Calabria, 2004
- Cooper 1999, pp. 123-148
- Pohl 2008, pp. 127-138
- Rupp 2014, p. 210-211

2.4.3 Scenario types

Templates are not very useful for scenarios, because they do not have a formal framework (certain attributes). However, one can distinguish the different scenarios according to their type, as shown below. Not all scenario types are equally important (are used equally often).

Categorization	Type	Brief description	**Usage frequency** for SW requirements	is used for	Example
Normal vs. Exception	Main scenario (standard scenario)	Normal way to fulfill the goal	always	Standard for use cases**,** rather as type than instance scenario. Can be both **interaction** (more common case) and **system scenario**.	*See example for chap. 7 (Use Cases) on p. 35*
	Alternative scenario	Alternative way to fulfill the goal	mostly	Standard for use cases	
	Exceptional scenario	Target is not met	mostly	Standard for use cases	
Positive vs. Negative	positive scenario	Sequence of interactions leading to the fulfillment of a goal	always	Very similar to the **main scenario**	*See example for chap. 7 (Use Cases) on p. 35*
	negative scenario	Interaction sequence that leads to the non-fulfillment of a goal	mostly	Game type of the **exceptional scenario**	
	Abuse scenario	Describes an unwanted use of the system	sometimes	Game type of the **exceptional scenario**	
Concrete vs. Abstract	Instance scenario	Concrete interactions with concrete inputs and outputs between concrete persons and/or systems	always	Very concrete game type of a scenario. Can be used well together with detailed persona, i.e. **early in** the requirements identification process.	Karl wants to drive to Potsdamer Platz 1 in Berlin. Karl uses the navigation system of his VW Golf with the license plate "E-IS-12". Karl selects "Enter destination" in the main menu, enters "Potsdamer Platz 1 in Berlin" as destination and presses the button "Determine route"... (Pohl 2008)
	Type scenario	Interactions between types of actors through types of inputs and outputs	always	Is rather the abstraction level that is used in use cases (i.e. somewhat **later in** the requirements elicitation process).	The driver enters the destination address into his navigation system. The system gives the feedback that the route calculation has been made.

Categorization	Type	Brief description	**Usage frequency** for SW requirements	is used for	Example
Interaction vs. system	System internal scenarios (Type A)	Internal system interactions in the form of interaction sequences between system components	mostly	Is always useful when a very technical system is to be described. Corresponds to a **white-box view of** the system.	The "Navigation Control" component requests the GPS coordinates from the "Determine GPS Location" component. The "Determine GPS location" component transmits the coordinates. The "Navigation Control" component calls the "Screen Output" component with the current position as well as the destination. The "Screen Input" component passes the route parameters to the "Navigation Control" component, which uses them to determine the final route. (Pohl 2008).
	Interaction scenarios (type B)	Interactions between system and system users (stakeholders and systems in context).	always	**Black-box view of the** system being described. The "standard case" of a scenario.	
	Context scenarios (type C)	Extension of type B with additional interactions and information in context	mostly	If context information is available, then useful, otherwise rather type B.	The driver wants to reach a destination that is outside the map material stored in the navigation system (system). Since the navigation system cannot guide the driver without the map material, the driver decides to have a route calculation performed by his mobile operator. [...] The driver enters the start and destination location as well as the route parameters (fastest route) in the user dialog on the cell phone [...]. (Pohl 2008)

Categorization	Type	Brief description	**Usage frequency** for SW requirements	is used for	Example
Descriptive vs. Explanatory	explanatory scenario	Justification and explanation of interactions	sometimes	Mixes **benefits** and **goals** into the scenario. Explanations can also be usefully mixed into use cases if the explanation portion does not get out of hand.	The distance between the vehicles continues to decrease. Since the vehicle is traveling across the highway at high speed (> 90 km/h!) and therefore the speed should be reduced before a possible evasive maneuver, the on-board computer initiates automatic emergency braking to avoid a rear-end collision. (Pohl 2008)
	descriptive scenario	descriptive representation of the interactions	always	The standard case of scenarios	
	exploratory scenario	Alternative solution and operation options	sometimes	Has a strong "brainstorming character", fits more into the early phase of requirements identification	Karl wants to drive his car to a destination using a navigation system. The first question is whether the starting point of the journey is always the current position of the vehicle, or whether Karl selects the starting point himself. Automatic selection of the starting point avoids additional user interaction and thus supports fast navigation. The input of the starting point would allow to calculate routes that do not have the current position of the vehicle as starting point. [...] (Pohl 2008)

3 Business glossary / Business data model

The goal is to form a common vocabulary between all project participants and to create a uniform data model that represents the application domain to be mapped. This facilitates communication with the stakeholders and prevents misunderstandings due to missing definitions.

3.1 Glossary / Clarification of terms

Term	Definition	Do you need for ...
Technical glossary	Common vocabulary defined between participants of a project.	the avoidance of conflicts that may arise from conflicting interpretations.
Business object	An element that provides professional value to a project.	the identification of the basic elements that exist in an application domain
Business data model *(also: domain model)*	Describes an implementation-independent model which reflects objects of the real project context (usually in the form of a UML class diagram)	interdisciplinary communication between all project participants

3.2 Procedure

The technical glossary is being created and expanded through discussions with stakeholders (see Chap. 1.3 on p. 6), domain experts, and research in other sources such as internal documentation and technical literature.

Caution. A common mistake is to make assumptions without consulting the stakeholders, assuming sufficient knowledge of the domain.

Rule of thumb: If you **don't know** the domain, you **underestimate** its complexity.

3.2.1 Method: noun analysis for filling the glossary from technical texts

1. mark all nouns in the requirements.
2. Clean up the created raw list
   a. Delete repetitions.
   b. Remove synonyms (make sure they are really synonyms!)
   c. Delete noun verbs (be careful with words like "order" - that's a business object!)
   d. Delete nouns that are only synonyms for conjunctions or verbs.
   e. Delete all sentences that do not describe business objects.
   f. The system to be modeled does not form a business object.
   g. Delete information on subsequent implementation at the technical level.
   h. Delete labels for e.g. optionality.
3. Each marked noun forms a business object.
4. Document the identified business objects in the form of a glossary.

- Creating the domain-oriented data model from a glossary

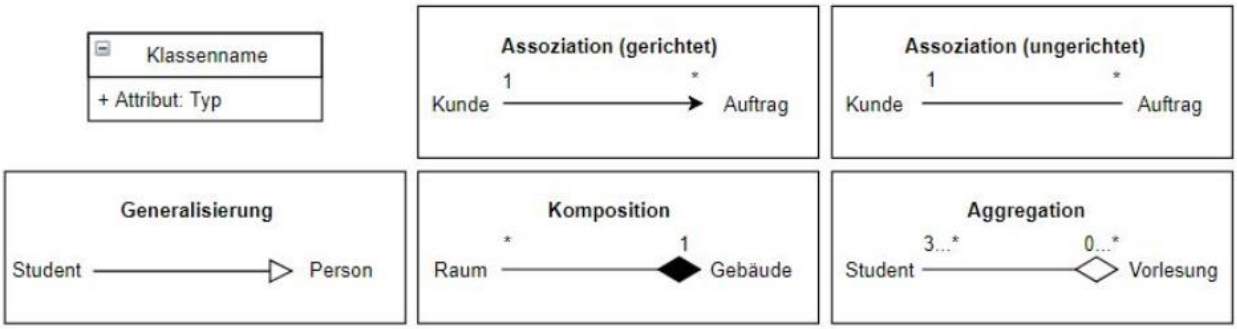
3.3 Business data model / domain model

Based on the determined business objects, a domain model can be created for the domain.

In addition to the glossary, the domain-oriented data model shows how a term / business object is **related to other** terms. The implementation as a simple UML class diagram is easy to understand even for IT laymen!

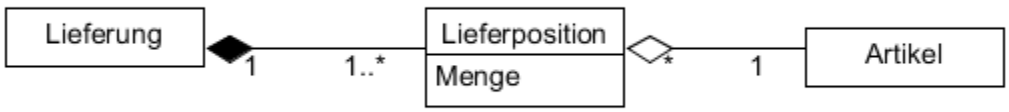
3.3.1 Method: Business data model as simple UML class diagram

Rule: Only the following relationships, no methods in the classes, little to no attributes.



3.3.2 Example

Term	Definition
Delivery	Delivery of a supplier to be placed in storage, consisting of individual delivery items
Article	Description of a delivered object
Delivery item	Component of a delivery, consisting of item and quantity



3.4 Further reading

- Ebert, 2014, p.204-206
- Evans, 2003, p25-55
- Pohl and Rupp, 2011, pp. 85-87; 95-99.
- Rupp 2014, pp. 207-208; 222-224

4 Functional requirements

Functional requirements describe the desired functionality of a system, its behavior and its data. These can often be interpreted differently. To counteract this problem, potential linguistic ambiguities should be eliminated and requirements should be formulated unambiguously using sentence templates.

4.1 Glossary / Clarification of terms

Term	Definition	Do you need for ...
Functional requirement	Describes desired functionality of a system, its data and / or behavior	Condenses the requirements for an IT system into a set of compact (1 set) requirements that can be easily prioritized and transferred into a project plan.
Sentence templates	Blueprint for the syntactic structure of a single request	Helps to formulate functional requirements without ambiguities, ambiguities and contradictions.

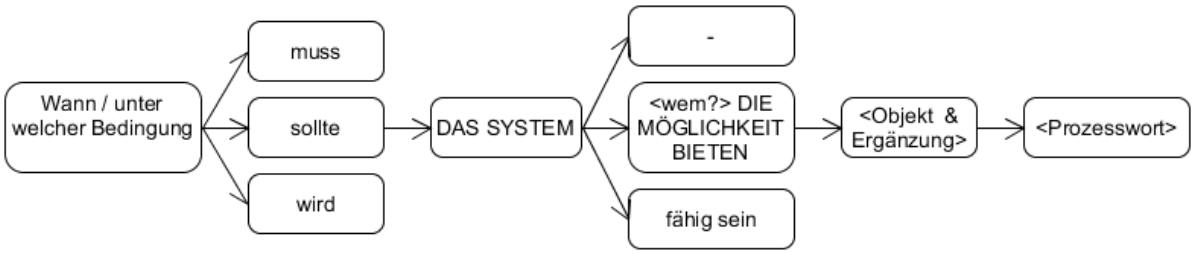
4.2 Checking and cleaning up natural language requirements

First, all potential ambiguities should be eliminated. Then, by means of a sentence template, the requirement is formalized according to a uniform scheme.

Testing	What is it about?	Typically	Example of possible problems
1) Nominalization	A (often long lasting) process is turned into a (one-time) event. As a result, an operation becomes an event, and much information relevant to the operation is lost.	• the integration • the test • the decrease • the transfer • the display • the user guidance • The confirmation	"The system allows **logging of** requests. • who logs? • when is logging done? • what is logged? • to what end? • in compliance with which rules?
2) Universal quantifiers	Universal quantifiers = information about frequencies. • combine a set of objects into a group • make statement about their behavior	• never • always • no • any • all • nothing • implicit all-quantors	"The user receives a statistical analysis of all request data." • Any user (implicit universal quantifier)? • All request data? • Always (implicitly!)?
3) Frequently used generic nouns	Commonly used nouns whose meaning is "overloaded". Further information is required to specify them unambiguously.	• the user • the system • the message • the data • the function	"For form queries, field conventions must be ensured by plausibilities in the form." • Which form? • Which field conventions? • What plausibilities?
4) Incompletely specified process words	Some process words (verbs) require more information to be fully specified.	Anything that doesn't hold up after W questions: Who? What? What for? Why? When? Where? How? How much?	"The user enters the login data" • **Where** and **when is** the input?

Testing	What is it about?	Typically	Example of possible problems
5) Incompletely specified conditions	The requirement contains a condition that is not fully specified.	Signal words: • if ... then • if • in the case of • depending on	"If the user is shown a lock for the selected record ..." • What if no lock? • What if all records are locked?

4.3 Record template for functional requirements



- "must": absolutely belongs to this release
- "should": optional
- "will": "must" for a future release, not now

For this classification, the Kano model can be taken as an aid (see chapter 2.2). The following applies:

- Basic factors are "must" requirements
- Performance factors can be "must", "should", or "will"
- Enthusiasm factors as well, but for tactical reasons some enthusiasm factors should be represented at "must"

4.4 Example

Example: from the following output set

> *The reasons for the request are to be stored when logging the requests.*

is done by applying the cleanup rules and using the record template:

> *In the first processing step | the system | must be capable of |*
> *log each incoming request with the complete data of the request as well as the timestamp of receipt |.*

4.5 Further reading

Pohl 2008, pp. 239-249
Rupp 2014, pp. 159-181

5 Non-functional requirements

Non-functional requirements (NFA) include all requirements that cannot be assigned to functional requirements.

They describe important quality characteristics and boundary conditions of the system to be developed.

5.1 Glossary / Clarification of terms

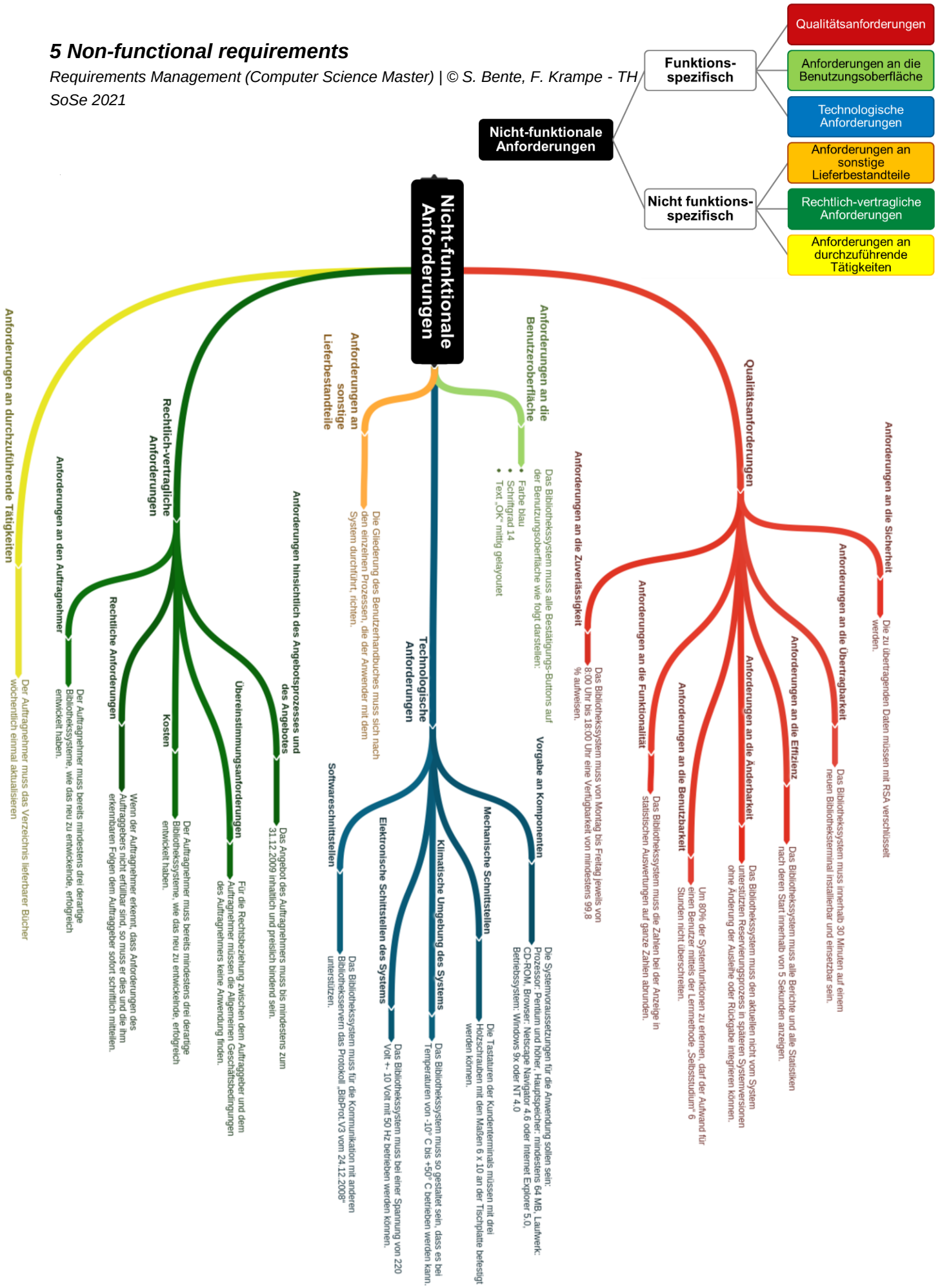
Term	Definition	Do you need for ...
Non-function-specific requirements	Concern the "how" and not the "what" of realization	Operational design of the service provision
Quality requirements	Determine in which "quality" the functional requirements are to be fulfilled	Especially for the description of load behavior and reliability / availability: **Performance** • Response time (min, max, avg) for a transaction • Throughput • Capacity (quantity structure) **Reliability** • Availability (%) • Mean time between failure / Mean time to repair / **Modifiability** **Security** **Interoperability** / Conformity
Technological requirements	Provide solution specifications, describe environment in which the system is to be operated	Ensure the conformity of the solution to be created to the operating environment and the strategic objective of the organization / company.

5.1 Further reading

- Leffingwell, 2003, pp. 257-269.
- Rupp, 2014, pp. 247-284
- Schienmann, 2001, pp. 132-137

5.2 Classification and examples

The figure overleaf shows the different types of non-functional requirements, each with a short example.


Nicht-funktionale Anforderungen
Funktions-spezifisch
Qualitätsanforderungen
Anforderungen an die Benutzungsoberfläche
Technologische Anforderungen
Nicht funktions-spezifisch
Anforderungen an sonstige Lieferbestandteile
Rechtlich-vertragliche Anforderungen
Anforderungen an durchzuführende Tätigkeiten
Nicht-funktionale Anforderungen
Qualitätsanforderungen
Anforderungen an die Sicherheit
Die zu übertragenden Daten müssen mit RSA verschlüsselt werden.
Anforderungen an die Übertragbarkeit
Das Bibliothekssystem muss innerhalb 30 Minuten auf einem neuen Bibliotheksterminal installierbar und einsetzbar sein.
Anforderungen an die Effizienz
Das Bibliothekssystem muss alle Berichte und alle Statistiken nach deren Start innerhalb von 5 Sekunden anzeigen.
Anforderungen an die Änderbarkeit
Das Bibliothekssystem muss den aktuellen nicht vom System unterstützten Reservierungsprozess in späteren Systemversionen ohne Änderung der Ausleihe oder Rückgabe integrieren können.
Anforderungen an die Benutzbarkeit
Um 80% der Systemfunktionen zu erlernen, darf der Aufwand für einen Benutzer mittels der Lernmethode „Selbststudium" 6 Stunden nicht überschreiten.
Anforderungen an die Funktionalität
Das Bibliothekssystem muss die Zahlen bei der Anzeige in statistischen Auswertungen auf ganze Zahlen abrunden.
Anforderungen an die Zuverlässigkeit
Das Bibliothekssystem muss von Montag bis Freitag jeweils von 8:00 Uhr bis 18:00 Uhr eine Verfügbarkeit von mindestens 99,8 % aufweisen.
Technologische Anforderungen
Vorgabe an Komponenten
Die Systemvoraussetzungen für die Anwendung sollen sein: Prozessor: Pentium und höher, Hauptspeicher: mindestens 64 MB, Laufwerk: CD-ROM, Browser: Netscape Navigator 4.6 oder Internet Explorer 5.0, Betriebssystem: Windows 9x oder NT 4.0
Mechanische Schnittstellen
Die Tastaturen der Kundenterminals müssen mit drei Holzschrauben mit den Maßen 6 x 10 an der Tischplatte befestigt werden können.
Klimatische Umgebung des Systems
Das Bibliothekssystem muss so gestaltet sein, dass es bei Temperaturen von -10° C bis +50° C betrieben werden kann.
Elektronische Schnittstellen des Systems
Das Bibliothekssystem muss bei einer Spannung von 220 Volt +- 10 Volt mit 50 Hz betrieben werden können.
Softwareschnittstellen
Das Bibliothekssystem muss für die Kommunikation mit anderen Bibliotheksservern das Protokoll „BibProt V3 vom 24.12.2008" unterstützen.
Anforderungen an die Benutzeroberfläche
Das Bibliothekssystem muss alle Bestätigungs-Buttons auf der Benutzungsoberfläche wie folgt darstellen:
Farbe blau
Schriftgrad 14
Text „OK" mittig gelayoutet
Anforderungen an sonstige Lieferbestandteile
Die Gliederung des Benutzerhandbuches muss sich nach den einzelnen Prozessen, die der Anwender mit dem System durchführt, richten.
Rechtlich-vertragliche Anforderungen
Anforderungen hinsichtlich des Angebotsprozesses und des Angebotes
Das Angebot des Auftragnehmers muss bis mindestens zum 31.12.2009 inhaltlich und preislich bindend sein.
Übereinstimmungsanforderungen
Für die Rechtsbeziehung zwischen dem Auftraggeber und dem Auftragnehmer müssen die Allgemeinen Geschäftsbedingungen des Auftragnehmers keine Anwendung finden.
Kosten
Der Auftragnehmer muss bereits mindestens drei derartige Bibliothekssysteme, wie das neu zu entwickelnde, erfolgreich entwickelt haben.
Rechtliche Anforderungen
Wenn der Auftragnehmer erkennt, dass Anforderungen des Auftraggebers nicht erfüllbar sind, so muss er dies und die ihm erkennbaren Folgen dem Auftraggeber sofort schriftlich mitteilen.
Anforderungen an den Auftragnehmer
Der Auftragnehmer muss bereits mindestens drei derartige Bibliothekssysteme, wie das neu zu entwickelnde, erfolgreich entwickelt haben.
Anforderungen an durchzuführende Tätigkeiten
Der Auftragnehmer muss das Verzeichnis lieferbarer Bücher wöchentlich einmal aktualisieren

6 Prioritization and conflict resolution

Prioritization is necessary to separate requirements from each other (important and unimportant), as well as to bring order to a project and ensure project success. These can be realized by ad-hoc, analytical and agile techniques, depending on the project.

Possible prioritization criteria include:

- Business or strategic value
- Implementation costs
- Interrelation / dependence between requirements
- Risks / costs in the event of development failure

6.1 Glossary / Clarification of terms

Term	Definition	Do you need for ...
Ad hoc prioritization	Simple and fast prioritization	Large quantities of requirements, pre-prioritization
Analytical prioritization	Prioritization methods on a mathematical-analytical basis	Complex systems of criteria for prioritization, sound decision-making processes
Agile Prioritization	Techniques based on team processes	Agile development in a cohesive team

6.2 Approaches for prioritization

6.2.1 Ad Hoc Prioritization

- **One-criteria classification.** A prioritization technique frequently used in practice with three priority classes
  - *Mandatory:* Urgent requirements
  - *Optional:* No urgent requirements
  - *Nice-to-have:* Additional requirements that do not jeopardize the success of the system if not taken into account.
- **Ranking.** Selected stakeholders determine a ranking of requirements with regard to a specific criterion. This can be done well, e.g., in a workshop with sticky dots.
- **Top Ten Technique.** Like ranking, but limited to the selection of <n> most important requirements.
- **Kano classification.** See explanation in previous chapter 6.2.2.

6.2.2 Prioritization according to Kano method

The Kano model was described in chap. 2.2 introduced. It can be used as an aid in prioritization - see also chapter 4.3:

- Basic factors are "must" requirements
- Performance factors can be "must", "should", or "will"
- Enthusiasm factors as well, but for tactical reasons some enthusiasm factors should be represented at "must"

6.2.3 *Consider-All-Facts with Plus-Minus-Interesting (CAF+PMI).*

One collects evaluation criteria, evaluates the impact of the requirement with points between 1 and 6 (CAF) and also determines whether they contribute positively (+), negatively (-) or neutrally (|) to the evaluation (PMI). This results in a point value for each requirement.

Example: I want to buy a new bicycle. My goals are listed in the first column.

		Requirement 1 **"The bike should cost well under 2000,-- EUR."**	*Requirement 2* **"..."**
Targets	**Target priority (1..6)**	**PMI**	**PMI**
I don't want to limit myself financially or go into debt because of purchases.	3	+	...
I want to get lots of exercise in the fresh air and thus become as fit as possible.	5	|	...
I want to be able to show off to my friends the things I buy.	1	-	...
I would rather own a few really good things than buy "throwaway" products frequently.	4	-	...
Total		3 + (0*5) -1 -4 = **-2**	...

- Template for CAF+PMI here.
- Complete bike example here.

6.2.4 *Analytical prioritization*

- **Analytical Hierarchy Process (AHP)**. See freely available template from SCB Associates: https://www.scbuk.com/AHP%20Template%20SCBUK.xls.
    1. Enter requirements in left column (up to 15 in template)
    2. Set number of requirements
    3. Enter pairwise weighting in matrix (yellow fields)
    4. If necessary, observe consistency check (far right)
    5. Read prioritization from green table at far right

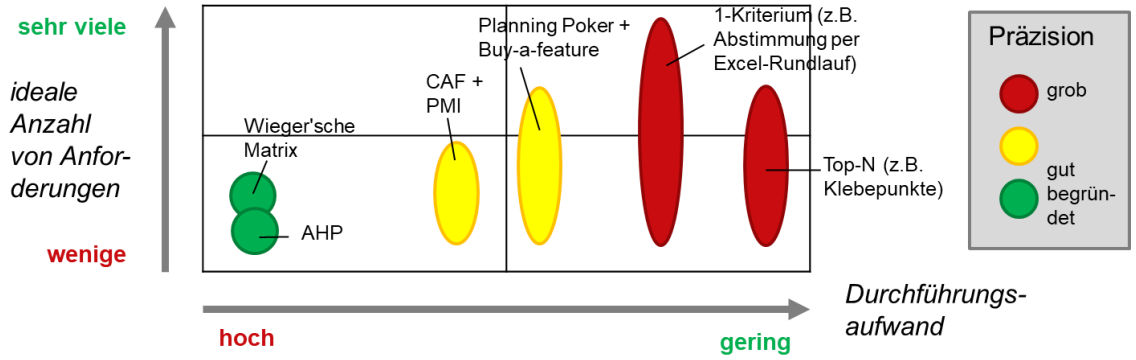
6.2.5 *Agile prioritization*

- **Buy a Feature:** Fits well for agile projects, e.g. to select features for the next release or sprint. Each requirement gets a price, depending on development cost / business value / risk / ... This can be determined e.g. with Planning Poker. Participants get play money and "buy" requirements. They have to form alliances in the process. This creates an orderly top-N selection.
  Rules:

1. No requirement should be able to be bought to a participant alone
2. all participants together should not be able to buy all requirements
3. Participants discuss to put money together for a specific feature
4. Watch discussion and note buy order!
5. **Example**: Team of 6, each participant 100 € (total: 600 €), features cost 120 €, 130 €, 150 €, 150 €, 200 €, 200 €, 350 € (total 1300 €)

6.2.6 When should you use which technique?

Ad hoc techniques provide a good framework for fast prioritization. A high number of requirements can be pre-filtered first and then further prioritized using other methods.



6.2.7 Further reading

- Ebert, 2014, pp. 224-229
- Pohl and Rupp, 2011, p. 117-123; p.132-136
- Rupp, 2014, pp. 482-494, 494-497, 498-501.

6.3 Conflict Management

With a high number of requirements, many stakeholders, and a politically charged environment, there can be conflicts in prioritization. Often there are underlying conflicts that are not necessarily factually based and are not immediately apparent. Conflict management helps to identify such a conflict, to analyze it and to resolve it with appropriate methods.

6.3.1 Glossary / Clarification of terms

up to here a requirements manager can act

Konfliktart	Mögliche Kennzeichen	Ursachen
Sachkonflikt	Einigkeit übers Ziel, Uneinigkeit über das Mittel/Vorgehen; Debatte	Mangel an Informationen; Unvollständige Fakten; Unterschiede in Interpretationen
Benennungs-konflikt	Verwendung von Synonymen und Homonymen	
Interessenkonflikt	Mangelnde Argumentation eigener Anforderungen Was könnte noch sein?	Nicht vereinbarte strategische Ziele; Unterschiede in subjektiven oder objektiven Interessen (Kosten, Qualität etc.)
Wertekonflikt		Unterschiedliche Wertesysteme (Kultur, Familie, persönliche Ideale wie Umwelt, Fairness etc.); unterschiedliche Bewertungssysteme von Sachverhalten
Beziehungskonflikt	Gegenseitiges Ablehnen von Anforderungen; wenig Interesse an der Auflösung von Anforderungskonflikten; negatives zwischenmenschliches Verhalten (Missachtung, Beleidigung)	Uneinigkeit über soziale oder hierarchische Beziehungen
Strukturkonflikt	Unbegründete Ablehnung von Anforderungen der Anderen	Ungleiche Macht- und Autoritätsverhältnisse
Rollenkonflikt	Personelle Aufteilung einer Rolle und der entsprechenden Funktion	vlt. weisst Du mehr dazu?

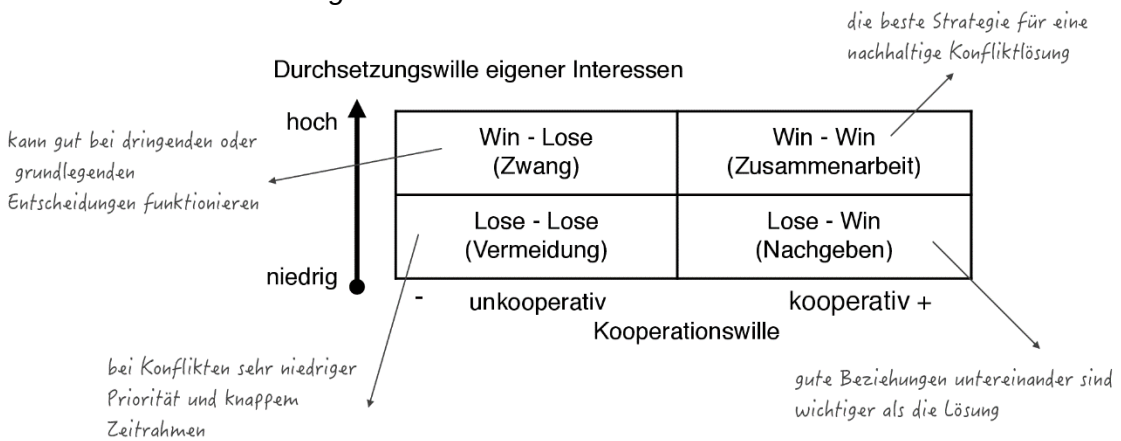
Diese Konfliktarten lassen sich nicht leicht entdecken. Also mehr Warum fragen an Stakeholder stellen?

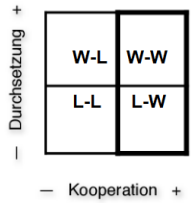
verstecken sich häufig hinter anderen Konfliktarten, z.B. Sach- oder Interessenkonflikten

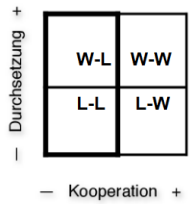
from here on trained coach necessary

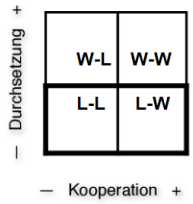
um die persönlichen Konflikte lösen zu können, muss der enthaltene Sachkonflikt isoliert werden

6.3.2 Conflict resolution strategies









Methoden	Beschreibung
Annäherungsmethoden	Ziel: eine gemeinsame Lösung des Konfliktes durch Diskussion und Austausch von Meinungen zu finden Besonderheiten: • zeitaufwändig; • können gut bei Interessens- und Wertekonflikten eingesetzt werden • offene Kommunikation der Konfliktparteien über ihre Interessen und Werte, sowie Wertschätzung untereinander sind notwendig für den Erfolg der Konfliktauflösung
Einigung	Durch den Austausch von Informationen, Meinungen und Argumenten sollen sich alle am Konflikt beteiligten Parteien/Stakeholder für genau eine aus bereits vorhandenen Lösungsalternativen entscheiden
Konsens	Ausarbeitung einer neuen Lösung, die sich auf die besten Aspekten bestehender Lösungsalternativen basiert, oder Entwicklung einer komplett neuen und kreativen Lösung
Kompromiss	Ausarbeitung einer gemeinsamen Lösung
Variantenbildung *kann gut bei harten verbalen Konfrontationen funktionieren*	Konfliktparteien entscheiden sich für die Umsetzung mehrerer Lösungsalternativen (Beispiel: Umsetzung von verschiedenen Konfigurationen für jede Benutzerrolle) *d.h. zusätzliche Komplexität des Systems und Kosten*
Nicht-Annäherung	Die Konfliktparteien beschließen gemeinsam, dass sie sich für keine Lösungsalternative entscheiden
Abstimmungs- und Weisungsmethoden	Ziel: eine Entscheidung durch Abstimmung oder Weisung zu treffen Besonderheiten: • zeitsparend; • gut geeignet bei einer großen Anzahl an beteiligten Konfliktparteien/ Stakeholdern; • können gut bei schwer zu lösenden Konflikten funktionieren, bei denen andere Techniken zu keiner Lösung führen
Ober-sticht-Unter *damit kann man die härtesten Konflikte lösen*	Der Konflikt wird anhand Hierarchie (organisatorischer Rang) der Konfliktparteien entschieden: • (org. Rang der Konfliktpartei 1 < org. Rang der Konfliktpartei 2) : Konfliktpartei 2 gewinnt den Konflikt • (org. Rang der Konfliktpartei 1 = org. Rang der Konfliktpartei 2) : eine gemeinsame übergeordnete Instanz (wie z.B. einen Vorgesetzter) entscheidet sich für eine Lösung, die sie auch begründet. Vor der Entscheidung werden ihr alle Lösungsalternativen von allen Konfliktparteien mündlich oder schriftlich vorgestellt werden.
Abstimmung *nicht geeignet bei stark ausgeprägten Lagern (z.B. Buchhaltung und Marketing)*	Am Anfang werden alle Losüngsalternativen in einer Plädoyer-Session präsentiert und anschließen gibt jeder Konfliktbeteiligte seine Stimme für eine Lösungsvariante. Die Lösungsvariante mit der höchsten Stimmenanzahl wird umgesetzt. *am besten geheim*
Geringstes Übel	Am Anfang werden alle Lösungsalternativen vorgestellt. Danach werfen die Konfliktparteien abwechselnd die für sie übelste aus den verbleibenden Alternativen raus. Dies wird solange wiederholt, bis für alle am wenigsten üble Lösung bleibt.
Unterstützende Techniken	Das sind Techniken, die oben genannte Methoden unterstützen können, um eine begründete Entscheidung zu treffen. Sie ermöglichen die Zerlegung komplexer Probleme in kleine Fragestellungen, die leicht zu analysieren und bewerten sind. Die bekanntesten analytischen Methoden sind: • Plus-Minus-Interesting (PMI) • gewichtetes Plus-Minus-Interesting • Consider-All-Facts (CAF) • Entscheidungsmatrix

6.3.3 Further reading

- Ebert, 2014, pp. 224-229
- Pohl and Rupp, 2011, p. 117-123; p.132-136
- Rupp, 2014, pp. 482-494, 494-497, 498-501.

7 Use Cases

Use cases are easy to write and read. They are very useful for detailing the functional requirements of the system to the extent that they can be implemented by a development team. They are also very suitable for communication when the requirements and development team do not know each other (nearshore / offshore development).

7.1 Glossary / Clarification of terms

Term	Definition	Do you need for ...
Use Case	Task field to be supported by the (IT) system, formulated from the point of view of one or more users (actuators) and triggered by an actuator.	Detailed requirements definition on the basis of which the architecture design and implementation can take place.

7.2 Template for a single use case

There are many suggestions for use case templates in the literature; this is a minimal one that has worked well in practice. Extend it where necessary, but you should not omit any fields.

ID	Unique identifier of the use case
Name	Name of the use case (usually object + verb)
Description	Short description of the use case
Trigger actuator	Actor that starts the use case; can also be a surrounding system
More actuators	Other actors involved, can be possible surrounding systems
Trigger	Event(s) that triggers the use case
Precondition	Condition to be able to start the use case
Postcondition	State after successful execution of the use case
Main scenario	Sequence of steps that describes the flow of the use case and leads to success. **(see also definition "main/alternative/exception scenario" in chap. 2.4.3 on p. 18)**
Alternative scenario	Step sequence that deviates from the main scenario, but still leads to success; describes an exceptional situation
Exceptional scenario	"Failure" scenario that does not lead to success (then with its own postcondition).

7.2.1 Numbering of the scenario steps

- Steps in the **main scenario** are numbered consecutively from 1. *Example:* 1, 2, 3, 4, ...
- **Alternatives** / **exceptions to** a particular step: corresponding number and "a, b, ...". *Example:* Alternative to step 2: "2a".
- If multiple alternate / exception steps belong to a main scenario step: number in the form "5a1, 5a2, 5a3" etc.

7.3 Example (customer portal of an insurance company for damage reports)

ID	KundPort_12
Name	Report damage
Description	Customer reports damage
Trigger actuator	Customer
More actuators	Insurance backend claims system
Trigger	Customer has a claim, e.g. car accident
Precondition	Customer is actually insured with this insurance
Postcondition	Damage has been reported to insurance company
Main scenario	1) Customer identifies himself with customer ID and password. 2) Customer enters date and description of damage 3) Portal forwards the damage to backend damage system 4) Backend system confirms receipt 5) Portal sends incoming report to customer via email
Alternative scenario	1a) Customer writes a letter to the insurance company, describing the damage. 2a) Clerk reads the letter and enters the date and description of the damage into the system instead of the customer *(then continue with 3).* 4b1) Backend system does not respond 4b2) System sends an email with the damage data to the person responsible for the backend system, so that he/she ensures the transfer of the data *(then continue with 5)*
Exceptional scenario	4c) Customer's insurance coverage is suspended because the premium has not been paid. Backend system sends corresponding message to the portal. 5c) Portal passes on the rejection of the damage report to the customer **Postcondition in this case:** damage is not considered reported.

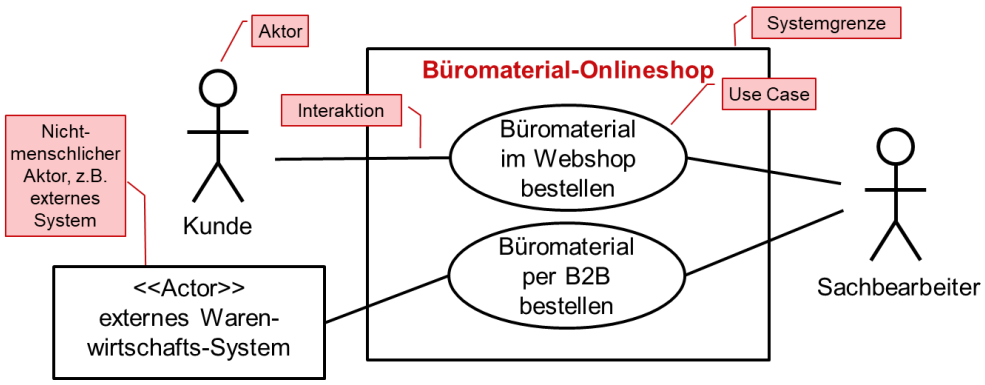
7.3.1 Checklist: Use Case too big / too small? (after Cockburn p. 57 ff.)

- *UC possibly too small?* **User Happiness Test.** "Is the user satisfied after the use case?"
  - Ex. for **ok**: Use case above, damage is reported.
  - Example for **not ok**: Customer sends email with damage report to insurance company (and this ends the use case). Then processing and feedback on the part of the insurance company is missing.
- *UC possibly too large?* **Coffee Break Test.** "After a UC like this, users take a coffee break."
  - Example for **ok**: Above UC should be doable in about 10 min.
  - Ex. for **not ok**: If the above UC also included settlement of the claim (which takes days to weeks).

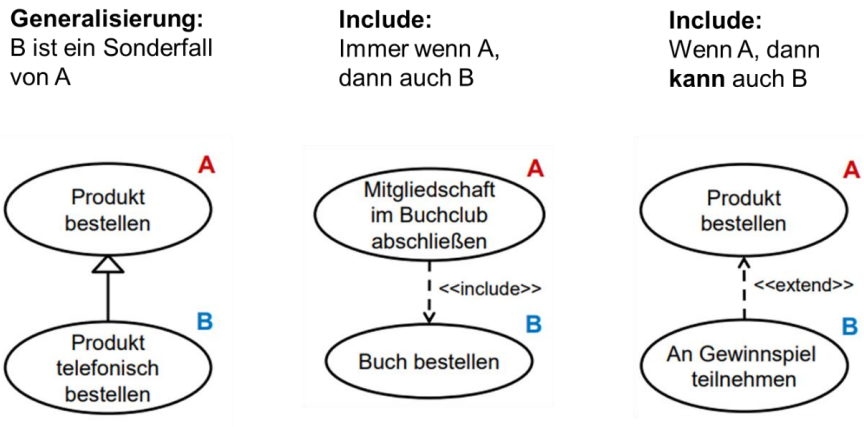
7.3.2 Extract use cases from textual requirements

1. mark verbs
2. syntactic cleanup
    a. Striking nouns, resolving auxiliary constructions
    b. Delete modal verbs (can, will, would like to, should, must, may)
    c. Convert passive to active
    d. Use presence, delete tense auxiliary verbs.
3. semantic cleanup
    a. Goal and context but without functionality - delete
    b. Delete repetitions
4. derivation of the use cases
    a. Name: object (singular) + verb (infinitive)
    b. Actor: infer from context, always singular

7.4 Overview of use cases with the help of use case diagrams



Relationships between use cases:



7.5 Further reading

Cockburn, 2000, pp. 81-110; 132-138.
Leffingwell, et al. 2003, pp. 147-156.
Pohl and Rupp, 2011, pp. 75-8ß; 91-95.
Rupp, 2014, pp. 217-219

8 Quality assurance

Quality assurance in requirements management serves to detect and eliminate discrepancies between the elicited requirements and the customer's wishes. The measures applied here can be divided into **constructive** and **analytical** quality assurance.

8.1 Glossary / Clarification of terms

Term	Definition	Do you need for ...
Constructive quality assurance	Assists in the creation of high quality requirements	Quality-assuring working principles **during** requirements determination
Analytical quality assurance	Verification of the quality of requirements	Testing and improvement of the requirement quality **after** determination

8.2 Quality criteria

8.2.1 ... for requirements in general

Criterion	Definition
Correctness	Describes without errors the property that the system should fulfill
Completeness	Presence of all necessary information
Uniqueness	The requirement is precisely formulated and cannot be interpreted more than once
Consistency	The requirement does not contradict itself or other requirements
Validity	There is consensus among all stakeholders on the requirement to
Prioritization	Requirements are ranked by evaluating the stakeholders
Verifiability	The fulfillment of the requirement is verifiable
Traceability	The requirement has a unique name (key) and provides all relevant context information (source, relation to other requirements)
Comprehensibility	All involved are able to understand the requirement
Feasibility	The requirement is feasible

8.2.2 ... especially for documents

Criterion	Definition
Structured	Correct, comprehensible structure of the document
Topicality	The document reflects the current status of the requirement
Modifiability	Changes to the requirement should be easy to make
Accessibility	Controlled access for all stakeholders
Projectability	Different views of the document possible, depending on the user role
Relevance	The document does not contain information irrelevant to the problem description

8.3 Quality assurance methods

At the beginning of a project, mainly **constructive** techniques are used. If the requirements identified are stable, **analytical** techniques should be used instead.

General principles for testing requirements	Checklist for **constructive quality assurance (** during requirements determination)
• Involvement of the right stakeholders • Separation of troubleshooting and error correction • Testing from different views • Repeated test	• Consistently defined and adhered to the type and level of detail of the requirement? • Goal and scope of the system comprehensively described? • All requirements considered? • Requirements documented using standard templates?

8.3.1 Analytical quality assurance techniques

Technology	**Works how?**	**When to use?**
Perspective based reading	Reviewer reads a document and explicitly assumes a specific stakeholder role (e.g. customer, user, architect, tester, developer, ...)	Mostly in combination with review techniques
Review: Opinion	Author asks reviewer about opinion on document	Small projects, fast implementation, non-formal
Review: Walkthrough	Author guides group of reviewers through the artifact so that questions and comments can be contributed. Reviewers have read document in advance. Results are recorded.	Medium to large projects without major quality constraints
Review: Inspection	Very formal statement with different phases: Planning, preliminary meeting, individual preparation, review meeting, follow-up and evaluation. Very heavyweight.	Projects with high quality requirements, very heavy / time intensive
Check requirements through **prototypes**	A portion of the system (often referred to as a "horizontal cut-through" is prototyped according to requirements and evaluated by stakeholders.	Technical / critical systems, resource intensive

8.4 Further reading

- Pohl and Rupp, 2011, pp. 101-117
- Rupp, 2014, pp. 287-298
- Schienmann, 2002, pp. 176-185

9 Creating an agile backlog

Agile requirements (unlike document-centric requirements elicitation, which is aligned with a waterfall development process) are initially only roughly recorded. A detailed specification is only made immediately before implementation, then usually as user stories.

9.1 Glossary / Clarification of terms

Term	Definition	Do you need for ...
(Investment) Theme	Common "mission" for a large interrelated set of features, taking into account business strategies	In general, the theme describes a new IT system to be created, or one of its main areas.
Epic	Roughly granular, general requirement that is further detailed in User Stories.	An Epic roughly specifies an area of the IT system that a development team implements over an extended period of time. • 1 Theme = <n> Epics
User Story	Describes concretely what a system should do from the point of view of a user.	Requirement unit defined for a development sprint (2-4 weeks). • 1 Epic = <n> User Stories • I.a.: Epic >= Use Case >= User Story • User story must be completed in one sprint
Task	Individual tasks for a story, e.g. writing specific pieces of code	Detailing of a user story; in contrast to the user story, the task usually has a technical reference. • 1 User Story = <n> Tasks
Iteration (Sprint)	Fixed time unit (2-4 weeks) in which the development team implements a certain amount of user stories.	Features are planned in releases and implemented in iterations / sprints.
Release	Long-term feature timing.	
Product Backlog	Collection of prioritized requirements (Investment Themes, Epics, User Stories).	The further away the implementation date is, the more coarsely the requirement can be formulated.
Sprint Backlog	Extracting user stories from the product backlog to implement them in an iteration / sprint.	Work plan for the sprint.
Product Owner	Prioritizes customer requirements and plans release.	The requirements manager in agile projects.

9.2 Demarcation (Investment) Theme - Epic - User Story - Task

The following example is adapted and translated from Cohn (n.d.). The requirements describe in extracts the requirements of a system for managing training services. The following applies: **TH** = (Investment) Theme, **EP** = Epic, **US** = User Story, **TA** = Task.

- **TH:** Trainer profile management

- **TH:** Management of courses and events
- **TH:** Provide documentation
  - **EP:** As a user of the portal, I can read FAQ entries.
  - **EP:** As a subject administrator, I can maintain FAQ entries.
    - **US:** As a technical administrator, I can add a new "FAQ" entry with fields for heading and explanation text, so that I only have to answer recurring customer questions once
    - **US:** As a subject administrator, I can format my FAQ explanation text with wiki markup so that I can easily structure the text and make it readable.
      - **TA:** Definition of the format for the wiki markup
      - **TA:** Implement configuration database for the allowed wiki markup.
      - **TA:** Implement Wiki => HTML Converter

9.3 Template: User Story

Component	Content	Example
As <role>,	Role of the user from whose perspective this requirement is described	As a professional administrator of our customer portal,
I can <activity>,	Main part of the user story: what should the IT system enable the user to do?	I can add a new "FAQ" entry with fields for heading and explanation text,
so that <business value>.	Reason why the user story makes sense to the user (useful for prioritization).	so that I only have to answer recurring customer questions once.

9.3.1 Wording Rules: Make sure that "INVEST" applies to user stories

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

9.3.2 Division rules for user stories (after Leffingwell, 2010, p. 112f.)

There is a tendency to formulate user stories "too big". Rule of thumb: A user story must be so small that **a developer** can implement at least **one user story** (preferably several) **in a sprint.**

If the development team gives feedback during sprint planning that the user story is too big, the product owner can break it down into several individual stories using the following rules.

No.	Method of splitting	Example before	Example behind
1	Decomposition of the activity steps	As a subject administrator, I want to edit a blog post, save it, and then publish the changed version.	• ... I would like to edit and save ... • ... I would like to trigger the publication ...
2	Decomposition according to CRUD (Create /	As a customer, I can manage my account myself, ...	• ... Create account ... • ... Read account data ...

	Read / Update / Delete)		• ... Change account data ... • ... Delete account ...
3	Decompose by data attribute	... I would like to filter the customer data by address attributes ...	• ... by zip code ... • ... by state ...
4	Decompose by Data type	As a customer, I would like to be able to pay by credit card, ...	• ... by Visa card ... • ... by Master Card ...
5	Decompose by database	As a customer in the EU, I would like to read emails from the portal in my national language, ...	• As a customer in Germany ... • As a customer in France ... • As a customer in Italy ...
6	Add Variation: "Simple solution first	As a subject administrator, I can edit a blog post to "WYSIWIG", ...	• ... as simple ASCII text ... • ... as ASCII text with wiki markup ... • ... after "WYSIWIG" ...
7	Add variation: Input type	As a clerk, I can edit the price of a quote, ...	• ... in the price field of the details view ... • ... additionally via inline editing in the overview table ...

9.4 Further reading

- Cohn, 2004, pp. 17-30 and 75-84.
- Cohn, (n.d.)
- Leffingwell, 2010, pp. 31-45, 83-92, and 99-117.

10 Appendix: Literature

- Bono, E. de. (1989). Six-color thinking. A new training model. Düsseldorf: Econ.
- Calabria, T. (2004). An introduction to personas and how to create them. Retrieved April 30, 2015, from http://www.steptwo.com.au/papers/kmc_personas/index.html
- Cockburn, A. (2000). Writing effective use cases. Boston: Addison Wesley.
- Cohn, M. (2004). User Stories Applied: For Agile Software Development (1st ed.). Addison-Wesley Professional.
- Cohn, M. (n.d.). Product Backlog Example. Retrieved January 20, 2019, from https://www.mountaingoatsoftware.com/agile/scrum/scrum-tools/product-backlog/example
- Cooper, A. (1999). The Inmates are Running the Asylum: Why Hightech Products Drive Us Crazy and How to Restore the Sanity (First Printing). Indianapolis, Ind: Sams.
- Ebert, C. (2014). Systematic requirements engineering: elicit, document, analyze, and manage requirements (5th, revised ed.). Heidelberg: dpunkt.verlag GmbH.
- Gürtler, J., & Meyer, J. (2013). 30 minutes of design thinking. Offenbach: GABAL.
- Kintz, M. (2007). Personas (Requirements engineering advanced seminar). Retrieved from http://www.iste.uni-stuttgart.de/fileadmin/user_upload/iste/se/teaching/courses/hsre/res-WS2007-2008/HSRE-WS0708-Maximilien_Kintz-Personas.pdf
- Leffingwell, D. (2010). Agile software requirements: Lean requirements practices for teams, programs, and the enterprise (1st ed.). Upper Saddle River, NJ: Addison Wesley.
- Leffingwell, D., Widrig, D., & Yourdon, E. (2003). Managing software requirements: A use case approach (0002 ed.). Boston: Addison Wesley Pub Co Inc.
- Pichler, R. (2013). Agiles Produktmanagement mit Scrum: Erfolgreich als Product Owner arbeiten (2nd, corrected edition). Heidelberg: dpunkt.verlag GmbH.
- Pohl, K. (2008). Requirements engineering: fundamentals, principles,techniques (2nd, corrected edition.). Heidelberg: dpunkt.Verlag GmbH.
- Pohl, K., & Rupp, C. (2011). Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level (3., korrigierte Auflage). dpunkt.verlag GmbH.
- Rupp, C., & the SOPHISTs (2014). Requirements engineering and management: From practice from classical to agile (6th, updated and expanded edition). Munich: Carl Hanser Verlag GmbH & Co. KG.
- Schienmann, B. (2001). Continuous requirements management . Processes - techniques - tools (1st ed.). Munich ; Boston et al: Addison-Wesley.
- SOPHISTs, the. (2014). Requirements engineering - the little RE primer. Self-published. Retrieved from https://www.sophist.de/fileadmin/SOPHIST/Puplikationen/Broschueren/RE-Broschuere_Komplett_Final_-_Update_1.pdf
- Weit e.V. (2006). The V-Modell XT, Version 2.2. http://ftp.tu-clausthal.de/pub/institute/informatik/v-modell-xt/Releases/2.2/V-Modell-XT-Gesamt.pdf, retrieved 20 Jan. 2019.
- Werner, S. (2006). Fundamentals of program design engineering 1. Retrieved from http://ti.uni-due.de/ti/de/education/teaching/ss06/pet/folien/Folien%201.pdf