Technology
Arts Sciences
TH Köln

Requirements Engineering

Master Digital Sciences
Stefan Bente, Fabian Krampe

Referenz

Diese Kurzreferenz ist dazu gedacht, als „Spickzettel“ im Projektalltag zu dienen. Die in der Veranstaltung behandelten Inhalte sind hier noch einmal in kompakter Form zusammengetragen. Die Nummerierung der Kapitel entspricht der Reihenfolge im pragmatischen Vorgehensmodell (nächste Seite) und in der Veranstaltung.

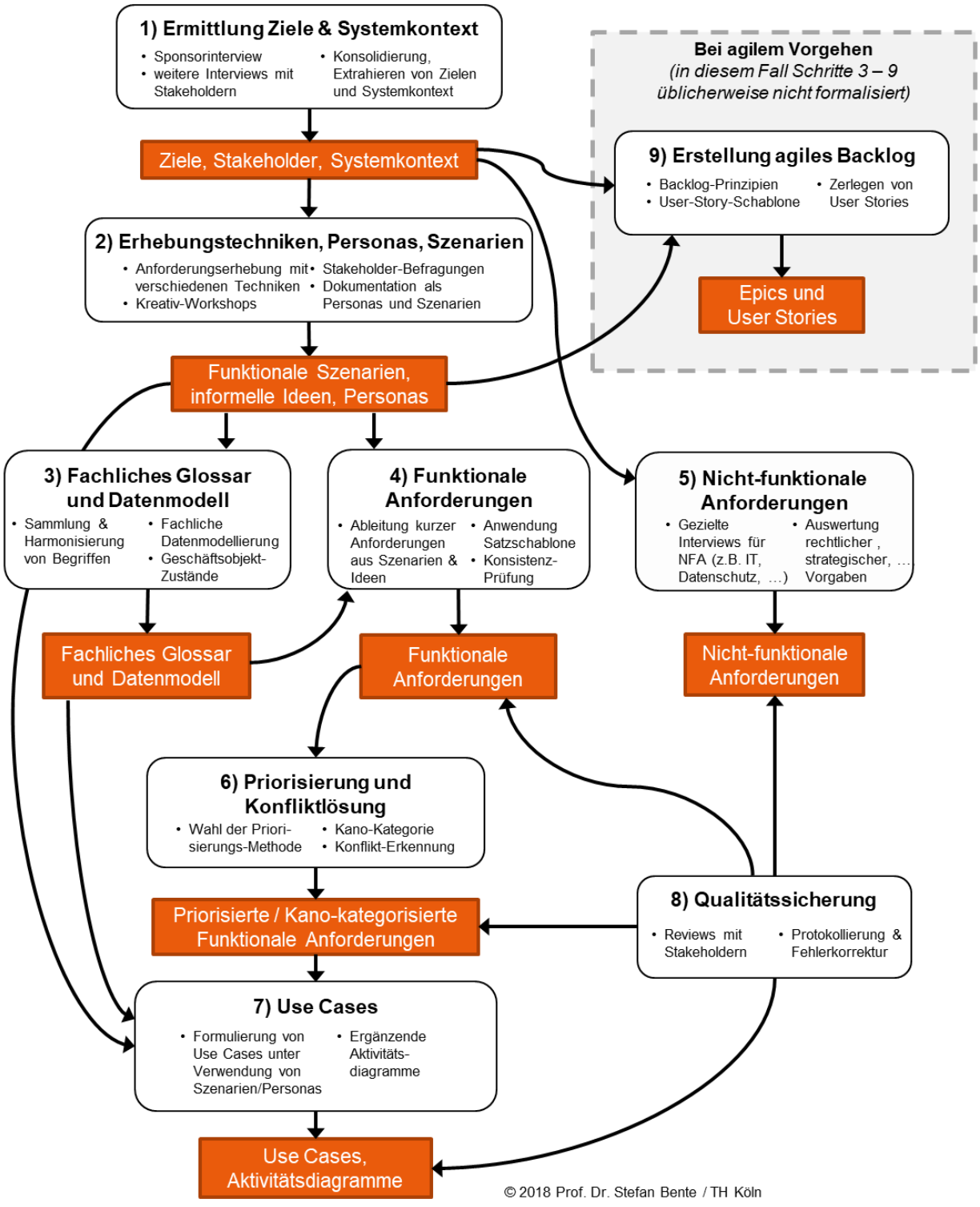
Die einzelnen Kapitel sind durchgehend nach einem ähnlichen Schema strukturiert:

- Glossar / Begriffsklärung
- Schablonen und Templates
- Gängige Vorgehensweisen / Techniken / Methoden, mit Empfehlungen
- Verweis auf weiterführende Literatur für Details, die in der Kurzreferenz keinen Platz haben

Inhalt

0 Einführung und Übersicht

Das folgende pragmatische Vorgehensmodell liegt dieser Veranstaltung zugrunde. Nutzen Sie es als Leitlinie und passen Sie es für Ihre Projekte an die Gegebenheiten an.



0.1 Glossar / Begriffsklärung für grundlegende Dokumente

Die folgenden Dokumente werden Ihnen während der Anforderungsermittlungs- und Umsetzungsphase eines Projekts begegnen. Mit dem Inhalt dieser Veranstaltung sind Sie in der Lage, Beiträge zu diesen Dokumenten zu leisten.

Begriff	Definition	Braucht man für …
Lastenheft	Erstellt durch den **Auftraggeber**. Enthält die Gesamtheit der Forderungen an Lieferungen und Leistungen eines Auftragnehmers (Systemvision, Systemziele, Funktionen und Qualitäten, Kontextaspekte).	Fasst alle **Anforderungen** an ein IT-System zusammen. Mit den Schritten 1-8 decken Sie den wesentlichen Teil der Lastenheft-Erstellung ab. Das Lastenheft bildet die Grundlage für eine **Ausschreibung**.
Pflichtenheft	Vom **Auftragnehmer** erarbeitete Spezifikation. Beschreibt die Umsetzung des vom Auftraggeber vorgegebenen Lastenhefts, die Architektur und geplante Details der Realisierung.	Bildet des Antwort des Auftragnehmers auf das Lastenheft.

0.1.1 Vorschlag für Gliederung eines Lastenhefts

- Einleitung
  - Problembeschreibung
  - Systemkontext
  - Übersicht der Stakeholder
  - Ziele des Systems
- Produktanforderungen
  - Personas
  - Szenarien
  - Funktionale Anforderungen
- Use Cases
- Fachlicher Systemkontext
  - Geschäftsprozesse, -Vorfälle
  - Geschäftsobjekte
  - Weitere fachliche Rahmenbedingungen
- Technischer Systemkontext
  - HW- und SW-Konfiguration
  - Schnittstellen zu anderen Systemen
  - Sonstige technische Rahmenbedingungen
- Entwicklungs-Rahmenbedingungen
  - Vorgehensmodell
  - Entwicklungswerkzeuge
- Zeit- und Kostenrahmen
  - Meilensteine
- Verzeichnisse
  - Referenzdokumente

0.1.2 Weiterführende Literatur

- Pohl, 2008, S. 232-236 und 251-257
- Rupp, 2014, S. 36-39
- Schienmann, 200, S. 141-148
- Weit e.V. (2006), insb. S. 1-61

1 Ermittlung von Zielen und Systemkontext

Der *Systemkontext* beschreibt Aspekte der Umgebung, die eine relevante Beziehung zum System besitzen. Der Systemkontext umfasst Anforderungsquellen, wie Dokumente, Systeme in Betrieb und *Stakeholder*, die für die Anforderungsermittlung genutzt werden. *Ziele* der Stakeholder dienen als erste Beschreibung intendierter Nutzungspraxis des Systems und können genutzt werden, um die Relevanz und Vollständigkeit der später erarbeiteten Anforderungen zu überprüfen.

1.1 Glossar / Begriffsklärung

Begriff	Definition	Braucht man für …
Systemkontext	Teil der Umgebung eines Systems, der für die Definition und das Verständnis der Anforderungen des betrachteten Systems relevant ist.	Identifikation von Anforderungsquellen und Vermeidung von missverständlichen Anforderungen.
Stakeholder	Person oder Organisation, die direkten Einfluss auf die Anforderungen des betrachteten Systems hat, und / oder ein Interesse an dem System.	Wichtige Quelle für Systemkontext, Ziele und Anforderungen.
Ziel	Intentionale Beschreibung eines charakteristischen Merkmals des zu entwickelnden Systems bzw. des zugehörigen Entwicklungsprozesses.	Schafft gemeinsames Systemverständnis. Anforderungen dienen zur Erreichung dieses Ziels.

1.2 Allgemeines Vorgehen

- **Stakeholder** identifizieren und dokumentieren (Stakeholderschablone)
  - Stakeholder klassifizieren (Einfluss/Motivations-Matrix, siehe Kap. 1.3.1)
  - Maßnahmen für den Umgang mit einzelnen Stakeholdern ableiten
  - Stakeholder befragen (Interviews)
- Ist-Situation wertungsfrei erheben, Probleme notieren, keine Lösungen
  - Ist-Situation bewerten, Ursachen identifizieren
- **Ziele** ableiten und den Stakeholdern zuordnen
  - Ziele verfeinern und dokumentieren (Zielschablone, 7 Regeln)
- Systemkontext definieren

1.2.1 Besondere Herausforderungen und Fallstricke

- Der **Systemkontext** kann zu Beginn meist *nicht vollständig spezifiziert* werden. Die Systemgrenze muss erst am Ende des Anforderungs-Ermittlungs-Prozesses klar definiert sein. Deshalb nicht zuviel Aufwand in die initiale Systemkontext-Bestimmung investieren.
- **Vergessene Stakeholder** haben fehlende Anforderungen zur Folge. Deshalb hier *besonderes Augenmerk* auf die Bestimmung.
- **Zielkonflikte** zwischen Stakeholdern sind oft nicht direkt sichtbar.

1.3 Stakeholder

1.3.1 Templates und Schablonen

Prüffragen für die Stakeholderermittlung

Die folgenden Fragen helfen bei der Zusammenstellung der Stakeholder-Listen.

- Wer sind die Nutzer des Systems?
- Wer ist der Kunde des Systems?
- Wer wird das neue System pflegen?
- Wer ist noch von den Outputs des Systems betroffen?
- Existieren weitere Personen oder Organisationen, die das System interessiert?
- Wer wird bewerten / genehmigen, ob / wann das System ausgeliefert oder bereitgestellt wird?

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

Stakeholder-Kategorisierung

1.3.2 Ermittlungstechniken / Methoden: Stakeholderinterview

Planung:

- Als Grundlage z.B. Kaiser (2014).
- Interviewziel erarbeiten und grobe Informationen über den Stakeholder sammeln
- Interviewleitfaden erstellen
- Rollen verteilen (Interviewer, Protokollant,…)

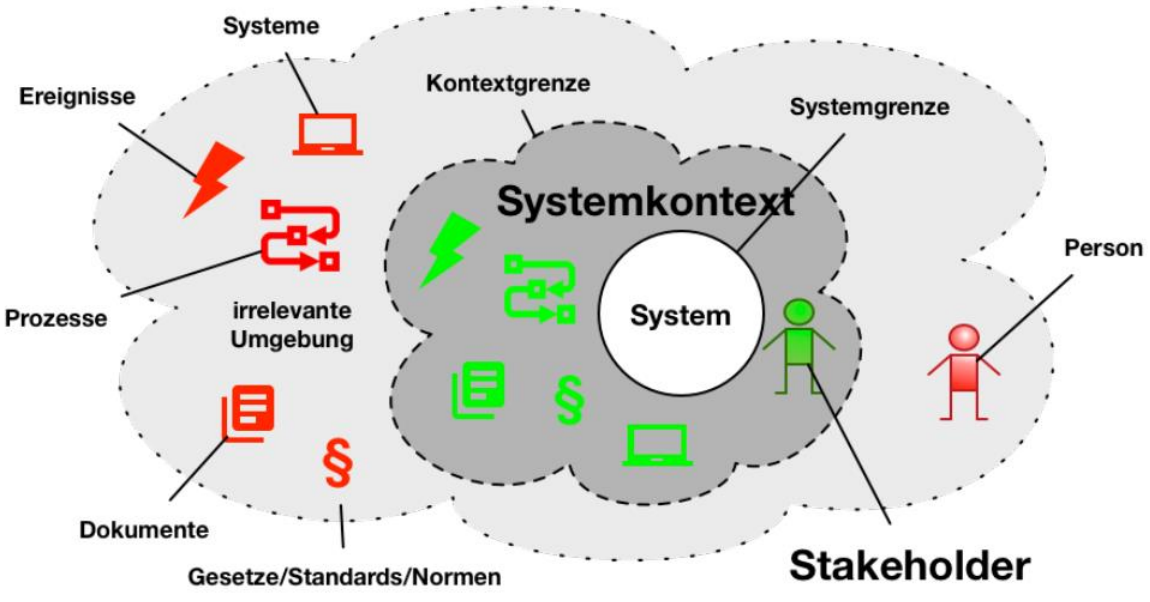
Interviewleitfaden:

- Einleitung:
  - Sich bzw. das Team vorstellen
  - Ziele des Interviews nennen
  - Den Stakeholder motivieren, frei heraus zu reden und zu antworten.
- Hauptteil:
  - Was ist Ihre Rolle bezogen auf das Produkt?
  - Welche Erwartungen haben Sie persönlich an dieses Projekt?
  - Haben Sie Bedenken / Sorgen, bezogen auf das Projekt? Wenn ja, welche?
  - Was soll dieses Produkt oder diese Dienstleistung sein/leisten?
  - Was soll mit dem neuen System (nicht) erreicht werden?
  - Wie würden Sie persönlich den Erfolg dieses Projektes definieren?
  - Welche Prozesse werden ausgeführt, und wie?
  - Welche Probleme treten bei der Ausführung von Prozessen auf?
  - Gibt es Regelwerke, die zu berücksichtigen sind?
  - Wie soll das System eingesetzt werden?
  - Welche Tools / Methoden werden zur Ausführung von speziellen Aufgaben genutzt?
  - Gibt es eine Technologie, die angewandt / eingebunden werden soll?
- Schluss:
  - Nach weiteren möglichen Stakeholdern fragen
  - Weitere Zusammenarbeit klären (z.B. „Wie würden Sie gerne in das Projekt mit einbezogen werden und was ist der beste Weg, Sie zu erreichen?“)

Nachbereitung:

- Dem Stakeholder das Protokoll des Interviews zur Verfügung stellen
- Bestätigen lassen, um evtl. Missverständnisse aufzulösen (Review)

1.4 Systemkontext



1.5 Ziele

1.5.1 Sieben Regeln für die Zieldefinition (nach Pohl, 2008)

1. Formulieren Sie Ziele **kurz und prägnant**, keine Füllworte, keine geschachtelten Sätze
2. Verwenden Sie **Aktiv-Formulierungen, da hier der** Akteur benannt werden muss.
   - *Nicht:* Die Dauer der Berichterstellung soll gegenüber dem jetzigen Zustand halbiert werden.
   - *Sondern:* Sachbearbeiter können Berichte in der halben Zeit wie bisher erstellen.
3. Formulieren Sie **überprüfbare** Ziele.
4. Wenn Ihr Ziel nicht überprüfbar ist, zerlegen Sie es in überprüfbare **Teilziele**.
5. Formulieren Sie den **Mehrwert** des Ziels (welcher Nutzen wird damit erreicht?).
6. Geben Sie eine **Begründung** für das Ziel.
7. Vermeiden Sie **Lösungsansätze**.

1.6 Weiterführende Literatur

- Broadbent & Kitzis, 2004, S. 51-55
- Kaiser, Robert, 2014, Qualitative Experteninterviews: Konzeptionelle Grundlagen und praktische Durchführung. Springer-Verlag[1].
- Leffingwell, 2003, S. 43-57
- Pohl, 2008, S. 89-108
- Pohl, 2011, S. 21-32
- Rupp, 2014, S. 62-77

[1] Als elektronische Ressource in der Hochschulbibliothek verfügbar

2 Erhebungstechniken, Personas, Szenarien

2.1 Erhebungsmethoden

2.1.1 Glossar / Begriffsklärung / Vor- und Nachteile

Techniken	Definition	Braucht man für	Vorteile	Nachteile
Befragungs-techniken	Ermittlung von Wünschen und Bedürfnissen der Stakeholdern durch gezielte Fragenstellungen	Erste Erhebung der „naheliegenden" Anforderungen, Erkennen von **Leistungs-faktoren** (siehe Kano-Modell in Kap. 6.2.2, S. 27)	• Gezielte Befragung möglich • Anforderungsermittlung gut strukturierbar, Ergebnisse gut abgleichbar	• Funktioniert nur mit Stakeholdern, die sich der Anforderungen bewusst sind und Gedanken verbalisieren können • Qualität der Antworten hängen stark von der Qualität der Fragen ab
Dokumenten-zentrierte / artefaktba-sierte Tech-niken	Anforderungs-Analyse auf Basis eines existierenden Systems, Dokumenten oder anderen Artefakten	Ermittlung oder Ergänzung von Anforderungen, wenn keine Stakeholder zur Befragung zu Verfügung stehen	• Übersicht über Funktionalitäten des Altsystems => Vollständigkeit des Neusystems • Wiederverwenden von Lösungen und Erfahrungen => Zeiteinsparung und Kosten-	• Nicht geeignet bei vielen potentiellen Änderungen • Gute Dokumentation / Struktur des Altsystems ist Voraussetzung • u.U. werden auch Fehler übernommen
Kreativitäts-techniken	Techniken zur Nutzung von Phantasie und Kreativität (bei der Ermittlung von Anforderungen)	Aufbrechen von starren Denkmustern, Erkennen von **Begeisterungsfaktoren** (siehe Kano-Modell)	• Neue, innovative Ideen • Hilft, Denkmuster aufzubrechen • Gibt manchmal ungewöhnliche Ideen	• Führt zu undetaillierten / unstrukturierten Anforderungen • Mitarbeit / Akzeptanz der Stakeholder zentral für Erfolg • Passt nicht in jede Unternehmenskultur
Beobach-tungstechni-ken	Erkennung und Analyse von Anforderungen durch Beobachten / Dokumentieren der Arbeitsschritte von Stakeholdern und potentiellen Nutzern	Dokumentation von Anforderungen, die bei der schriftlichen Ausarbeitung häufig unter den Tisch fallen, weil man sie für selbstverständlich hält (**Basisfaktoren**, s. Kano)	• Erkennen ineffizienter Prozesse durch externe Sicht • Geeignet, falls Stakeholder ihr Know-How nicht sprachlich ausdrücken können	• Stakeholder könnten sich überwacht fühlen und deshalb den Prozess verfälschen • Nicht bei Neuentwicklungen einsetzbar

2.1.2 Befragungstechniken

- **Interview**: Geführtes Interview anhand eines Frageleitfadens. Sinnvoll, um einzelne Experten gezielt und in der Tiefe zu befragen. Ergebnissicherung ist von wesentlicher Bedeutung.
  - Liste von Fragen vorbereiten, aber Flexibilität zulassen. Siehe Anleitung für Stakeholderinterviews (Kap. 1.3.2 auf S. 2).
  - Respektieren Sie die Meinung bzw. das Wissen des Interviewpartners!
- **(Online-)Umfrage** Durch eine Umfrage lassen sich viele potentielle Nutzer mit geringen Aufwand erreichen. Zusätzlich fühlen sich die Probanden/Befragten nicht einem Druck durch eine reale Person ausgesetzt, so wie es bei einem Interview der Fall ist
  - Sorgfältige Vorbereitung mit Testläufen nötig.
  - Wichtig: Freitextfelder zur Nennung von bisher nicht berücksichtigten Aspekten.
- **Selbstaufschreibung:** Ausgewählte Stakeholder dokumentieren ihre Tätigkeitsfelder sowie ihre Anforderungen, Änderungs- und Optimierungsvorschläge.
- **On-Site-Customer:** Ein Stakeholder steht dem Entwicklerteam beratend während der Konzept- und Umsetzungsphase zur Verfügung.

2.1.3 Dokumentenzentrierte / artefaktbasierte Techniken

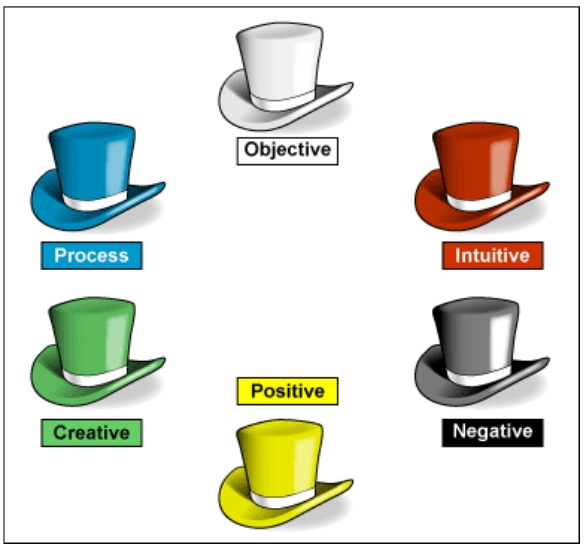
- **Systemarchäologie:** Die funktionalen Anforderungen werden aus Altsystem abgeleitet, beispielsweise durch Analyse von Nutzerdokumentation, Analyse von User Interfaces, Analyse von Sourcecode etc.
- **Perspektivenbasiertes Lesen**: Existierende Spezifikationsdokumente wird gezielt aus der Perspektive eines Stakeholders / potentiellen Nutzers ausgewertet, um funktionale Anforderungen zu extrahieren.

2.1.4 Kreativitätstechniken

- **Brainstorming**: In einer Gruppe von 5-10 Personen werden in einer vorgegebenen Zeit Ideen gesammelt. Jede Idee wird für alle sichtbar aufgeschrieben oder aufgehängt, ohne Kommentar des Erstellers oder der anderen Teilnehmer. Die Methode funktioniert sehr gut mit verschiedensten Stakeholdern. Es wird eine große Bandbreite an Ideen abgebildet. Es sollte aber darauf geachtet werden, dass die Stimmung positiv ist und keine frühzeitige Kritik entsteht.
- **Brainstorming Paradox**: Hierbei werden Ideen gesammelt, die **nicht erreicht** werden sollen. Das ist vor allem dann sinnvoll, wenn die Gruppe sich an einem „Knoten“ befindet und eine Stimmung der Ideenlosigkeit da ist, oder bei starkem Dissenz in der Gruppe. Durch die bewusst negative Perspektive wird eine Fokussierung auf die gewünschten Lösungen einfacher.
- **6-3-5 Methode**: 6 Personen erstellen zu Beginn jeweils 3 Ideen und schreiben sie auf einen Zettel. Die Zettel werden dann an die nächste Person (im Uhrzeigersinn) weitergegeben. Diese schreiben nun zu jeder Idee eine Erweiterung oder eine ergänzende, neue Idee auf. Das Ganze wiederholt man 5 mal, so dass jeder Teilnehmer jede Idee einmal betrachtet hat. Jede Runde hat einen vorgegebenen Zeitrahmen. Diese Methode funktioniert sehr gut bei einer schwierigen Gruppendynamik, da die Diskussion schriftlich erfolgt.

Die Methode produziert weniger Ideen als Brainstorming, dafür sind diese meist genauer. (Je nach Gruppengröße sind Anpassungen möglich, z.B. 4-2-3).

- **Walt-Disney-Methode**: Die Gruppe zieht nacheinander in drei verschiedene Räume, die jeweils für konkurrierende Sichten stehen: 1) Visionär, 2) Realist, 3) Kritiker. Dort sollen alle Teilnehmer*innen den „Geist" des Raumes annehmen. Die jeweiligen Räume sollten durch ihre Ausstattung die Sicht transportieren (z.B. Realist = nüchterne Einrichtung).
- **6-Hüte-Methode:** Mit dieser Methode werden verschiedene Sichten auf das Problem / Projekt erhoben. Jeder Stakeholder bekommt eine Sicht zugeordnet und muss diese einnehmen. Diese Sichten decken von "kritisch" über "analytisch" bis hin zu "positiv" verschiedene Meinungsbilder ab. Auf diese Methode müssen sich die Beteiligten einlassen, da diese ihre eigene Meinung in den Hintergrund stellen müssen.
  - **blau**: ordnendes, moderierendes Denken, Überblick, Prozesse, Big Picture („der blaue Himmel")
  - **weiß**: analytisches Denken, Konzentration auf Tatsachen und Erreichbares („das weiße Blatt")
  - **rot**: emotionales Denken, Empfinden, Konzentration auf Gefühle und Meinungen („Feuer und Wärme")
  - **schwarz**: kritisches Denken, Risikobetrachtung, Probleme, Skepsis, Kritik und Ängste („Schwarzmalerei", Advocatus Diaboli)
  - **gelb**: bedingungslos optimistisches Denken, Best-Case Szenario („Sonnenschein")
  - **grün**: kreatives, assoziatives Denken, neue Ideen („grüne Wiese")



2.1.5 Beobachtungstechniken

- **Feldbeobachtung**: Die Anforderungsermittler spielen „Fliege an der Wand" und beobachten / protokollieren die existierenden Arbeitsabläufe der potentiellen Nutzer. Durch eine Beobachtung der Nutzer kann deren Verhalten in dem jeweiligen Kontext besonders gut analysiert und verstanden werden. Zudem sind die Nutzer in der Regel hoch verfügbar, da diese ihren normalen Aufgaben nachgehen können. Des Weiteren fallen den Beobachtern unter Umständen Aspekte auf, an die die Nutzer gar nicht gedacht hatten, weil diese für sie offensichtlich waren.
  - Wie reagieren die Nutzer unter Zeitdruck?
  - Welchen Einfluss haben außenstehende Aspekte wie Licht, Geräusche, andere Nutzer?
  - Wie arbeiten die Nutzer, wenn sie abgelenkt sind?
- **Apprenticing**: Der Anforderungsermittler lässt sich über einen begrenzten Zeitraum hinweg für operative Tätigkeiten der potentiellen Nutzer „anlernen" (Apprentice = Auszubildende*r). Außerhalb des Anlernprozesses werden die Abläufe dann dokumentiert und analysiert.

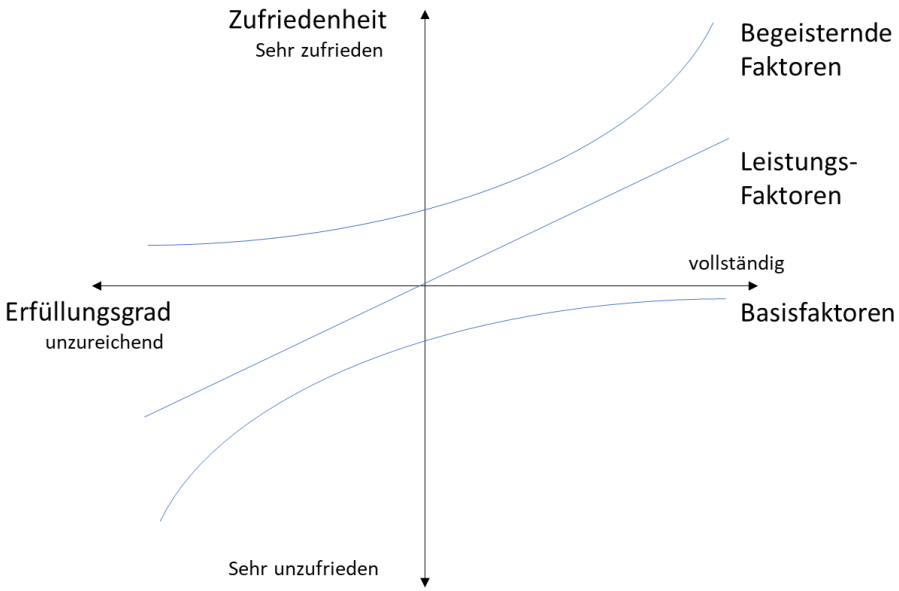
2.1.6 Weiterführende Literatur

- Bono, E. de. (1989).
- Gürtler, Meyer, 2013
- Pohl, 2008, S. 32-41
- Rupp, 2014, S. 80-106
- Schienmann, 2001, S. 203-213

2.2 Einordnung nach Kano-Methode

Das Kano-Modell hilft dabei, Anforderungen zu klassifizieren. Als Kriterium dient dabei die Zufriedenheit von Stakeholdern und potentiellen Nutzern, genauer: wie diese die jeweilige Eigenschaft wahrnehmen und bewerten würden.

Dafür werden die Anforderungen in drei Merkmalskategorien (**Basis**-, **Leistungs**- und **Begeisterungs**faktoren) eingeteilt. Diese Einteilung kann dann als Basis oder Input für eine Anforderungspriorisierung dienen.



Die Klassifikation nach Kano-Faktoren wird später wichtig, wenn man funktionale Anforderungen („1-Satz-Anforderungen“) aufstellt (siehe Kapitel 4). Man kann diese dann nach Kano gruppieren und später für die Priorisierung verwenden (siehe auch Kapitel XXX).

2.2.1 Glossar / Begriffsklärung

Begriff	Definition	Braucht man für …
Basisfaktor	Systemeigenschaft, die als selbstverständlich vorausgesetzt wird	Muss erfüllt sein, um Zufriedenheit zu garantieren. Wird aber häufig von Stakeholdern nicht explizit genannt, da diese das für selbstverständlich halten. Daher eine potentielle Missverständnis- und Konfliktquelle während der Implementierung.
Leistungsfaktor	Bewusst und explizit gefordertes Merkmal des Systems	Leistungsfaktoren bilden i.a. den Hauptteil eines Lastenheftes (einer Anforderungsspezifikation).
Begeisterungsfaktor	Systemeigenschaft, die nicht erwartet wird, aber als angenehm und nützlich empfunden wird, wenn sie vorhanden ist.	Steigert die Akzeptanz der Stakeholder für das System, da sie unerwartete und als positiv empfundene Eigenschaften vorfinden.

2.2.2 Welche Ermittlungstechniken (siehe Kap. 2.1 auf S. 8) liefern welche Kano-Faktoren?

Basisfaktoren	Leistungsfaktoren	Begeisterungsfaktoren
Beobachtungstechniken • Feldbeobachtung • Apprenticing **Dokumentenzentrierte / artefakt-basierte Techniken** • Systemarchäologie • Perspektivbezogenes Lernen	**Befragungstechniken** • Selbstaufschreibung seitens des Stakeholders • On-Site-Customer • Interviews • Workshops *(aber auch alle anderen Techniken)*	**Kreativitätstechniken** • 6-Hüte-Methode • 6-3-5 Methode • Walt Disney Methode • …

2.2.3 Hilfe für die Kano-Klassifikation

Obwohl die Definition der Kano-Faktoren intuitiv verständlich ist, bereitet die Zuordnung einer konkreten Anforderung zu einem Faktor oft Probleme. Hier hilft es, dem Stakeholder (oder sich selbst, als Gedankenexperiment) zwei Fragen zu stellen und die Antworten entsprechend der nachfolgenden Matrix einzuordnen.

Dysfunktionale Frage: Was wäre, wenn das Merkmal fehlt?

Funktionale Frage: Was wäre, wenn das Merkmal vorhanden ist?

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 Weiterführende Literatur

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

2.3 Personas

Personas sind fiktive Personen, die in einer Zielgruppe die **typischen Anwender** repräsentieren. Die Persona sollte die **wichtigen Eigenschaften** der Zielgruppe abbilden. Personas werden in narrativer Form erstellt und basieren auf echten Nutzern Stakeholdern.

2.3.1 Glossar / Begriffsklärung

Begriff	Definition	Braucht man für …
Persona	Textbasierte Archetypen von realen Benutzern, stellt Merkmale eines Stakeholders dar.	Ermittlung von möglichen Funktionalitäten für das zu erstellende System. Hineinversetzen in den Anwender.
primäre Persona	Persona im Fokus des Systemdesigns.	Priorisierung durch Identifikation der „wichtigen“ Stakeholder, die in die Persona einfließen. Gegenteil sind *sekundäre* Personas.

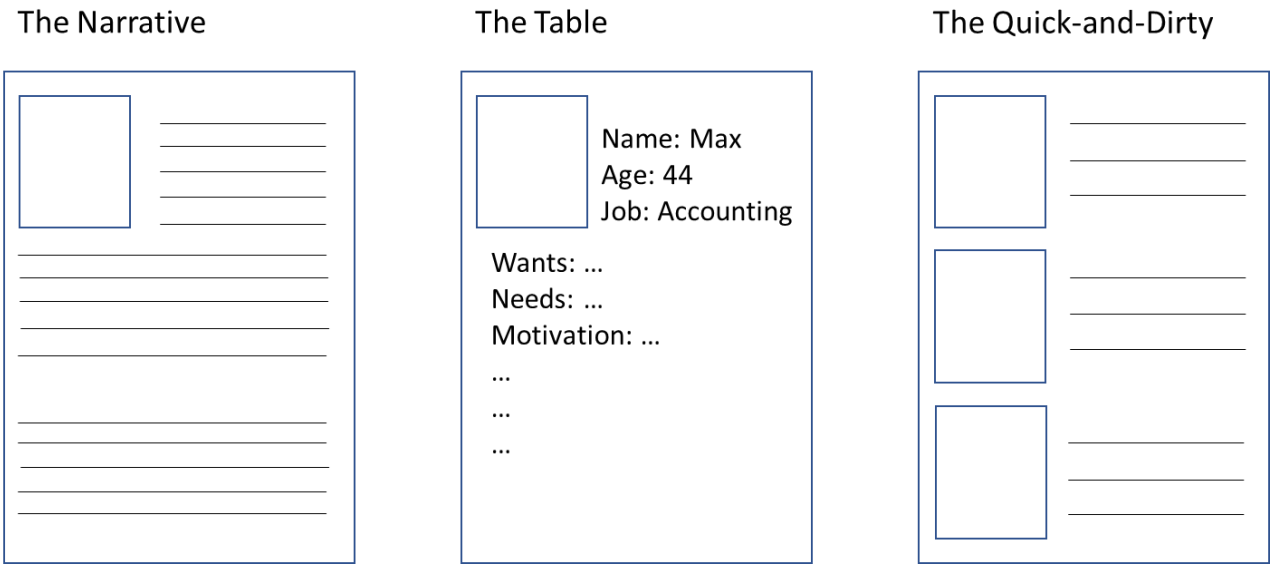
2.3.2 Templates: Beschreibung von Personas

Checkliste:

- Foto
- Name, Alter
- Bildung
- Job
- Familie
- Behinderungen
- Technologie
- Motivation, Ziele, Bedürfnisse
- Erwartungen
- Verhaltensweise
- Arbeitsumgebung
- Fähigkeiten

Max. 1 Seite, relevante Eigenschaften sind je nach Kontext verschieden.

Bei Personas besteht die Gefahr, dass Ersteller zu stark von den eigenen Vorstellungen geleitet werden. Es ist also wichtig, dass der Designer bzw. Entwickler erkennt, dass er nicht für sich selbst entwickelt.



2.3.3 Wie erstelle ich Personas?

Je nach Kontext und Aufgabenstellung bieten sich verschiedene Möglichkeiten an. Teilweise ist eine geschickte Kombination die ertragreichste Möglichkeit. So bietet sich bei demografischen Fragen in den meisten Fällen eine Marktrecherche in Kombination mit einer Online-Umfrage an.

Geeignete Techniken sind Befragungs- und Beobachtungstechniken (s. auch die Beschreibung der Erhebungstechniken in Kap. 2.1 auf S. 8):

- Interview
  - Interviews eignen sich, um Anforderungen, Verhalten und Bedürfnisse der Nutzer und Einflüsse der Umgebung besser zu verstehen
  - Dauern in der Regel 30-60 min
  - Interview, wenn möglich, in der Arbeitsumgebung der Person halten, um einen Einblick in die Aktivitäten des Nutzers durch Beobachtung zu bekommen
  - Respektieren Sie die Meinung bzw. das Wissen des Interviewpartners!
  - W-Fragen: **W**er, **W**o, **W**as, **W**ann, **W**ie, **W**arum, **W**oher?
- Umfrage
- Feldbeobachtung der Nutzer (im realen Nutzungskontext)
- Marktrecherche
  - Durch eine Marktrecherche können viele Parameter bereits im Vorfeld mit einem geringen Aufwand ermittelt werden. Durch eine Kombination mit einer weiteren Ermittlungstechnik können aus beiden Vorteile gezogen werden.
  - Eine reine Marktrecherche zur Erstellung nur dann Sinn, wenn die Personas bei-zeiten durch fundierte projektbezogene Daten angereichert werden.

2.3.4 Weiterführende Literatur

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

2.4 Szenarien

Szenarien beschreiben Situationen, in denen Menschen mit einem System interagieren (Handlungsstrang). Diese Situationen haben einen klar definierten Kontext. Szenarien dienen einerseits zur informellen Erfassung von Anforderungen an ein System, und andererseits zur Evaluation eines Systems bzgl. formalisierter Ziele und Anforderungen.

Szenarien orientieren sich an den zuvor erstellten Personas (siehe Kap. 0) und können sich auf die aktuelle Situation beziehen (IST-Szenario) oder auf die zukünftige (verbesserte) Situation (SOLL-Szenario). Szenarien fokussieren sich dabei auf die Aktivitäten der Menschen und weniger auf konkrete Interaktionen. Zudem beinhalten diese die Nutzungsumgebung der Anwendung oder des Systems, die Nutzungsdauer, die Unterbrechungswahrscheinlichkeit und die Nutzung anderer Anwendungen.

2.4.1 Glossar / Begriffsklärung

Begriff	Definition	Braucht man für …
Szenario	Narrative, informelle Beschreibung einer Situationen, in denen Personas im dem System im bestimmten Nutzungskontext interagieren	Erfassung typischer Nutzungsabläufe (z.B. als Ergebnis von Erhebungstechniken aus Kap. 2.1), ohne sich zunächst durch zu viele Formalismen einzuengen
Nutzungskontext	Handlungsraum und Umgebung des Nutzers (Aufgaben, physische und soziale Umgebung, Hard- und Software, Verbrauchsmittel, etc.)	Reduktion auf die wichtigsten Details der Interaktion zwischen Nutzer und System

2.4.2 Weiterführende Literatur

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

2.4.3 Typen von Szenarien

Templates (Schablonen) sind für Szenarien wenig sinnvoll, da diese keinen formalen Rahmen (bestimmte Attribute) haben. Man die verschiedenen Szenarien aber nach ihrem Typ unterscheiden, wie nachfolgend dargestellt. Nicht alle Szenarientypen sind gleich wichtig (werden gleich häufig verwendet).

Kategorisierung	Typ	Kurzbeschreibung	Verwendungshäufigkeit für SW-Anforderungen	wird verwendet für	Beispiel
Normal vs. Ausnahme	Hauptszenario (Standardszenario)	Normaler Weg zur Erfüllung des Ziels	immer	Standard für Use Cases**,** eher als Typ- denn als Instanzszenario. Kann sowohl **Interaktions**- (häufigerer Fall) wie auch **Systemszenario** sein.	*Siehe Beispiel zu Kap. 7 (Use Cases) auf S. 33*
	Alternativszenario	Alternativer Weg zur Erfüllung des Ziels	meistens	Standard für Use Cases	
	Ausnahmeszenario	Ziel wird nicht erfüllt	meistens	Standard für Use Cases	
Positiv vs. Negativ	positives Szenario	Interaktionsfolge, die zur Erfüllung eines Ziels führt	immer	Sehr ähnlich dem **Hauptszenario**	*Siehe Beispiel zu Kap. 7 (Use Cases) auf S. 33*
	negatives Szenario	Interaktionsfolge, die zur Nichterfüllung eines Ziels führt	meistens	Spielart des **Ausnahmeszenarios**	
	Missbrauchsszenario	Beschreibt eine unerwünschte Verwendung des Systems	manchmal	Spielart des **Ausnahmeszenarios**	
Konkret vs. Abstrakt	Instanzszenario	Konkrete Interaktionen mit konkreten Ein- und Ausgaben zwischen konkreten Personen und/oder Systemen	immer	Sehr konkrete Spielart eines Szenarios. Kann gut zusammen mit detaillierter Persona verwendet werden, also **früh** im Anforderungsermittlungs-Prozess.	Karl möchte zum Potsdamer Platz 1 in Berlin fahren. Karl benutzt das Navigationssystem seines VW Golf mit dem Kennzeichen „E-IS-12“. Karl selektiert „Zielort eingeben“ im Hauptmenü, gibt „Potsdamer Platz 1 in Berlin“ als Zielort ein und betätigt die Taste „Wegstrecke ermitteln“... (Pohl 2008)
	Typszenario	Interaktionen zwischen Typen von Akteuren durch Typen von Ein- und Ausgaben	immer	Ist eher die Abstraktionsebene, die in Use Cases Verwendung findet (also etwas **später** im Anforderungsermittlungs-Prozess)	Der Fahrer gibt die Zieladresse in sein Navigationssystem ein. Das System gibt die Rückmeldung, dass die Routenberechnung vorgenommen wurde.

Kategorisierung	Typ	Kurzbeschreibung	Verwendungshäufigkeit für SW-Anforderungen	wird verwendet für	Beispiel
Interaktion vs. System	Systeminterne Szenarien (Typ A)	Systeminterne Interaktionen in Form von Interaktionsfolgen zwischen Systembestandteilen	meistens	Ist immer dann sinnvoll, wenn ein sehr technisches System zu beschreiben ist. Entspricht einer **White-Box-Sicht** auf das System.	Die Komponente „Navigationssteuerung“ fragt von der Komponente „GPS Lokation bestimmen“ die GPS-Koordinaten an. Die Komponente „GPS Lokation bestimmen“ übermittelt die Koordinaten. Die Komponente „Navigationssteuerung“ ruft die Komponente „Bildschirmausgabe“ mit der aktuellen Position sowie dem Zielort auf. Die Komponente „Bildschirmeingabe“ gibt die Routenparameter an die Komponente „Navigationssteuerung“ weiter, die damit die endgültige Route bestimmt. (Pohl 2008).
	Interaktionsszenarien (Typ B)	Interaktionen zwischen System und Systemnutzern (Stakeholdern und Systemen im Kontext)	immer	**Black-Box-Sicht** auf das zu beschreibende System. Der "Standardfall" eines Szenarios.	
	Kontextszenarien (Typ C)	Erweiterung von Typ B mit zusätzlichen Interaktionen und Informationen im Kontext	meistens	Wenn Kontextinformationen verfügbar, dann sinnvoll, sonst eher Typ B.	Der Fahrer möchte einen Zielort erreichen, der sich außerhalb des Kartenmaterials befindet, das im Navigationssystem (System) gespeichert ist. Da das Navigationssystem den Fahrer ohne das Kartenmaterial nicht führen kann, entscheidet sich der Fahrer, eine Routenberechnung über seinen Mobilfunkbetreiber durchführen zu lassen. […] Der Fahrer gibt in dem Benutzerdialog am Mobiltelefon Start- und Zielort sowie die Routenparameter (schnellster Weg) ein […]. (Pohl 2008)

Kategorisierung	Typ	Kurzbeschreibung	Verwendungshäufigkeit für SW-Anforderungen	wird verwendet für	Beispiel
Beschreibend vs. Erklärend	erklärendes Szenario	Begründung und Erklärung von Interaktionen	manchmal	Mischt **Nutzen** und **Ziele** in das Szenario. Erklärungen können auch sinnvoll in Use Cases gemischt werden, wenn der Erklärungsanteil nicht überhand nimmt.	Der Abstand zwischen den Fahrzeugen verringert sich weiter. Da das Fahrzeug mit hoher Geschwindigkeit (> 90 km/h!) über die Autobahn fährt und daher vor einem evtl. Ausweichmanöver die Geschwindigkeit reduziert werden sollte, leitet der On-Board-Computer, um einen Auffahrunfall zu vermeiden, eine automatische Notbremsung ein. (Pohl 2008)
	deskriptives Szenario	beschreibende Darstellung der Interaktionen	immer	Der Standardfall von Szenarien	
	exploratives Szenario	Alternative Lösungs- und Bedienungsmöglichkeiten	manchmal	Hat starken "Brainstorming-Charakter", passt eher in frühe Phase der Anforderungsermittlung	Karl möchte mit seinem PKW mittels Navigationssystem zu einem Zielort fahren. Es stellt sich zunächst die Frage, ob der Startpunkt der Fahrt immer die aktuelle Position des Fahrzeugs ist, oder ob Karl den Startpunkt selbst auswählt. Die automatische Wahl des Startpunkts vermeidet eine zusätzliche Benutzerinteraktion und unterstützt somit ein schnelles Navigieren. Die Eingabe des Startpunkts würde es ermöglichen, auch Strecken zu berechnen, die als Startpunkt nicht den aktuellen Standpunkt des Fahrzeugs besitzen. [...] (Pohl 2008)

3 Fachliches Glossar / Fachliches Datenmodell

Ziel ist das Bilden eines gemeinsamen Wortschatzes zwischen allen Projektbeteiligten, sowie das Erstellen eines einheitlichen Datenmodells, das die abzubildende Anwendungsdomäne repräsentiert. Dies erleichtert die Kommunikation mit den Stakeholdern und beugt Missverständnisse aufgrund fehlender Definitionen vor.

3.1 Glossar / Begriffsklärung

Begriff	Definition	Braucht man für …
Fachliches Glossar	Gemeinsamer Wortschatz, der zwischen beteiligten eines Projektes definiert wird.	die Vermeidung von Konflikten, die durch widersprüchliche Interpretationen entstehen können.
Geschäftsobjekt	Ein Element, das einem Projekt einen fachlichen Wert bietet.	die Feststellung der grundlegenden Elemente, die in einer Anwendungsdomäne existieren
Fachliches Datenmodell *(auch: Domänenmodell)*	Beschreibt ein implementierungsunabhängiges Modell, welches Gegenstände des realen Projektkontextes widerspiegelt (üblicherweise in Form eines UML-Klassendiagramms)	interdisziplinäre Kommunikation zwischen allen Projektbeteiligten

3.2 Vorgehen

Das fachliche Glossar entsteht und wird erweitert durch Gespräche mit Stakeholdern (siehe Kap. 1.3 auf S. 5), Fachexperten der jeweiligen Domäne und durch Recherche in anderen Quellen wie etwa interne Dokumentation und Fachliteratur.

Achtung! Ein häufiger Fehler ist das Treffen von Annahmen ohne Rücksprache mit den Stakeholdern, da von einer ausreichenden Kenntnis der Fachdomäne ausgegangen wird.

Daumenregel: Wer die Domäne **nicht kennt**, **unterschätzt** ihre Komplexität.

3.2.1 Methode: Hauptwortanalyse zur Befüllung des Glossars aus fachlichen Texten

1. alle Hauptwörter in den Anforderungen markieren.
2. Bereinigen der erstellten Rohliste
   a. Wiederholungen streichen.
   b. Synonyme entfernen (sicherstellen, dass es wirklich Synonyme sind!)
   c. Substantivierte Verben streichen (Vorsicht bei Wörtern wie „Bestellung“ – das ist ein Geschäftsobjekt!)
   d. Substantive die nur Synonyme für Konjunktionen oder Verben bilden streichen.
   e. Alle Sätze streichen, die keine Geschäftsobjekte beschreiben.
   f. Das zu modellierende System bildet kein Geschäftsobjekt.
   g. Informationen zur späteren Umsetzung auf technischer Ebene streichen.
   h. Kennzeichnungen für z.B. Optionalität streichen.
3. jedes markierte Hauptwort bildet ein Geschäftsobjekt.
4. die ermittelten Geschäftsobjekte in Form eines Glossars dokumentieren

- Erstellen des fachlichen Datenmodells aus einem Glossar

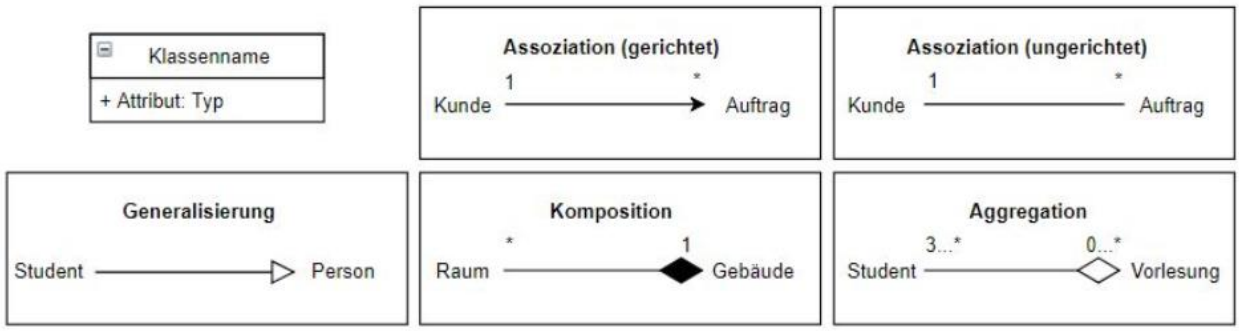
3.3 Fachliches Datenmodell / Domänenmodell

Auf Basis der ermittelten Geschäftsobjekte kann ein fachliches Datenmodell (Domänenmodell) der Domäne erstellt werden.

Zusätzlich zum Glossar bildet das fachliche Datenmodell ab, wie ein Begriff / Geschäftsobjekt **in Beziehung zu anderen** Begriffen steht. Die Umsetzung als einfaches UML-Klassendiagramm ist auch für IT-Laien gut verständlich!

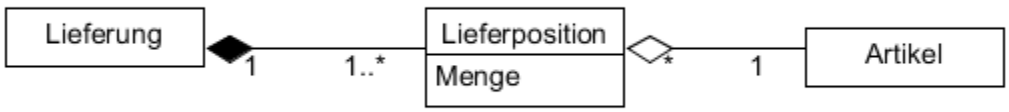
3.3.1 Methode: Fachliches Datenmodell als einfaches UML-Klassendiagramm

Regel: Nur die folgenden Beziehungen, keine Methoden in den Klassen, wenig bis keine Attribute.



3.3.2 Beispiel

Begriff	Definition
Lieferung	Einzulagernde Lieferung eines Lieferanten, bestehend aus einzelnen Lieferpositionen
Artikel	Beschreibung eines gelieferten Objekts
Lieferposition	Bestandteil einer Lieferung, bestehend aus Artikel und Menge



3.4 Weiterführende Literatur

- Ebert, 2014, S.204-206
- Evans, 2003, S25-55
- Pohl und Rupp, 2011, S. 85-87; 95-99
- Rupp 2014, S. 207-208; 222-224

4 Funktionale Anforderungen

Funktionale Anforderungen beschreiben die gewünschte Funktionalität eines Systems, dessen Verhalten und dessen Daten. Diese können oft unterschiedlich interpretiert werden. Um diesem Problem entgegenzuwirken, sollten potentielle sprachliche Unklarheiten eliminiert und Anforderungen mittels Satzschablonen eindeutig formuliert werden.

4.1 Glossar / Begriffsklärung

Begriff	Definition	Braucht man für …
Funktionale Anforderung	Beschreibt gewünschte Funktionalität eines Systems, dessen Daten und / oder Verhalten	Kondensiert die Anforderungen an ein IT-System in eine Menge von kompakten (1 Satz) Forderungen, die man gut priorisieren und in eine Projektplanung überführen kann.
Satzschablonen	Bauplan für die syntaktische Struktur einer einzelnen Anforderung	Hilft dabei, funktionale Anforderungen ohne Mehrdeutigkeiten, Unklarheiten und Widersprüche zu formulieren.

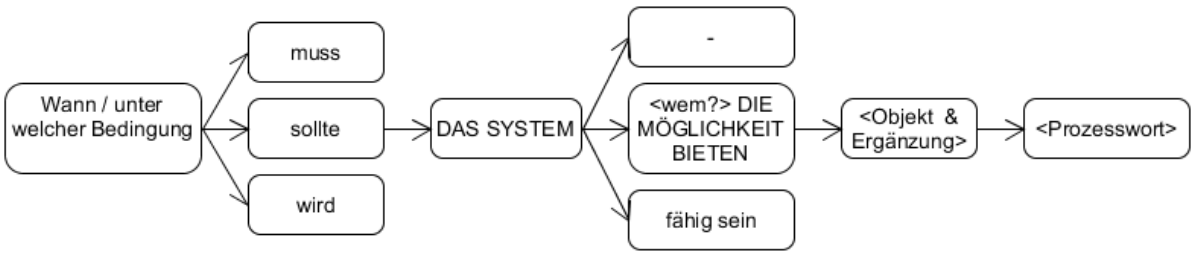
4.2 Prüfung und Bereinigung von natürlichsprachlichen Anforderungen

Zunächst sollten alle potentiellen Unklarheiten / Mehrdeutigkeiten eliminiert werden. Anschließend wird mittels einer Satzschablone die Anforderung nach einem einheitlichen Schema formalisiert.

Prüfung	Worum geht es?	Typischerweise	Beispiel für mögliche Probleme
1) Nominalisierung	Ein (oft lange währender) Prozess wird zu einem (einmaligen) Ereignis gemacht. Dadurch wird ein Vorgang zu einem Ereignis, und viele vorgangsrelevante Informationen gehen verloren.	• die Integration • die Prüfung • die Abnahme • die Übergabe • die Anzeige • die Benutzerführung • die Bestätigung	„Das System ermöglicht **Protokollierung** von Anfragen. • wer protokolliert? • wann wird protokolliert? • was wird protokolliert? • zu welchem Ziel? • unter Einhaltung welcher Regeln?
2) Universalquantoren	Universalquantoren = Angaben über Häufigkeiten. • fassen Menge von Objekten zu einer Gruppe zusammen • treffen Aussage über deren Verhalten	• nie • immer • kein • jeder • alle • nichts • implizite Allquantoren	„Der Benutzer erhält eine statistische Auswertung von allen Anfragedaten.“ • Jeder Benutzer (impliziter Universalquantor)? • Alle Anfragedaten? • Immer (implizit!)?

Prüfung	Worum geht es?	Typischerweise	Beispiel für mögliche Probleme
3) Häufig gebrauchte generische Hauptworte	Häufig gebrauchte Substantive, deren Bedeutung „überladen“ ist. Weitere Informationen sind erforderlich, um sie eindeutig zu spezifizieren.	• der Anwender • das System • die Meldung • die Daten • die Funktion	„Bei Formularanfragen sind die Feldkonventionen durch Plausibilitäten im Formular sicher zu stellen.“ • Welches Formular? • Welche Feldkonventionen? • Welche Plausibilitäten?
4) Unvollständig spezifizierte Prozesswörter	Manche Prozesswörter (Verben) erfordern mehr Informationen, um vollständig spezifiziert zu sein.	Alles, was nach W-Fragen nicht standhält: Wer? Was? Wozu? Warum? Wann? Wo? Wie? Wieviel?	„Der Benutzer gibt die Login-Daten ein“ • **Wo** und **wann** erfolgt die Eingabe?
5) Unvollständig spezifizierte Bedingungen	Die Anforderung enthält eine Bedingung, die nicht vollständig spezifiziert ist.	Signalwörter: • wenn … dann • falls • im Falle von • abhängig von	„Falls dem Benutzer eine Sperre für den ausgewählten Datensatz angezeigt wird …“ • Was, wenn keine Sperre? • Was, wenn alle Datensätze gesperrt?

4.3 Satzschablone für funktionale Anforderungen



- „muss“: gehört unbedingt zu diesem Release
- „sollte“: optional
- „wird“: „muss“ für ein zukünftiges Release, nicht jetzt

Für diese Einordnung kann das Kano-Modell zur Hilfe genommen werden (siehe Kapitel 2.2). Dabei gilt:

- Basisfaktoren sind “muss”-Anforderungen
- Leistungsfaktoren können “muss”-, „sollte“ oder „wird“ sein
- Begeisterungsfaktoren ebenfalls, aber aus taktischen Gründen sollten einige Begeisterungsfaktoren bei „muss“ vertreten sein

4.4 Beispiel

Beispiel: aus folgendem Ausgangssatz

> *Die Gründe für die Anfrage sind bei der Protokollierung der Anfragen zu speichern.*

wird unter Anwendung der Bereinigungsregeln und Verwendung der Satzschablone:

> *Im ersten Verarbeitungsschritt | muss | das System | fähig sein, |*
> *jede eingehende Anfrage mit den vollständigen Daten der Anfrage sowie dem Zeitstempel des Eingangs | zu protokollieren.*

4.5 Weiterführende Literatur

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

5 Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen (NFA) umfassen alle Anforderungen, die nicht den funktionalen Anforderungen zugeordnet werden können.

Sie beschreiben wichtige Qualitätseigenschaften und Randbedingungen des zu entwickelnden Systems.

5.1 Glossar / Begriffsklärung

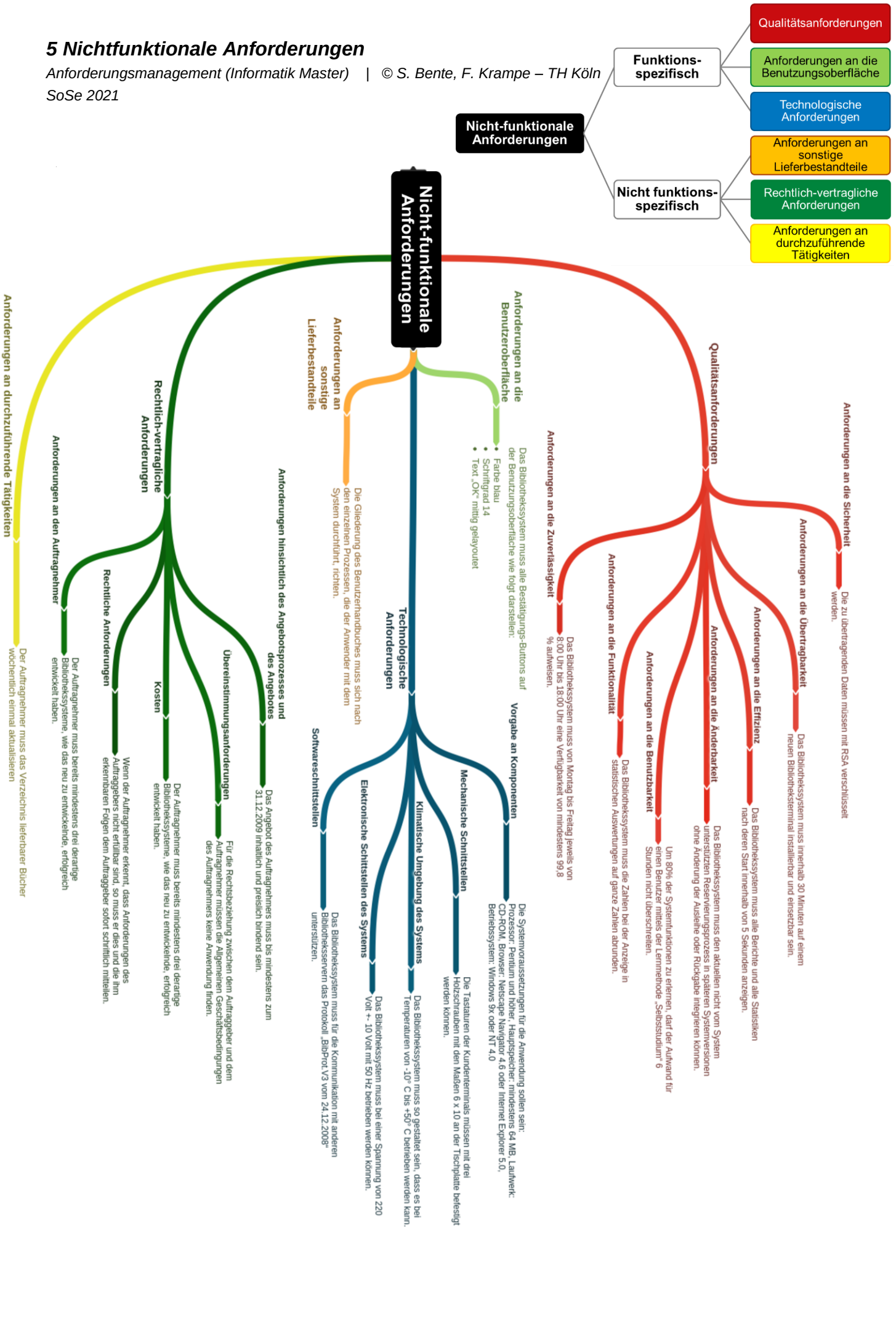
Begriff	Definition	Braucht man für …
Nicht funktionsspezifische Anforderungen	Betreffen das „Wie“ und nicht das „Was“ der Realisierung	Operative Gestaltung der Leistungserbringung
Qualitätsanforderungen	Legen fest, in welcher „Güte“ die funktionalen Anforderungen zu erfüllen sind	Insbesondere zur Beschreibung von Lastverhalten und Zuverlässigkeit / Verfügbarkeit: **Performanz** • Antwortzeit (min, max, avg) für eine Transaktion • Durchsatz • Kapazität (Mengengerüst) **Zuverlässigkeit** • Verfügbarkeit (%) • Mean time between failure / Mean time to repair / **Änderbarkeit** **Sicherheit** **Interoperabilität** / Konformität
Technologische Anforderungen	Geben Lösungsvorgaben, beschreiben Umgebung, in der das System betrieben werden soll	Stellen die Konformität der zu schaffenden Lösung zur Betriebsumgebung und der strategischen Zielrichtung der Organisation / des Unternehmens sicher.

5.1 Weiterführende Literatur

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

5.2 Klassifizierung und Beispiele

In der umseitigen Abbildung finden Sie die verschiedenen Arten nicht-funktionaler Anforderungen, jeweils mit kurzem Beispiel.


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 Priorisierung und Konfliktlösung

Priorisierung ist notwendig, um Anforderungen voneinander zu trennen (wichtig und unwichtiger), sowie Ordnung in ein Projekt zu bringen und den Projekterfolg sicherzustellen. Diese können je nach Projekt durch Ad-hoc-, analytische und agile Techniken verwirklicht werden.

Mögliche Priorisierungskriterien umfassen:

- Geschäftlicher oder strategischer Wert
- Kosten der Umsetzung
- Zusammengehörigkeit / Abhängigkeit zwischen Anforderungen
- Risiken / Kosten bei Entwicklungsfehlschlag

6.1 Glossar / Begriffsklärung

Begriff	Definition	Braucht man für …
Ad-Hoc-Priorisierung	Einfache und schnelle Priorisierung	Große Mengen von Anforderungen, Vorpriorisierung
Analytische Priorisierung	Priorisierungsmethoden auf mathematisch-analytischer Basis	Komplexe Kriteriensysteme zur Priorisierung, fundierte Entscheidungsprozesse
Agile Priorisierung	Techniken basierend auf Teamprozessen	Agile Entwicklung in einem eingeschwungenen Team

6.2 Ansätze für Priorisierung

6.2.1 Ad-Hoc-Priorisierung

- **Ein-Kriteriums-Klassifikation.** Eine in der Praxis häufig verwendete Priorisierungstechnik mit drei Prioritätsklassen
  - *Mandatory:* Dringende Anforderungen
  - *Optional:* Keine dringenden Anforderungen
  - *Nice-to-have:* Zusätzliche Anforderungen, die bei Nichtberücksichtigung den Erfolg des Systems nicht gefährden
- **Ranking.** Ausgewählte Stakeholder bestimmen eine Rangfolge von Anforderungen im Hinblick auf ein bestimmtes Kriterium. Lässt sich gut z.B. im Workshop mit Klebepunkten machen.
- **Top-Ten-Technik.** Wie Ranking, jedoch beschränkt auf die Auswahl der <n> wichtigsten Anforderungen.
- **Kano-Klassifikation.** Siehe Erklärung in vorigem Kapitel 6.2.2.

6.2.2 Priorisierung nach Kano-Methode

Das Kano-Modell wurde in Kapitel 2.2 vorgestellt. Es kann als Hilfe bei der Priorisierung eingesetzt werden – siehe hierzu auch Kapitel 4.3:

- Basisfaktoren sind “muss”-Anforderungen
- Leistungsfaktoren können “muss”-, „sollte“ oder „wird“ sein
- Begeisterungsfaktoren ebenfalls, aber aus taktischen Gründen sollten einige Begeisterungsfaktoren bei „muss“ vertreten sein

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

Man sammelt Bewertungskriterien, bewertet die Auswirkung der Anforderung mit Punkten zwischen 1 und 6 (CAF) und legt außerdem fest, ob sie positiv (+), negativ (-) oder neutral (|) zur Bewertung beitragen (PMI). Daraus ergibt sich für jede Anforderung ein Punktwert.

Beispiel: Ich möchte mir ein neues Fahrad kaufen. Meine Ziele sind in der ersten Spalte gelistet.

		Anforderung 1 **„Das Fahrrad soll deutlich unter 2000,-- EUR kosten.“**	*Anforderung 2* **„…“**
Ziele	**Ziel-Priorität (1..6)**	**PMI**	**PMI**
Ich will mich wegen Anschaffungen nicht finanziell einschränken oder verschulden.	3	+	…
Ich will viel Bewegung an der frischen Luft und dadurch möglichst fit werden.	5	|	…
Ich möchte vor meinen Freunden mit den Dingen angeben können, die ich kaufe.	1	-	…
Ich möchte lieber wenige wirklich gute Dinge besitzen, als häufig "Wegwerfprodukte" zu kaufen.	4	-	…
Summe		3 + (0*5) -1 -4 = **-2**	…

- Template für CAF+PMI hier.
- Komplettes Fahrrad-Beispiel hier.

6.2.4 *Analytische Priorisierung*

- **Analytical Hierarchy Process (AHP)**. Siehe frei verfügbares Template von SCB Associates: https://www.scbuk.com/AHP%20Template%20SCBUK.xls.
    1. Anforderungen in linke Spalte eintragen (bis 15 in Template)
    2. Anzahl der Anforderungen festlegen
    3. Paarweise Gewichtung in Matrix eintragen (gelbe Felder)
    4. Ggfs. Konsistenzcheck (ganz rechts) beachten
    5. Priorisierung aus grüner Tabelle ganz rechts ablesen

6.2.5 *Agile Priorisierung*

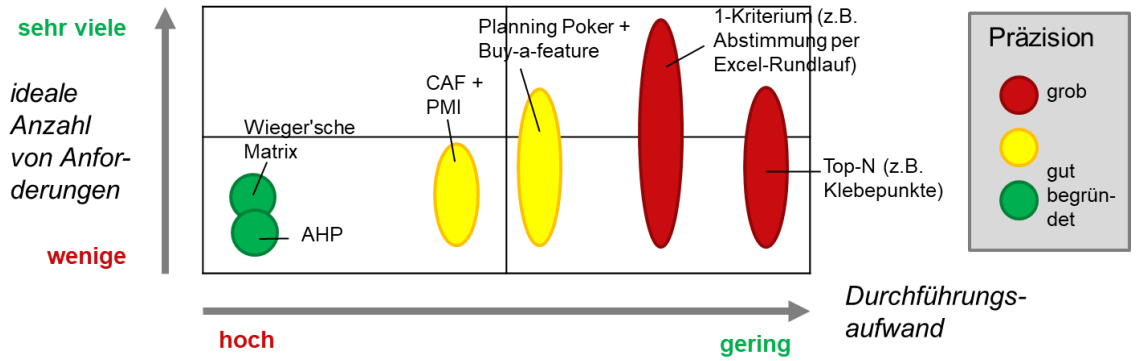
- **Buy a Feature:** Passt gut für agile Projekte, z.B. zur Auswahl von Features für das nächste Release oder den nächsten Sprint. Jede Anforderung erhält einen Preis, abhängig von Entwicklungskosten / Geschäftswert / Risiko / … Dies kann z.B. mit Planning Poker festgelegt werden. Die Teilnehmer erhalten Spielgeld und „kaufen“ Anforderungen.

Sie müssen dabei Allianzen eingehen. So entsteht eine geordnete Top-N-Auswahl.
Regeln:

1. Keine Anforderung sollte einem Teilnehmer alleine gekauft werden können
2. alle Teilnehmer zusammen sollten nicht alle Anforderungen kaufen können
3. Teilnehmer diskutieren, um Geld für ein bestimmtes Feature zusammen zu legen
4. Diskussion beobachten und Kaufreihenfolge notieren!
5. **Beispiel**: Team von 6, jeder Teilnehmer 100 € (gesamt: 600 €), Features kosten 120 €, 130 €, 150 €, 150 €, 200 €, 200 €, 350 € (ges. 1300 €)

6.2.6 Wann sollte man welche Technik einsetzen?

Ad-hoc-Techniken bilden ein gutes Grundgerüst für schnelle Priosierung. Eine hohe Anzahl von Anforderungen kann man so zuerst vorfiltern und dann mit anderen Methoden weiter priorisieren.



6.2.7 Weiterführende Literatur

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

6.3 Konfliktmanagement

Bei einer hohen Anzahl von Anforderungen, vielen Stakeholdern und einem politisch aufgeladenen Umfeld kann es zu Konflikten bei der Priorisierung kommen. Oft gibt es unterliegende, nicht unbedingt sachlich begründete Konflikte, die nicht sofort erkennbar sind. Konfliktmanagement hilft einen solchen Konflikt zu identifizieren, zu analysieren und mit geeigneten Methoden aufzulösen.

6.3.1 Glossar / Begriffsklärung

bis hier kann ein Anforderungsmanager agieren

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

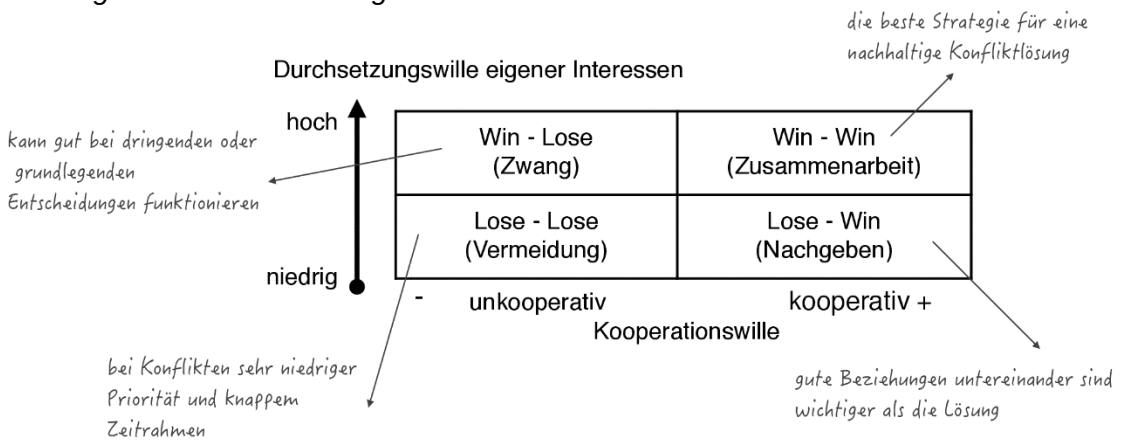
verstecken sich häufig hinter anderen Konfliktarten, z.B. Sach- oder Interessenkonflikten

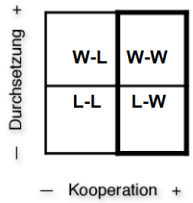
ab hier ausgebildeter Coach nötig

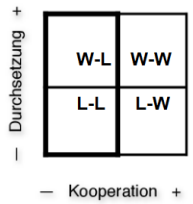
Konfliktart	Mögliche Kennzeichen	Ursachen
Sachkonflikt	Einigkeit übers Ziel, Uneinigkeit über das Mittel/Vorgehen; Debatte	Mangel an Informationen; Unvollständige Fakten; Unterschiede in Interpretationen
Benennungs-konflikt	Verwendung von Synonymen und Homonymen	
Interessenkonflikt	Mangelnde Argumentation 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?

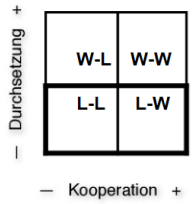
um die persönlichen Konflikte lösen zu können, muss der enthaltene Sachkonflikt isoliert werden

6.3.2 Strategien zur Konfliktlösung









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 Weiterführende Literatur

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

7 Use Cases

Use Cases sind einfach zu schreiben und zu lesen. Sie sind sehr hilfreich, um die funktionalen Anforderungen an das System soweit auszudetaillieren, dass zu von einem Entwicklungsteam umgesetzt werden können. Sie eignen sich auch sehr gut zur Kommunikation, wenn sich Anforderungs- und Entwicklungsteam nicht kennen (Nearshore- / Offshore-Entwicklung).

7.1 Glossar / Begriffsklärung

Begriff	Definition	Braucht man für …
Use Case	Durch das (IT-)System zu unterstützendes Aufgabenfeld, aus Sicht von einem oder mehreren Nutzern (Aktoren) formuliert und ausgelöst durch einen Aktor.	Detaillierte Anforderungsdefinition, auf deren Basis der Architekturentwurf und die Implementierung erfolgen kann.

7.2 Template für einen einzelnen Use Case

Es gibt in der Literatur viele Vorschläge für Use-Case-Templates; dieses ist ein minimales, das sich in der Praxis gut bewährt hat. Erweitern Sie es wo nötig, aber Sie sollten keine Felder weglassen.

ID	Eindeutiger Identifier des Use Cases
Name	Name des Use Case (i.a. Objekt + Verb)
Beschreibung	Kurze Beschreibung des Use Cases
Auslösender Aktor	Akteur, der den Use Case beginnt; kann auch ein Umsystem sein
Weitere Aktoren	Weitere beteiligte Akteure, können mögliche Umsysteme sein
Auslöser	Ereignis(se), das den Use Case auslöst
Vorbedingung	Bedingung um den Use Case starten zu können
Nachbedingung	Zustand nach erfolgreichem Durchführen des Use Cases
Hauptszenario	Abfolge von Schritten, die den Ablauf des Use Cases beschreibt und zum Erfolg führt. **(siehe auch Definition „Haupt-/Alternativ-/Ausnahmeszenario in Kap. 2.4.3 auf S. 16)**
Alternativszenario	Vom Hauptszenario abweichende Schrittfolge, die aber dennoch zum Erfolg führt; beschreibt eine Ausnahmesituation
Ausnahmeszenario	„Misserfolgs“-Szenario, das nicht zum Erfolg führt (dann mit eigener Nachbedingung)

7.2.1 Nummerierung der Szenarioschritte

- Schritte im **Hauptszenario** werden ab 1 durchnummeriert. *Beispiel:* 1, 2, 3, 4, …
- **Alternativen** / **Ausnahmen** zu einem bestimmten Schritt: entsprechende Nummer und „a, b, …“. *Beispiel:* Alternative zu Schritt 2: „2a“
- Wenn zu einem Hauptszenario-Schritt mehrere Alternativ- / Ausnahmeschritte gehören: nummeriere in der Form „5a1, 5a2, 5a3“ etc.

7.3 Beispiel (Kundenportal einer Versicherung für Schadensmeldungen)

ID	KundPort_12
Name	Schaden melden
Beschreibung	Kunde meldet einen Schaden
Auslösender Aktor	Kunde
Weitere Aktoren	Backend-Schadenssystem der Versicherung
Auslöser	Kunde hat einen Schadensfall, z.B. Autounfall
Vorbedingung	Kunde ist tatsächlich bei dieser Versicherung versichert
Nachbedingung	Schaden ist bei Versicherung gemeldet
Hauptszenario	1) Kunde identifiziert sich mit Kunden-ID und Passwort 2) Kunde gibt Datum und Beschreibung des Schadens ein 3) Portal leitet den Schaden weiter an Backend-Schadenssystem 4) Backend-System bestätigt Eingang 5) Portal schickt Eingangsbericht per Email an Kunden
Alternativszenario	1a) Kunde schreibt einen Brief an die Versicherung, mit Beschreibung des Schadens 2a) Sachbearbeiter liest den Brief und gibt anstelle des Kunden Datum und Beschreibung des Schadens in das System ein *(dann weiter mit 3)* 4b1) Backendsystem antwortet nicht 4b2) System schickt Email mit den Schadensdaten an den Verantwortlichen des Backend-Systems, damit dieser für Übergabe der Daten sorgt *(dann weiter mit 5)*
Ausnahmeszenario	4c) Versicherungsschutz des Kunden ist ausgesetzt, weil der Beitrag nicht bezahlt wurde. Backend-System schickt entsprechende Meldung an das Portal. 5c) Portal gibt die Ablehnung der Schadensmeldung an Kunden weiter **Nachbedingung in diesem Fall:** Schaden gilt nicht als gemeldet.

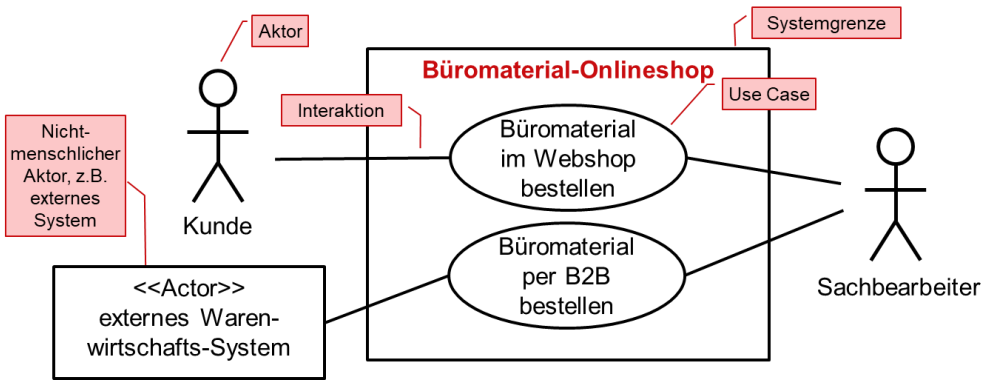
7.3.1 Checkliste: Use Case zu groß / zu klein? (nach Cockburn S. 57 ff.)

- *UC evtl. zu klein?* **User Happiness Test.** „Ist der Nutzer nach dem Use Case zufrieden?“
  - Bsp. für **ok**: Obiger Use Case, Schaden ist gemeldet.
  - Bsp. für **nicht ok**: Kunde schickt Email mit Schadensmeldung an Versicherung (und damit endet der Use Case). Dann fehlt Verarbeitung und Rückmeldung auf Seiten der Versicherung.
- *UC evtl. zu groß?* **Coffee Break Test.** „Nach so einem UC macht Nutzer Kaffeepause.“
  - Bsp. für **ok**: Obiger UC sollte in ca. 10 min machbar sein.
  - Bsp. für **nicht ok**: Wenn der obige UC auch noch die Regelung des Schadens umfassen würde (was Tage bis Wochen dauert).

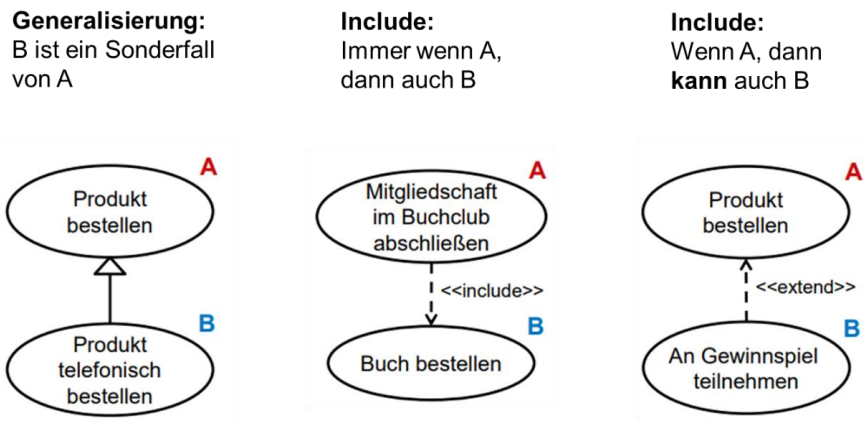
7.3.2 Extrahieren von Uses Cases aus textuellen Anforderungen

1. Verben anstreichen
2. Syntaktische Bereinigung
    a. Substantivierungen anstreichen, Hilfskonstruktionen auflösen
    b. Modalverben streichen (können, wollen, möchten, sollen, müssen, dürfen)
    c. Passiv in aktiv umwandeln
    d. Präsenz verwenden, Tempus-Hilfsverben streichen
3. Semantische Bereinigung
    a. Ziel und Kontext aber ohne Funktionalität - streichen
    b. Wiederholungen streichen
4. Ableitung der Use Cases
    a. Name: Objekt (Singular) + Verb (Infinitiv)
    b. Aktor: Aus Kontext erschließen, immer im Singular

7.4 Übersicht über Use Cases mit Hilfe von Use-Case-Diagrammen



Beziehungen zwischen Use Cases:



7.5 Weiterführende Literatur

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

8 Qualitätssicherung

Qualitätssicherung im Anforderungsmanagement dient dem Aufdecken und Beheben von Abweichungen zwischen den erhobenen Anforderungen und den Wünschen des Kunden. Die hierbei angewandten Maßnahmen lassen sich in **konstruktive** und **analytische** Qualitätssicherung aufteilen.

8.1 Glossar / Begriffsklärung

Begriff	Definition	Braucht man für …
Konstruktive Qualitätssicherung	Unterstützt bei der Erstellung von qualitativ hochwertigen Anforderungen	Qualitätssichernde Arbeitsprinzipien **während** der Anforderungsermittlung
Analytische Qualitättsicherung	Überprüfung der Güte von Anforderungen	Prüfung und Verbesserung der Anforderungs-Qualität **nach** der Ermittlung

8.2 Qualitätskriterien

8.2.1 … für Anforderungen allgemein

Kriterium	Definition
Korrektheit	Beschreibt fehlerfrei die Eigenschaft, welche das System erfüllen soll
Vollständigkeit	Vorhandensein aller notwendigen Informationen
Eindeutigkeit	Die Anforderung ist präzise formuliert und kann nicht mehrfach interpretiert werden
Konsistenz	Die Anforderung steht nicht im Widerspruch zu sich selbst oder zu anderen Anforderungen
Geltung	Es herrscht Konsens aller Beteiligten über die Anforderung
Priorisierung	Durch Bewertung der Beteiligten werden Anforderungen in eine Reihenfolge gebracht
Verifizierbarkeit	Die Erfüllung der Anforderung ist überprüfbar
Nachvollziehbarkeit	Die Anforderung besitzt eine eindeutige Bezeichnung (Schlüssel) und bietet alle relevanten Kontextinformationen (Quelle, Bezug zu anderen Anforderungen)
Verständlichkeit	Alle beteiligten sind in der Lage, die Anforderung zu verstehen
Umsetzbarkeit	Die Anforderung ist realisierbar

8.2.2 … speziell für Dokumente

Kriterium	Definition
Strukturiertheit	Korrekter, nachvollziehbarer Aufbau des Dokuments
Aktualität	Das Dokument gibt den aktuellen Stand der Anforderung wieder
Modifizierbarkeit	Änderungen an der Anforderung sollen leicht vorgenommen werden können
Zugreifbarkeit	Gesteuerter Zugriff für alle Beteiligten

Kriterium	Definition
Projizierbarkeit	Unterschiedliche Sichten auf das Dokument möglich, abhängig von der Nutzerrolle
Relevanz	Das Dokument enthält keine für die Problembeschreibung unwichtigen Informationen

8.3 Methoden der Qualitätssicherung

Zu Beginn eines Projekts werden vor allem **konstruktive** Techniken eingesetzt. Sind die ermittelten Anforderungen stabil, sollten stattdessen **analytische** Techniken angewendet werden.

Generelle Prinzipien zur Prüfung von Anforderungen	Checkliste für **konstruktive Qualitätssicherung** (während der Anforderungsermittlung)
• Beteiligung der richtigen Stakeholder • Trennung von Fehlersuche und -korrektur • Prüfung aus unterschiedlichen Sichten • Wiederholte Prüfung	• Durchgehend Art und Detailgrad der Anforderung festgelegt und eingehalten? • Ziel und Scope des Systems umfassend beschrieben? • Alle Anforderungen berücksichtigt? • Anforderungen unter Verwendung von Standard-Templates dokumentiert?

8.3.1 Analytische Qualitätssicherungstechniken

Technik	Funktioniert wie?	Wann anzuwenden?
Perspektivenbasiertes Lesen	Prüfer liest ein Dokument und nimmt dabei explizit eine bestimmte Stakeholder-Rolle ein (z.B. Kunde, Nutzer, Architekt, Tester, Entwickler, …)	Meist in Kombination mit Reviewtechniken
Review: Stellungnahme	Autor fragt Prüfer über Meinung zum Dokument	Kleine Projekte, schnelle Durchführung, nicht formal
Review: Walkthrough	Autor leitet Gruppe von Prüfern durch das Artefakt, so dass Fragen und Anmerkungen eingebracht werden können. Prüfer haben Dokument vorab gelesen. Ergebnisse werden protokolliert.	Mittel bis große Projekte ohne größere Qualitätsauflagen
Review: Inspektion	Sehr formale Stellungnahme mit unterschiedlichen Phasen: Planung, Vorbesprechung, individuelle Vorbereitung, Reviewsitzung, Nachbereitung und Bewertung. Sehr schwergewichtig.	Projekte mit hohen Qualitätsanforderungen, sehr schwergewichtig / zeitintensiv
Anforderungen durch **Prototypen** prüfen	Ein Teil des Systems (oft als „horizontaler Durchstich“ wird gemäß der Anforderungen als Prototyp implementiert und durch Stakeholder bewertet.	Technische / kritische Systeme, ressourcenintensiv

8.4 Weiterführende Literatur

- Pohl und Rupp, 2011, S. 101-117
- Rupp, 2014, S. 287-298
- Schienmann, 2002, S. 176-185

9 Erstellen eines agilen Backlogs

Agile Anforderungen werden (anders als bei der dokumentenzentrierten Anforderungsermittlung, die auf einen Wasserfall-Entwicklungsprozess abgestimmt ist) zunächst nur grob festgehalten. Eine detaillierte Spezifikation erfolgt erst unmittelbar vor der Implementierung, dann in der Regel als User Stories.

9.1 Glossar / Begriffsklärung

Begriff	Definition	Braucht man für …
(Investment) Theme	Gemeinsame „Mission“ für eine große zusammenhängende Menge von Features, unter Berücksichtigung von Unternehmensstrategien	Im allgemeinen beschreibt das Theme ein neu zu erstellendes IT-System, oder eines seiner Hauptbereiche.
Epic	Grobgranulare, allgemeine Anforderung, die weiter in User Storys detailliert wird.	Ein Epic spezifiert grob einen Bereich des IT-Systems, den ein Entwicklungsteam über eine längere Zeit umsetzt. • 1 Theme = <n> Epics
User Story	Beschreibt konkret, was ein System aus Sicht eines Nutzers (User) tun soll.	Anforderungseinheit, die für einen Entwicklungs-Sprint (2-4 Wochen) definiert wird. • 1 Epic = <n> User Stories • I.a.: Epic >= Use Case >= User Story • User Story muss in einem Sprint abzuschließen sein
Task	Einzelne Aufgaben zu einer Story, z. B. bestimmte Code-Teile schreiben	Detaillierung einer User Story; im Gegensatz zur User Story hat der Task i.a. einen Technikbezug • 1 User Story = <n> Tasks
Iteration (Sprint)	Feste Zeiteinheit (2-4 Wochen), in der das Entwicklungsteam eine bestimmte Menge User Storys umgesetzt.	Features werden in Releases geplant und in Iterationen / Sprints umgesetzt.
Release	Langfristige Zeitplanung der Features.	
Product Backlog	Sammlung der priorisierten Anforderungen (Investment Themes, Epics, User Stories).	Je weiter der Zeitpunkt der Umsetzung entfernt ist, desto gröber kann die Anforderung formuliert sein.
Sprint Backlog	Auszug von User Stories aus dem Product Backlog, um sie in einer Iteration / einem Sprint umzusetzen.	Arbeitsplan für den Sprint.
Product Owner	Priorisiert Kundenanforderungen und plant das Release.	Der Anforderungsmanager in agilen Projekten.

9.2 Abgrenzung (Investment) Theme – Epic – User Story – Task

Das folgende Beispiel ist adaptiert und übersetzt von Cohn (o. J.). Die Anforderungen beschreiben auszugsweise die Anforderungen ein System zum Management von Schulungs-Dienstleistungen. Es gilt: **TH** = (Investment) Theme, **EP** = Epic, **US** = User Story, **TA** = Task.

- **TH:** Management von Trainer-Profilen
- **TH:** Management von Kursen und Veranstaltungen
- **TH:** Bereitstellen von Dokumentation
  - **EP:** As Nutzer des Portals kann ich FAQ-Einträge lesen.
  - **EP:** Als fachlicher Administrator kann ich FAQ-Einträge pflegen.
    - **US:** Als fachlicher Administrator kann ich einen neuen „FAQ“-Eintrag mit Feldern für Überschrift und Erklärungstext einfügen, so dass ich wiederkehrende Kundenfragen nur einmal beantworten muss
    - **US:** Als fachlicher Administrator kann ich meinen FAQ-Erklärungstext mit Wiki-Markup formatieren, so dass ich den Text einfach struktiert und lesbar gestalten kann.
      - **TA:** Definition des Formats für das Wiki-Markup
      - **TA:** Implementiere Konfigurations-Datenbank für das erlaubte Wiki-Markup
      - **TA:** Implementiere Wiki => HTML Converter

9.3 Template: User Story

Bestandteil	Inhalt	Beispiel
Als <Rolle>,	Rolle des Nutzers, aus dessen Perspektive diese Anforderung beschrieben ist	Als fachlicher Administrator unseres Kundenportals,
kann ich <Aktivität>,	Hauptteil der User Story: was soll das IT-System dem Nutzer ermöglichen?	kann ich einen neuen „FAQ“-Eintrag mit Feldern für Überschrift und Erklärungstext einfügen,
so dass <Geschäftswert>.	Grund, aus dem die User Story für den Nutzer sinnvoll ist (sinnvoll für Priorisierung)	so dass ich wiederkehrende Kundenfragen nur einmal beantworten muss.

9.3.1 Formulierungsregeln: Stellen Sie sicher, dass für User Stories “INVEST” gilt

- **I**ndependent
- **N**egotiable
- **V**aluable
- **E**stimable
- **S**mall
- **T**estable

9.3.2 Teilungsregeln für User Stories (nach Leffingwell, 2010, p. 112f.)

Man tendiert dazu, User Stories „zu groß“ zu formulieren. Daumenregel: Eine User Story muss so klein sein, dass **ein Entwickler** mindestens **eine User Story** (besser mehrere) **in einem Sprint** umsetzen kann.

Wenn das Entwicklungsteam bei der Sprintplanung das Feedback gibt, dass die User Story zu groß ist, kann der Product Owner sie anhand der folgenden Regeln jeweils mehrere Einzel-Stories zerlegen.

Nr.	Methode der Aufspaltung	Beispiel vorher	Beispiel hinterher
1	Zerlegen der Aktivitätsschritte	Als fachlicher Administrator möchte ich einen Blog-Post editieren, speichern, und dann die geänderte Version veröffentlichen.	• … möchte ich editieren und speichern … • … möchte ich die Veröffentlichung anstoßen …
2	Zerlegen nach CRUD (Create / Read / Update / Delete)	Als Kunde kann ich meinen Account selbst verwalten, …	• … Account anlegen … • … Accountdaten nachlesen … • … Accountdaten ändern … • … Account löschen …
3	Zerlegen nach Daten-Attribut	… möchte ich die Kundendaten nach Adressattributen filtern …	• … nach PLZ … • … nach Bundesland …
4	Zerlegen nach Datenart	Als Kunde möchte ich per Kreditkarte zahlen können, …	• … per Visa-Karte … • … per Master-Karte …
5	Zerlegen nach Datenbasis	Als Kunde in der EU möchte ich Emails des Portals in meiner Landessprache lesen, …	• Als Kunde in Deutschland … • Als Kunde in Frankreich … • Als Kunde in Italien …
6	Ergänze Variation: „Simple Lösung zuerst“	Als fachlicher Administrator kann ich einen Blog-Post nach „WYSIWIG“ editiere, …	• … als einfachen ASCII-Text … • … als ASCII-Text mit Wiki-Markup … • … nach „WYSIWIG“ …
7	Ergänze Variation: Eingabe-Art	Als Sachbearbeiter kann ich den Preis eines Angebots editieren, …	• … im Preisfeld der Detailsansicht … • … zusätzlich per Inline-Editing in der Übersichtstabelle …

9.4 Weiterführende Literatur

- Cohn, 2004, S. 17-30 und 75-84
- Cohn, (o.D.)
- Leffingwell, 2010, S. 31-45, 83-92 und 99-117

10 Anhang: Literatur

- Bono, E. de. (1989). Das Sechsfarben-Denken. Ein neues Trainingsmodell. Düsseldorf: Econ.
- Calabria, T. (2004). An introduction to personas and how to create them. Abgerufen 30. April 2015, von 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 (1. Aufl.). Addison-Wesley Professional.
- Cohn, M. (o. J.). Product Backlog Example. Abgerufen 20. Januar 2019, von 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). Systematisches Requirements Engineering: Anforderungen ermitteln, dokumentieren, analysieren und verwalten (5., überarb. Aufl.). Heidelberg: dpunkt.verlag GmbH.
- Gürtler, J., & Meyer, J. (2013). 30 Minuten Design Thinking. Offenbach: GABAL.
- Kintz, M. (2007). Personas (Hauptseminar Requirements Engineering). Abgerufen von 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 (1. Aufl.). Upper Saddle River, NJ: Addison Wesley.
- Leffingwell, D., Widrig, D., & Yourdon, E. (2003). Managing Software Requirements: A Use Case Approach (0002 Aufl.). Boston: Addison Wesley Pub Co Inc.
- Pichler, R. (2013). Agiles Produktmanagement mit Scrum: Erfolgreich als Product Owner arbeiten (2., korrigierte Auflage). Heidelberg: dpunkt.verlag GmbH.
- Pohl, K. (2008). Requirements Engineering: Grundlagen, Prinzipien,Techniken (2., korrigierte Auflage.). 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., & die SOPHISTen (2014). Requirements-Engineering und -Management: Aus der Praxis von klassisch bis agil (6., aktualisierte und erweiterte Auflage). München: Carl Hanser Verlag GmbH & Co. KG.
- Schienmann, B. (2001). Kontinuierliches Anforderungsmanagement . Prozesse - Techniken - Werkzeuge (1. Aufl.). München ; Boston u.a.: Addison-Wesley.
- SOPHISTen, die. (2014). Requirements-Engineering - die kleine RE-Fibel. Selbstverlag. Abgerufen von https://www.sophist.de/fileadmin/SOPHIST/Puplikationen/Broschueren/RE-Broschuere_Komplett_Final_-_Update_1.pdf
- Weit e.V. (2006). Das 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, abgerufen 20.01.2019
- Werner, S. (2006). Grundlagen der Programmentwurfstechnik 1. Abgerufen von http://ti.uni-due.de/ti/de/education/teaching/ss06/pet/folien/Folien%201.pdf