



Kapitel 1: Requirements Engineering

Dr. Dominik Haneberg

Software Engineering 2

Wintersemester 2018/2019



Abschnitt 1.1

MOTIVATION UND INHALTE

Re



Foto: D. Haneberg

Inhalte dieses Kapitels

- Requirements Engineering: Das Problem
- Requirements Engineering Begriffe und Aufgaben
- Integration in den Softwareentwicklungsprozess
- Kontext, Stakeholder, Ziele und Anforderungen
- Arten von Anforderungen
- Anforderungsgewinnung
- Anforderungsdokumentation
- Prüfen und Verwalten von Anforderungen

Probleme, Fragen und Herausforderungen



Institute for
Software & Systems
Engineering

*„There is nothing so useless as
doing efficiently that which
should not be done at all.“*

— Peter Drucker

*„The hardest single part
of building a software
system is deciding
precisely what to build.“*

— Frederick P. Brooks

*„It isn't that they can't see the solution.
It is that they can't see the problem.“*

— Gilbert Keith Chesterton

*„The cheapest, fastest and
most reliable components of
a computer system are
those that aren't there.“*

— Gordon Bell

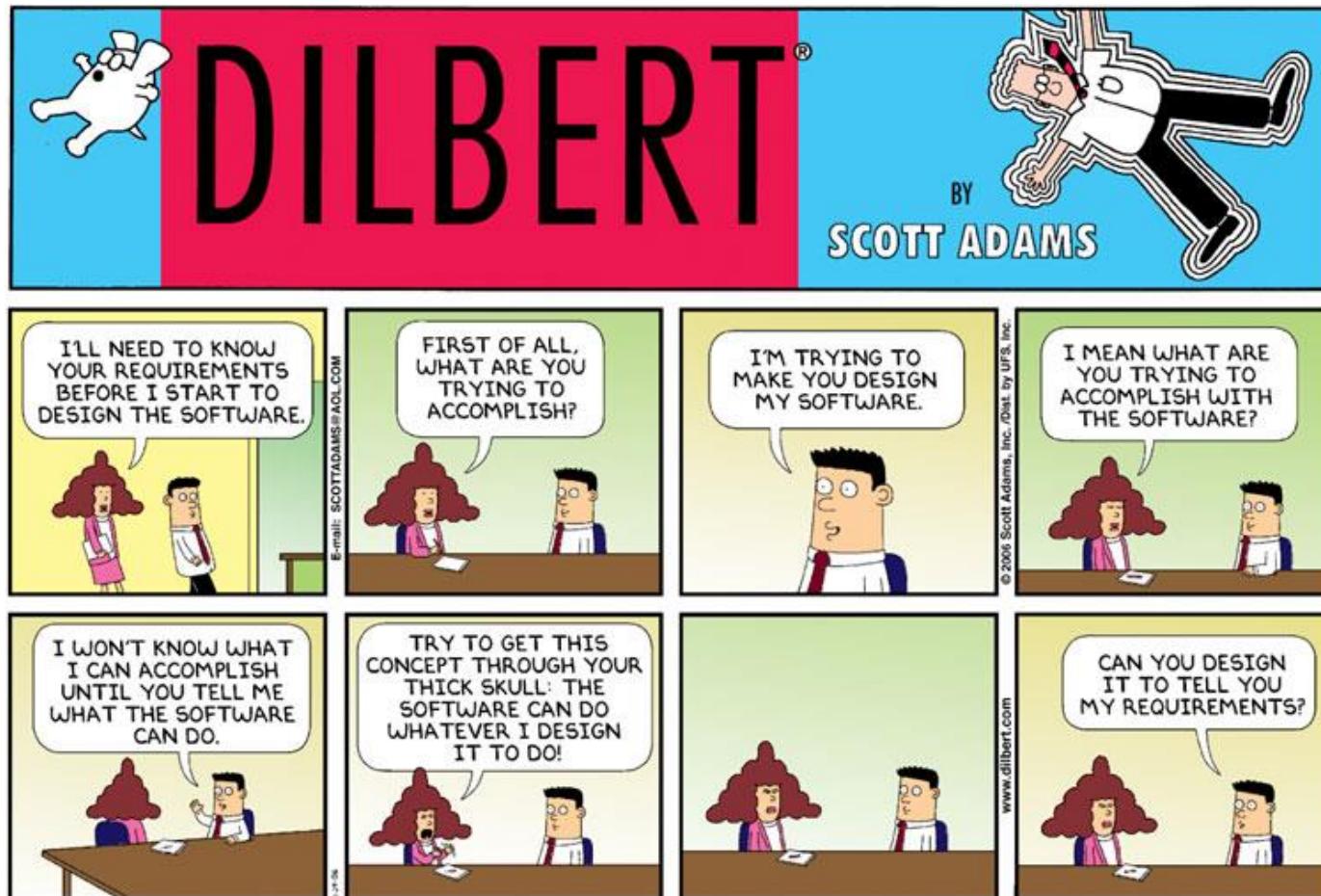
„Bittet, so wird euch gegeben“

— Matthäus 7,7

*„I have always wished for my computer to be as
easy to use as my telephone; my wish has come
true because I can no longer figure out how to
use my telephone.“*

— Bjarne Stroustrup

Der Stakeholder, das unbekannte Wesen



© Scott Adams, Inc./Dist. by UFS, Inc.

Abschnitt 1.2

GRUNDBEGRIFFE UND AUFGABEN

Definition Requirements Engineering



Institute for
Software & Systems
Engineering

Requirements Engineering (RE), im Deutschen gelegentlich Anforderungsmanagement genannt, ist eine Managementaufgabe für die effiziente und fehlerarme Entwicklung komplexer Systeme. Es umfasst die Themengebiete Anforderungsdefinition (englisch requirements definition) und Anforderungsverwaltung (englisch requirements management).

Anforderungsdefinition beinhaltet dabei die Teilgebiete Anforderungsermittlung (englisch requirements elicitation), Anforderungsdokumentation (englisch requirements documentation) und Anforderungsvalidierung (englisch requirements validation), während Anforderungsverwaltung Maßnahmen zur Steuerung, Kontrolle und Verwaltung von Anforderungen, also Risikomanagement, Änderungsmanagement und Umsetzungsmanagement umfasst.

Wikipedia: http://de.wikipedia.org/wiki/Requirements_Engineering

Drei zentrale Aspekte des Requirements Engineering

Requirements Engineering

1. Das systematische, disziplinierte und quantitativ erfassbare Vorgehen beim Spezifizieren (Erfassen, Beschreiben und Prüfen) von Anforderungen an ein System
2. Verstehen und Dokumentieren, was der Kunde will oder braucht
3. Spezifikation und Verwaltung von Anforderungen, mit dem Ziel, das Risiko zu minimieren, etwas zu entwickeln, was dem Kunden nicht gefällt oder nicht nützt

Was ist Requirements Engineering?



Technische Sicht

Requirements Engineering ist das systematische und disziplinierte Vorgehen beim Spezifizieren (Erfassen, Beschreiben und Prüfen) und Verwalten von Anforderungen an ein System.

- Typischerweise **Softwaresysteme** oder **software-intensive Systeme**
- Ziel ist eine vollständige, unmissverständliche und widerspruchsfreie Spezifikation
- Typischerweise am Anfang
- Klingt nach Aktenbergen und Bürokratie
- Wo sind die Menschen?
- Ist das realistisch?

Was ist Requirements Engineering?



Kundenorientierte Sicht

Institute for
Software & Systems
Engineering

Requirements Engineering — Verstehen und Dokumentieren, was der Kunde will oder braucht.

- Menschenzentrierte Sicht
- Ziel sind zufriedene Kunden
- Was sind Kunden?
- Warum wollen oder brauchen?
- Warum nicht gleich programmieren?

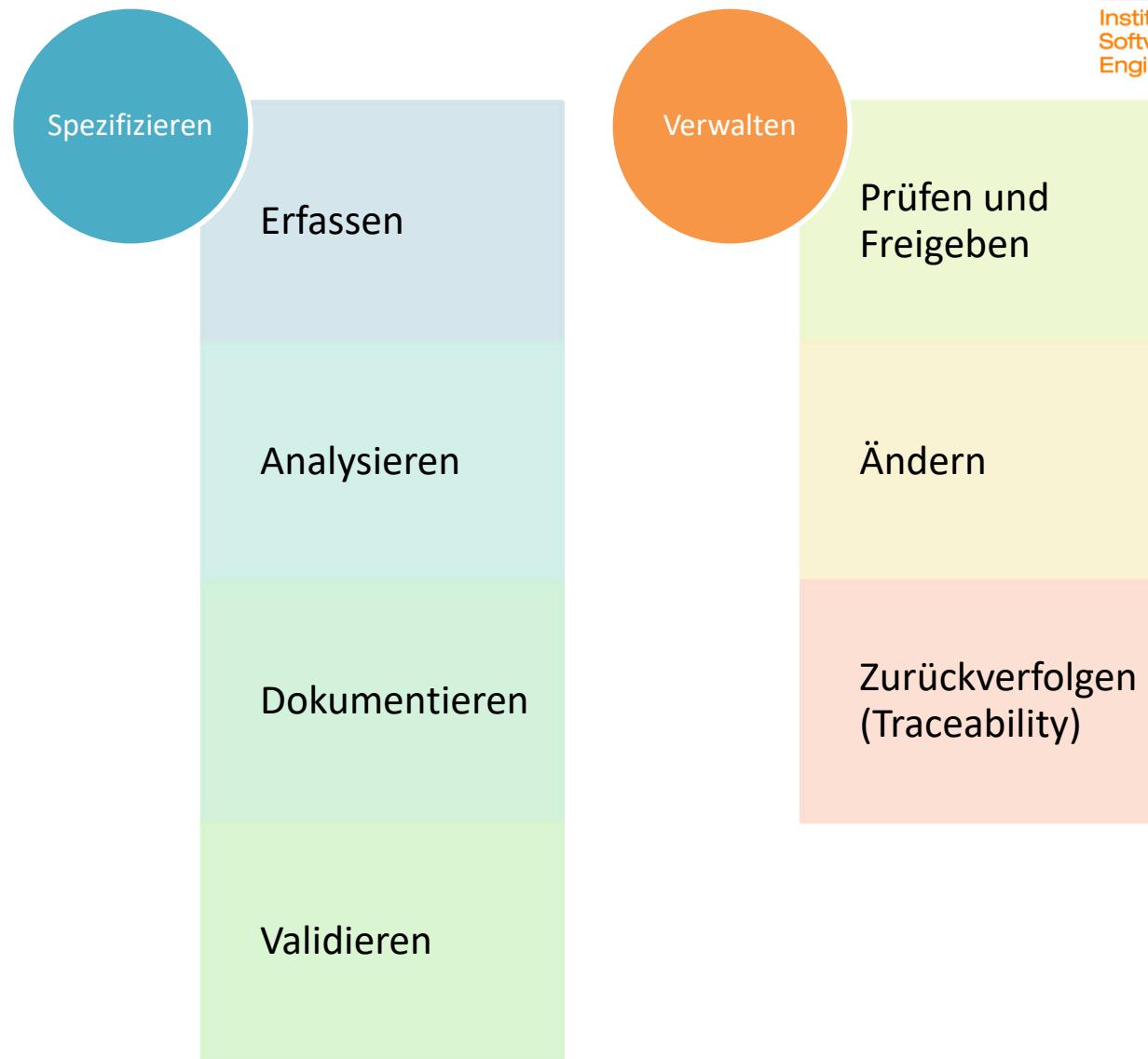
Was ist Requirements Engineering?

Risikoorientierte Sicht

Requirements Engineering — Dokumentieren und Verwalten von Anforderungen mit dem Ziel, das **Risiko** zu minimieren, ein System zu entwickeln, das vom Kunden nicht akzeptiert wird (ihm nicht gefällt oder nicht nützt).

- Man stelle sich die Frage: „Wie viel müssen wir tun, damit das Risiko so klein wird, dass wir bereit sind es zu akzeptieren?“
- Aufwand für das Requirements Engineering sollte umgekehrt proportional zur Größe des Risikos sein, das man bereit ist, einzugehen.

Aufgaben des Requirements Engineering



- Kosten reduzieren

- Geringere Herstellungskosten
- Weniger Nachbesserungen im Betrieb
- Geringere Wartungskosten

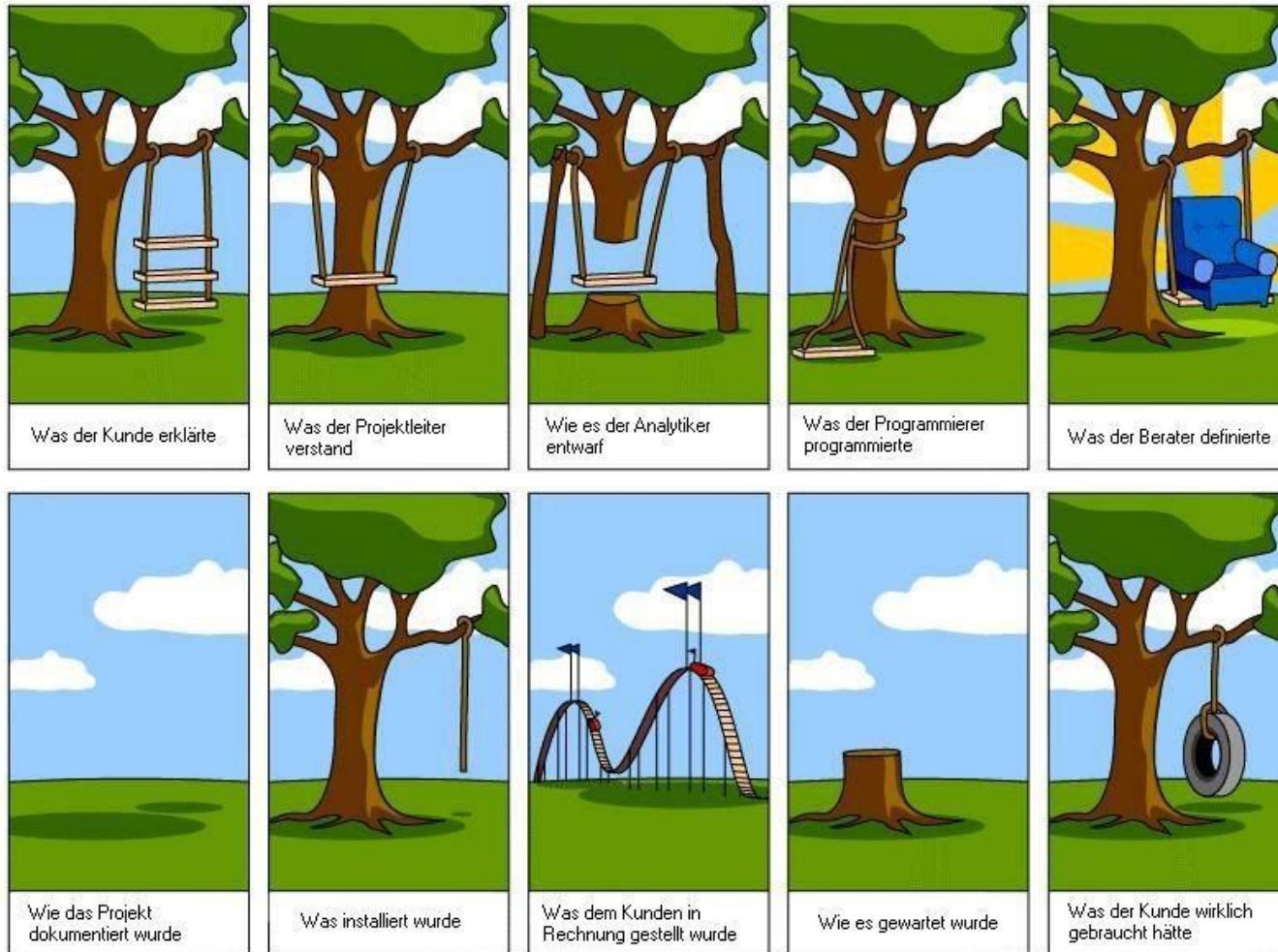
Mehr verdienen

- Risiken minimieren

- Kundenerwartungen erfüllen
- Termine und Kosten einhalten

Kundenzufriedenheit
verbessern

Anforderung im Wandel der Zeit



<http://projectcartoon.com/create/>

Anforderung und Spezifikation

Anforderung

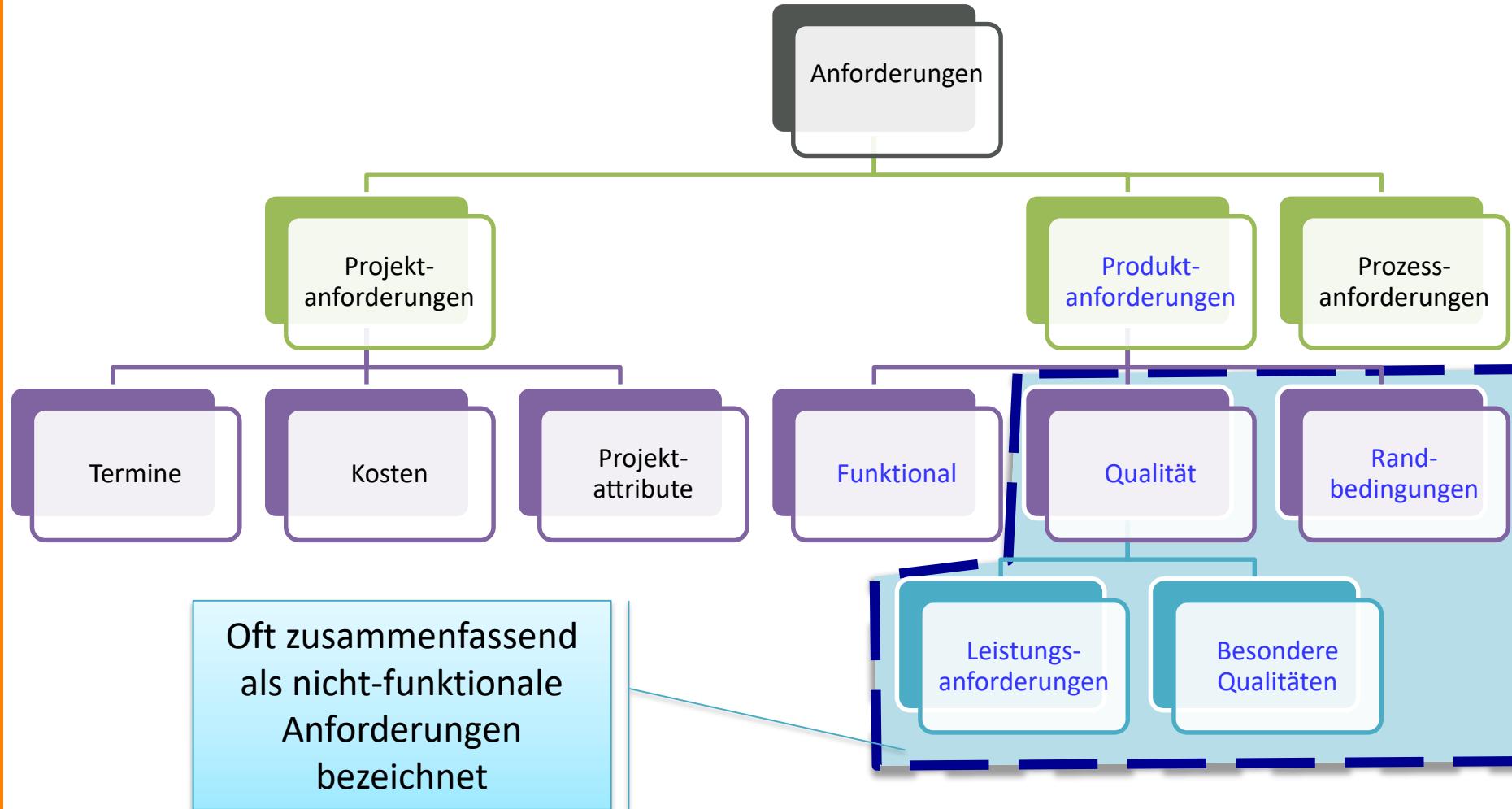
1. Eine Bedingung oder Fähigkeit, die von einem Benutzer zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird.
2. Eine Bedingung oder Fähigkeit, die ein System erfüllen oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder andere, formell vorgegebene Dokumente zu erfüllen.
3. Eine dokumentierte Repräsentation einer Bedingung oder Fähigkeit nach (1) oder (2).

Anforderungsspezifikation

Zusammenstellung (aller) Anforderungen an ein System.

Institute of Electric and Electronic Engineers: IEEE Standard Glossary of Software Engineering Terminology (IEEE Std. 610.12-1990)

Kategorisierung von Anforderungen



Abschnitt 1.3

VARIANTEN VON SOFTWAREPROJEKTEN

- **Neuentwicklung:** System wird komplett neu konzipiert und realisiert: Keine Altlasten, keine Datenmigration
- **Reengineering-Projekt:** Verbesserung oder Weiterentwicklung eines bestehenden Systems, möglicherweise Plattformwechsel
- **Migrationsprojekt:** Übertragung von Daten aus einer (alten) Datenquelle in eine andere. Reines Migrationsprojekt hat nur geringe Implementierungsanteile
- **Weiterentwicklungsprojekt:** Erweiterung oder Verbeserung eines Systems im Produktiveinsatz

- **Individualsoftware:** Wird im Auftrag eines einzelnen Kunden speziell für diesen entwickelt, Kunde zahlt für die Entwicklungsarbeit
- **Software-Produkt:** Wird für eine beliebige Anzahl potentieller Kunden entwickelt, Kunden Zahlen für die Benutzung des Produkts (über unterschiedliche Lizenzformen)

- **Datenzentriertes System:** Primäraufgabe des Systems ist die Verwaltung von Daten als zentraler Ressource: Erzeugen, Ändern, Persistieren, Löschen und Transformieren
- **Eingebettetes System:** Systeme aus Hard- und Softwarekomponenten, mit Sensoren und Aktuatoren zur Untersuchung bzw. Veränderung der Umgebung. Reagiert auf Ereignisse. Softwareanteil für den Anwender nicht unmittelbar erkennbar

Softwareentwicklung: Beauftragungsmodelle

- **Inhouse-Projekt:** Kunde und Ersteller des Systems sind aus derselben Organisation, Systemerstellung erfolgt für den Eigenbedarf. Systemerstellung durch IT-Abteilung, Auftraggeber ist ein Fachbereich. Meist keine formalisierte Vertragssituation zwischen Auftraggeber und -nehmer
- **Vergabeprojekt:** Systemersteller nicht aus der Organisation des Auftraggebers (Fremdvergabe), explizite Vertragssituation zwischen Auftraggeber und -nehmer
- **Ausschreibungsprojekt:** Variante des Vergabeprojekts, Auftraggeber holt im Rahmen einer Ausschreibung Angebote verschiedener Dienstleister ein. Zuteilung nach Bewertung der Angebote anhand definierter Kriterien. Explizite Vertragssituation zwischen Auftraggeber und –nehmer, typischerweise Festpreisprojekt

Abschnitt 1.4

ZIELE, ANFORDERUNGEN UND SZENARIEN

Ziel

Bezeichnet einen in der Zukunft liegenden, gegenüber dem Gegenwärtigen im Allgemeinen veränderten, erstrebenswerten oder angestrebten Zustand.

Anforderung

1. Eine Bedingung oder Fähigkeit, die von einem Benutzer zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird.
2. Eine Bedingung oder Fähigkeit, die ein System erfüllen oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder andere, formell vorgegebene Dokumente zu erfüllen.
3. Eine dokumentierte Repräsentation einer Bedingung oder Fähigkeit nach (1) oder (2).

Wikipedia: <http://de.wikipedia.org/wiki/Ziel>

- Ziel und Anforderung beschreiben beide etwas **zu Erreichendes**
- Begriffe sind aber nicht synonym
 - Ziel
 - Ein zu erreichender **Zustand**
 - Globalere oder abstraktere Sicht
 - Anforderung
 - Eine zu erreichende **Eigenschaft**
 - Konkreter
 - Meist eine zur Erreichung eines Ziels **notwendige Bedingung** oder **Eigenschaft**

Szenario

Beschreibt die Erfüllung oder Nichterfüllung von Zielen und somit die Stakeholder-Intentionen anhand eines konkreten Beispiels. Ein Szenario enthält typischerweise eine Folge von Interaktionsschritten mit dem System.

- Szenarien sind zentrales Element bei der Beschreibung von Geschäftsvorfällen (**Use-Cases**, ursprünglich aus Jacobsons OOSE-Ansatz)
- Heute zentraler Bestandteil von UML und RUP



Abschnitt 1.5

PRIORISIERUNG

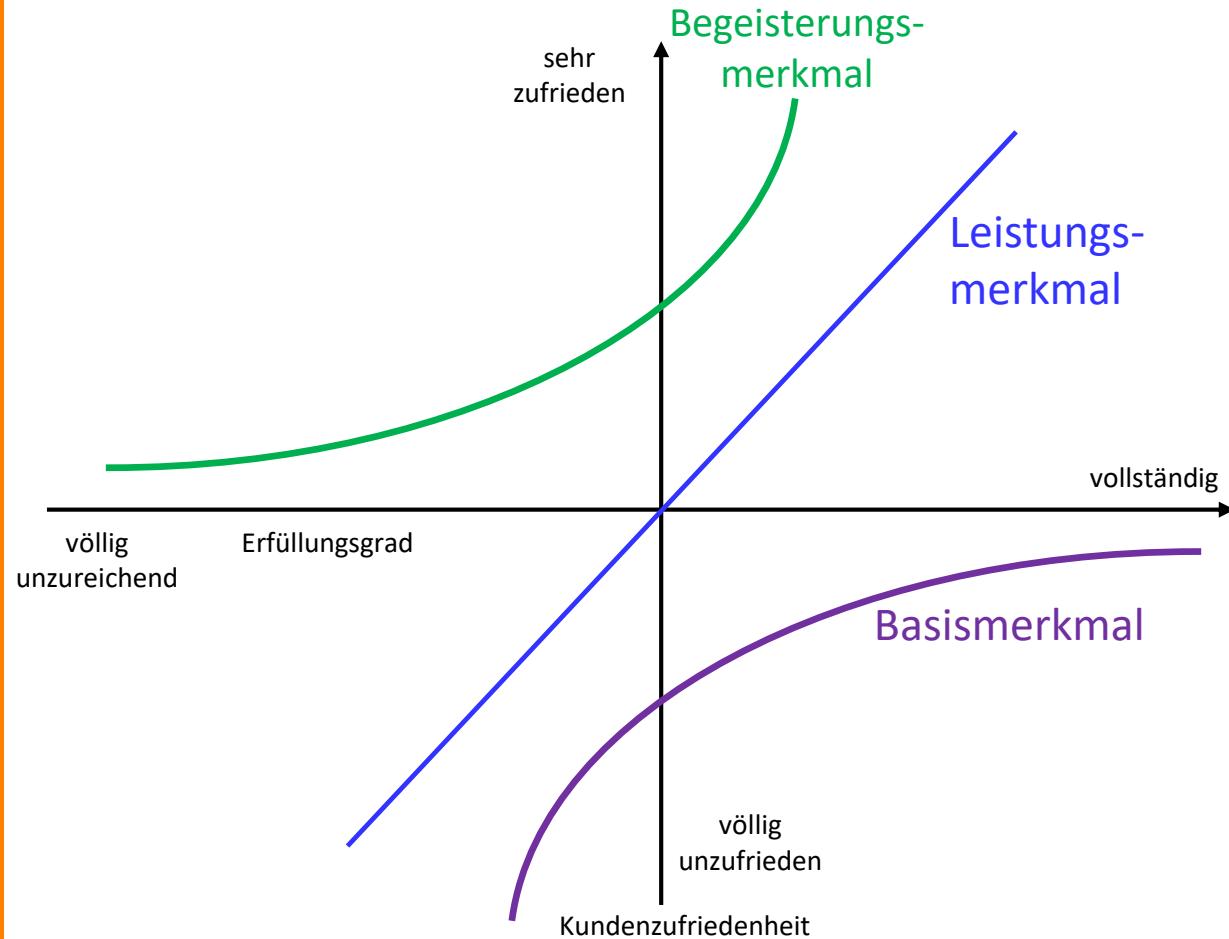
- Einfache Priorisierungsstrategie [IEEE 830]:
 - **Muss:** Unverzichtbare Anforderung
 - **Soll:** Wichtig, aber bei zu hohen Kosten verzichtbar
 - **Wunsch:** Wäre nett, aber nicht essentiell

Priorisierung ist

- nötig bei
 - harten Kostengrenzen
 - Beschaffung
- wünschenswert bei
 - Festlegung der Inkremeante bei inkrementeller Entwicklung
 - Releaseplanung bei der Weiterentwicklung bestehender Systeme

Institute of Electric and Electronic Engineers: IEEE Recommended Practice
for Software Requirements Specifications (IEEE Std. 830-1998)

Entwickelt von Noriaki Kano 1978 an der Uni Tokio



Außerdem:

- **Unerhebliche Merkmale:** Verursachen weder Zufriedenheit, wenn vorhanden, noch Unzufriedenheit, wenn nicht vorhanden
- **Rückweisungsmerkmale:** Ablehnung des Produkts, wenn Merkmal vorhanden

Das Kano-Modell: Messung der Erwartungshaltung

- Messung der Erwartungshaltung mit Fragebogen
- Für jedes Produktmerkmal zwei Fragen: Einmal positiv (funktional) einmal negativ (dysfunktional)
- **Positive Frage:** Was würden Sie davon halten, wenn das Produkt über die Eigenschaft X verfügt?
- **Negative Frage:** Was würden Sie davon halten, wenn das Produkt NICHT über die Eigenschaft X verfügt?
- Beantwortung mit einer von fünf vordefinierte Antworten:
 - Das würde mich sehr freuen (1)
 - Das setze ich voraus (2)
 - Das ist mir egal (3)
 - Das nehme ich gerade noch hin (4)
 - Das würde mich sehr stören (5)

Das Kano-Modell: Messung der Erwartungshaltung

Kundenanforderung		Dysfunktional				
		(1)	(2)	(3)	(4)	(5)
Funktional	(1)	W	Be	Be	Be	L
	(2)	R	U	U	U	Ba
	(3)	R	U	U	U	Ba
	(4)	R	U	U	U	Ba
	(5)	R	R	R	R	W

Be = Begeisterungsmerkmal

L = Leistungsmerkmal

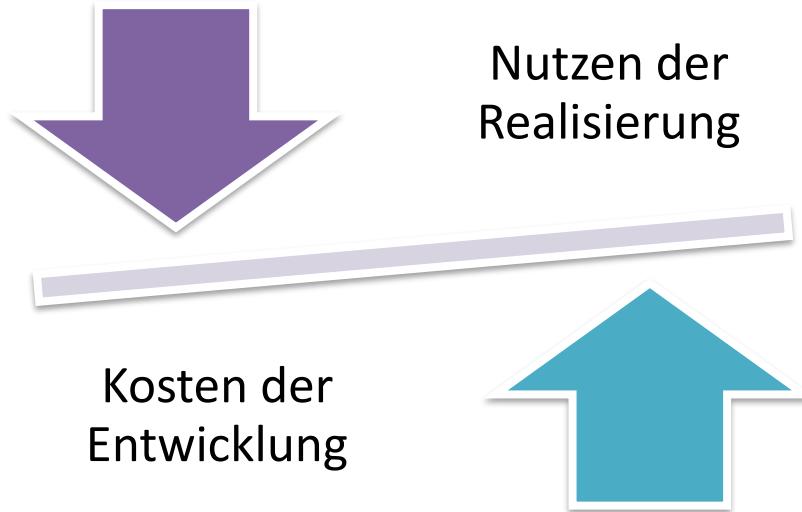
Ba = Basismerkmal

U = Unerhebliches Merkmal

R = Rückweisungsmerkmal

W = Widerspruch

Wertbasierte Priorisierung



Kosten und Nutzen **realistisch** rechnen:

- **Kumulierter Nutzen** über die Lebensdauer
- **Kapitalkosten** (inkl. Verzinsung) über die Entwicklungs- und Lebensdauer

Besonders geeignet, wenn viele wichtige aber nicht unverzichtbare Funktionen und Merkmale für ein System gefordert sind

M. Denne, J. Cleland-Huang: The Incremental Funding Method – A Data Driven Approach to Software Development, IEEE Software, 2004

- Wertbasierte Priorisierung setzt präzise und absolute Werte für Kosten und Nutzen voraus
- Der **Cost-Value Approach** bewertet Requirements anhand von **Schätzungen** der **Kosten** und des **Nutzens**
- Cost-Value Approach verwendet Expertenschätzungen für Nutzen (durch Domänenexperten) und Kosten (durch erfahrene Entwickler)

Cost-Value-Approach: Vorgehen

Schritt 1: Nutzen bewerten

- Bewertung erfolgt durch paarweises Vergleichen aller n Requirements
- Die Vergleichswerte bilden eine Matrix $A = (a_{ij})$, wobei folgende Vergleichswerte verwendet werden:

Zahlenwert	Bedeutung in Worten
1	Gleich wichtig
3	Etwas wichtiger
5	Viel wichtiger
7	Sehr viel wichtiger
9	Uneingeschränkt wichtiger

- Die Werte 2,4,6,8 werden als Zwischenwerte eingesetzt, wenn Kompromisse nötig sind.
- Der zu a_{ij} umgekehrte Vergleich a_{ji} entspricht dem Kehrwert von a_{ij} , also $\frac{1}{a_{ij}}$

Eintrag in Zelle (i,j) heißt: Req_i ist „ (a_{ij}) “ als Req_j

Schritt 1: Nutzen bewerten

- Aus der Vergleichsmatrix wird der **relative Wert** jedes Requirements bestimmt
- Der relative Wert eines Requirements ist der Anteil, den das Requirement am Gesamtwert der Requirementsmenge hat
- Der relative Wert der Requirements ergibt sich aus dem sogenannten **priority vector**
- Der priority vector ist der zum maximalen Eigenwert gehörende Eigenvektor

Schritt 1: Nutzen bewerten

- Den priority vector bestimmt man durch ein Näherungsverfahren:
 1. Alle Spalten normieren (Spaltensumme ist 1)
 2. Zeilensummen der normierten Matrix bestimmen
 3. Zeilensummen durch Anzahl der Requirements dividieren
 4. Der Ergebnisvektor ist der genäherte priority vector

Schritt 2: Konsistenz prüfen

- Bestimme **consistency index** und **consistency ratio**, um die innere Konsistenz der Vergleichsmatrix abzuschätzen
- Den consistency index berechnet man für eine Vergleichsmatrix A und priority vector p wie folgt:
 1. Bilde Matrixprodukt $r = Ap$
 2. Dividiere jedes Element von r durch das korrespondierende

$$\text{von } p: s = \begin{pmatrix} \frac{r_1}{p_1} \\ \vdots \\ \frac{r_n}{p_n} \end{pmatrix}$$

3. Bestimme maximalen Eigenwert von A : $\lambda_{max} = \frac{\sum_{i=1}^n s_i}{n}$
4. Der consistency index ist dann: $CI = \frac{\lambda_{max}-n}{n-1}$

Schritt 2: Konsistenz prüfen

- Die consistency ratio ist ein Maß für die Qualität der Matrix von paarweisen Vergleichen
- Zur Ermittlung benötigt man neben CI noch den passenden **random index**:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RI	0,00	0,00	0,58	0,9	1,12	1,24	1,32	1,41	1,45	1,49	1,51	1,48	1,56	1,57	1,59

- Die consistency ratio ist dann: $CR = \frac{CI}{RI}$
- consistency ratio-Werte $\leq 0,1$ gelten als akzeptabel

Cost-Value-Approach: Beispiel

4 Requirements Req1 bis Req4 sollen priorisiert werden

Experteneinschätzung lieferte folgende Vergleichsmatrix:

	Req1	Req2	Req3	Req4
Req1	1	1/3	2	4
Req2	3	1	5	3
Req3	1/2	1/5	1	1/3
Req4	1/4	1/3	3	1

Cost-Value-Approach: Beispiel

Normierung der Spalten ergibt:

	Req1	Req2	Req3	Req4
Req1	0,21	0,18	0,18	0,48
Req2	0,63	0,54	0,45	0,36
Req3	0,11	0,11	0,09	0,04
Req4	0,05	0,18	0,27	0,12

$$a'_{ij} = \frac{a_{ij}}{\sum_{l=1}^n a_{lj}}$$

Berechnung der Zeilensummen ergibt:

	Req1	Req2	Req3	Req4	Σ
Req1	0,21	0,18	0,18	0,48	1,05
Req2	0,63	0,54	0,45	0,36	1,98
Req3	0,11	0,11	0,09	0,04	0,34
Req4	0,05	0,18	0,27	0,12	0,62

Für den priority vector ergibt sich:

$$\frac{1}{4} \begin{pmatrix} 1,05 \\ 1,98 \\ 0,34 \\ 0,64 \end{pmatrix} = \begin{pmatrix} 0,26 \\ 0,50 \\ 0,09 \\ 0,16 \end{pmatrix}$$

Der priority
vector

Das heißt, in unserem Beispiel repräsentiert Req1 26% des gesamten Werts der Requirements, Req2 50%, Req3 9% und Req4 16%

Cost-Value-Approach: Beispiel

Für λ_{max} ergibt sich:

$$\begin{pmatrix} 1 & 1/3 & 2 & 4 \\ 3 & 1 & 5 & 3 \\ 1/2 & 1/5 & 1 & 1/3 \\ 1/4 & 1/3 & 3 & 1 \end{pmatrix} \begin{pmatrix} 0,26 \\ 0,50 \\ 0,09 \\ 0,16 \end{pmatrix} = \begin{pmatrix} 1,22 \\ 2,18 \\ 0,37 \\ 0,64 \end{pmatrix}$$

$$\begin{pmatrix} 1,22/0,26 \\ 2,18/0,50 \\ 0,37/0,09 \\ 0,64/0,16 \end{pmatrix} = \begin{pmatrix} 4,66 \\ 4,40 \\ 4,29 \\ 4,13 \end{pmatrix}$$

$$\lambda_{max} = \frac{4,66 + 4,40 + 4,29 + 4,13}{4} = 4,37$$

Für den consistency index ergibt sich:

$$CI = \frac{4,37 - 4}{4 - 1} = 0,12$$

Für die consistency ratio ergibt sich:

$$CR = \frac{0,12}{0,90} = 0,14$$

Schritt 3: Kosten bewerten

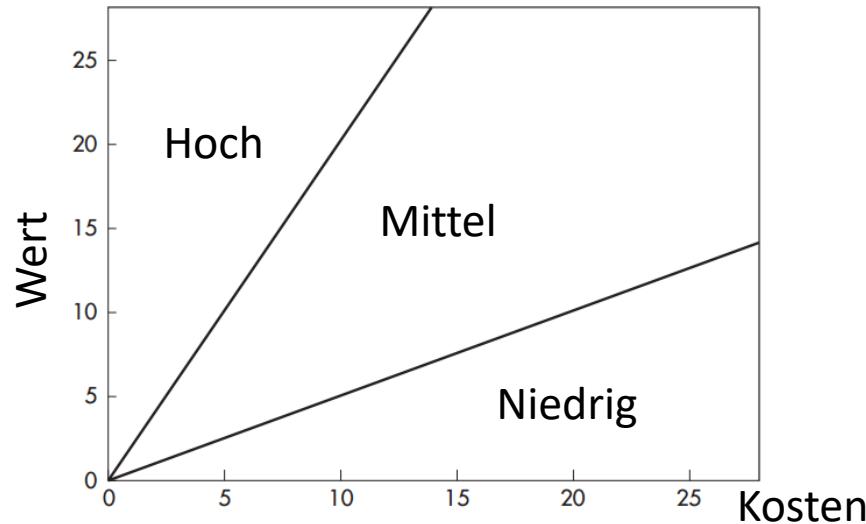
Bewertung der Kosten erfolgt auf gleiche Weise, wie die Bewertung des Werts der Requirements, nur auf der Basis einer paarweisen Vergleichsmatrix für die Herstellungskosten der Requirements, nicht deren Nutzen

Schritt 4: Konsistenz prüfen

Ebenfalls auf gleiche Weise wie beim Wert der Requirements wird auch für die Vergleichsmatrix der Kostenwerte die consistency ratio bestimmt

Schritt 5: Ergebnisse darstellen

- Die relativen Werte und Kosten jedes Requirements werden in einem Kosten-Nutzen-Diagramm eingetragen
- Typischerweise entlang der x-Achse die relativen Kosten, auf der y-Achse der relative Wert



Cost-Value-Approach: Fortsetzung Beispiel

Expertenbewertung lieferte folgende Vergleichsmatrix für die Kosten:

	Req1	Req2	Req3	Req4
Req1	1	7	1/2	4
Req2	1/7	1	1/5	2
Req3	2	5	1	5
Req4	1/4	1/2	1/5	1

Cost-Value-Approach: Fortsetzung Beispiel

Normierung der Spalten ergibt:

	Req1	Req2	Req3	Req4
Req1	0,29	0,52	0,26	0,33
Req2	0,04	0,07	0,11	0,39
Req3	0,59	0,37	0,53	0,42
Req4	0,07	0,04	0,11	0,08

Berechnung der Zeilensummen ergibt:

	Req1	Req2	Req3	Req4	Σ
Req1	0,29	0,52	0,26	0,33	1,41
Req2	0,04	0,07	0,11	0,39	0,39
Req3	0,59	0,37	0,53	0,42	1,90
Req4	0,07	0,04	0,11	0,08	0,30

Für den priority vector ergibt sich:

$$\frac{1}{4} \begin{pmatrix} 1,41 \\ 0,39 \\ 1,90 \\ 0,30 \end{pmatrix} = \begin{pmatrix} 0,35 \\ 0,10 \\ 0,48 \\ 0,07 \end{pmatrix}$$

Das heißt, in unserem Beispiel repräsentiert Req1 35% der gesamten Kosten der Requirements, Req2 10%, Req3 48% und Req4 7%

Cost-Value-Approach: Fortsetzung Beispiel

Für λ_{max} ergibt sich:

$$\begin{pmatrix} 1 & 7 & 1/2 & 4 \\ 1/7 & 1 & 1/5 & 2 \\ 2 & 5 & 1 & 5 \\ 1/4 & 1/2 & 1/5 & 1 \end{pmatrix} \begin{pmatrix} 0,35 \\ 0,10 \\ 0,48 \\ 0,07 \end{pmatrix} = \begin{pmatrix} 1,57 \\ 0,39 \\ 2,04 \\ 0,31 \end{pmatrix}$$

$$\begin{pmatrix} 1,57/0,35 \\ 0,39/0,10 \\ 2,04/0,48 \\ 0,31/0,07 \end{pmatrix} = \begin{pmatrix} 4,45 \\ 4,04 \\ 4,29 \\ 4,10 \end{pmatrix}$$

$$\lambda_{max} = \frac{4,45 + 4,04 + 4,29 + 4,10}{4} = 4,22$$

Für den consistency index ergibt sich:

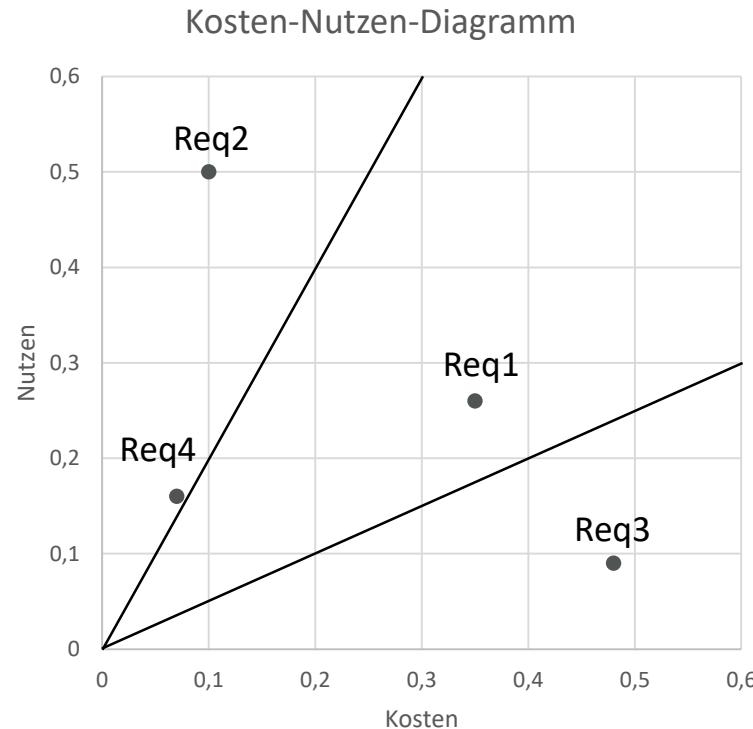
$$CI = \frac{4,22 - 4}{4 - 1} = 0,07$$

Für die consistency ratio ergibt sich:

$$CR = \frac{0,07}{0,90} = 0,08$$

Cost-Value-Approach: Fortsetzung Beispiel

Es entsteht folgendes Kosten-Nutzen-Diagramm:



Abschnitt 1.6

VOLERE REQUIREMENTS ENGINEERING PROZESS

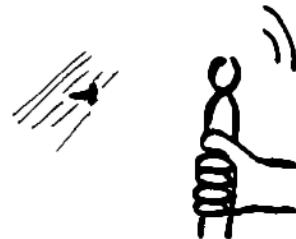
Was macht einen guten RE-Prozess aus?



Kundenorientierung



Vorgehen zielgerichtet und
methodisch klar

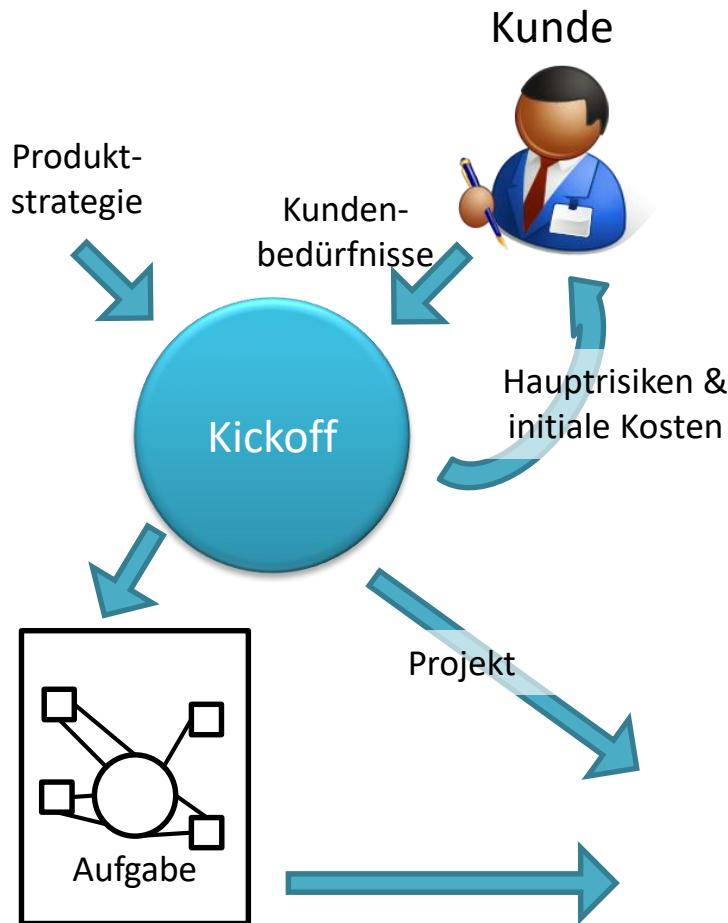


Einsatz adäquater Mittel



Integration von Erstellung und
Prüfung von Anforderungen

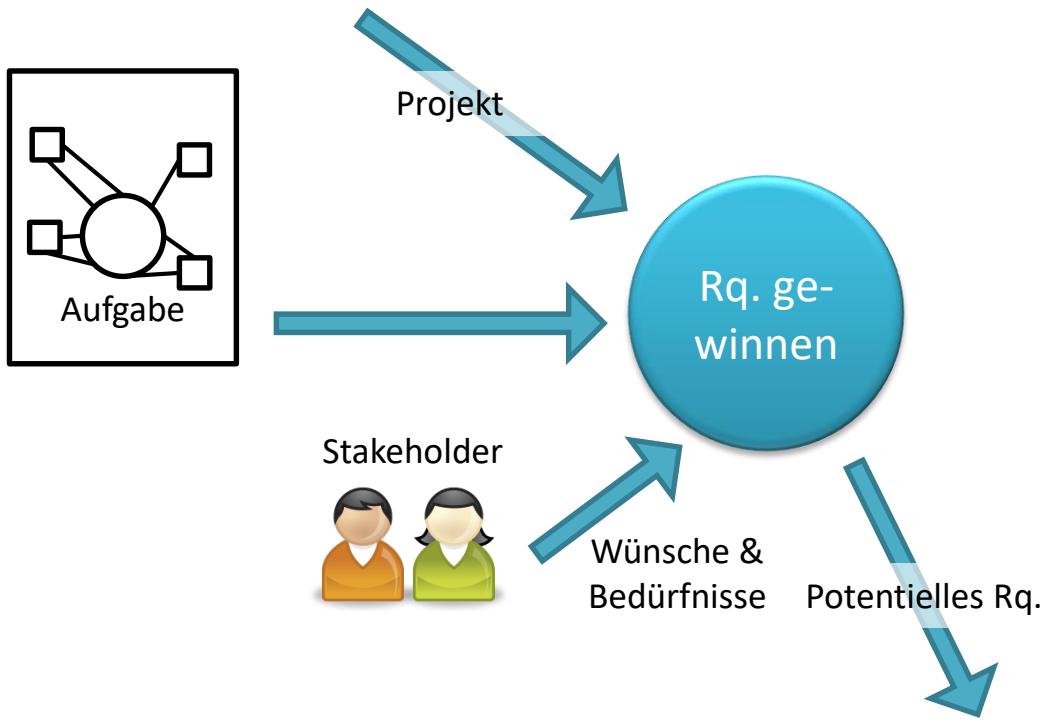
Das Volere Requirements Process Model I



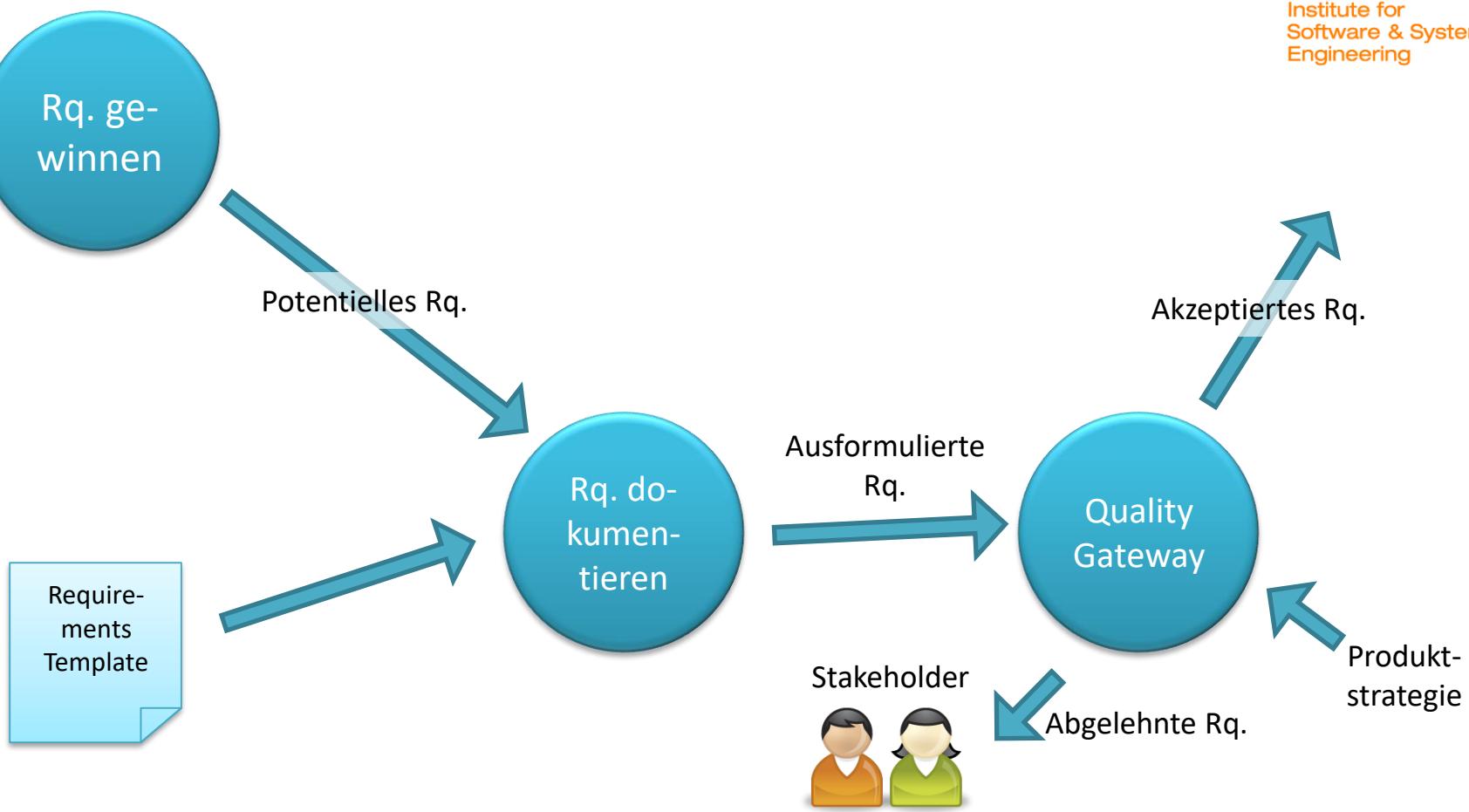
Volere Requirements Process: The Atlantic Systems Guild Ltd.

S. Robertson, J. Robertson: Mastering the Requirements Process, Addison-Wesley, 2006

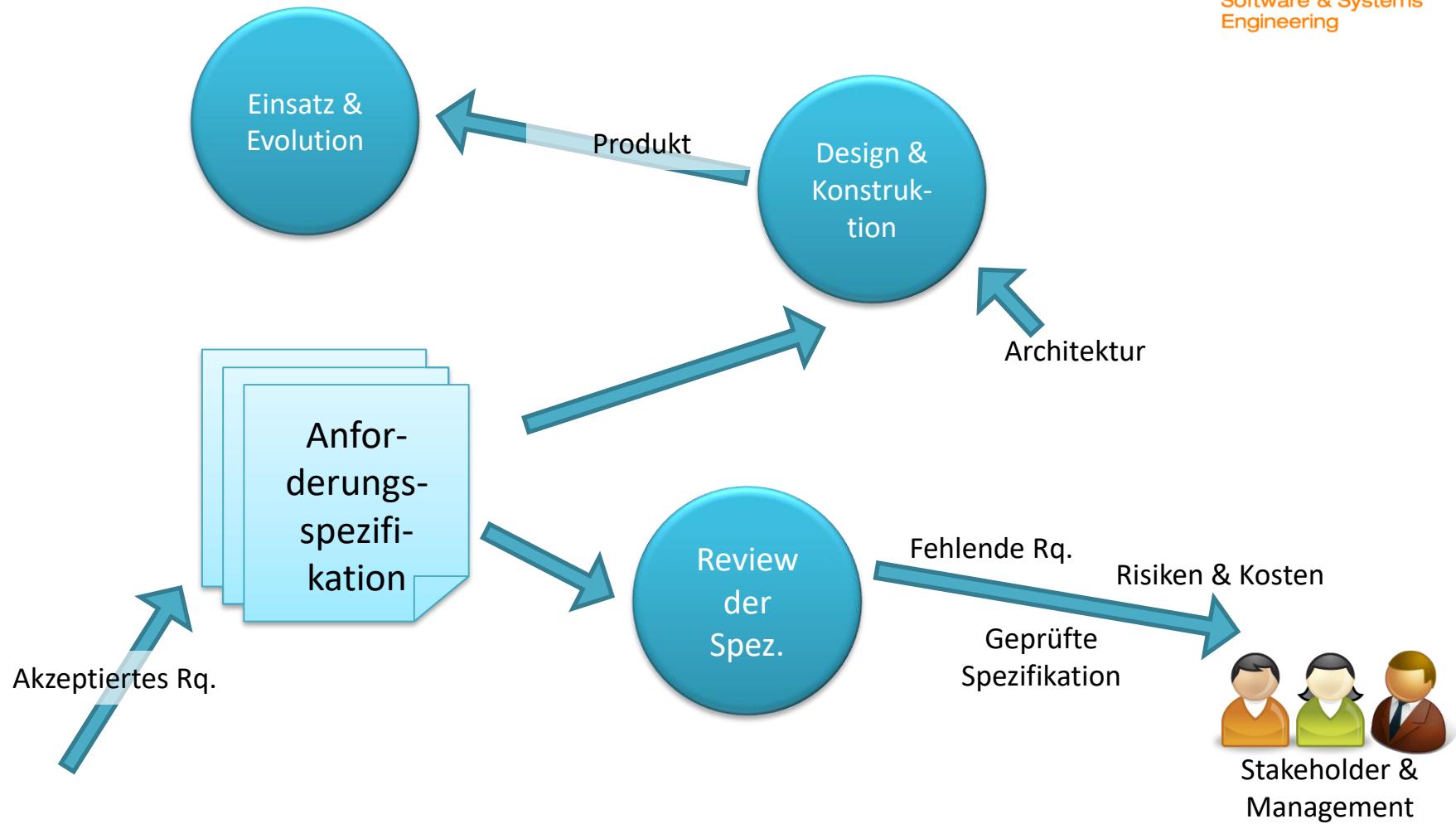
Das Volere Requirements Process Model II



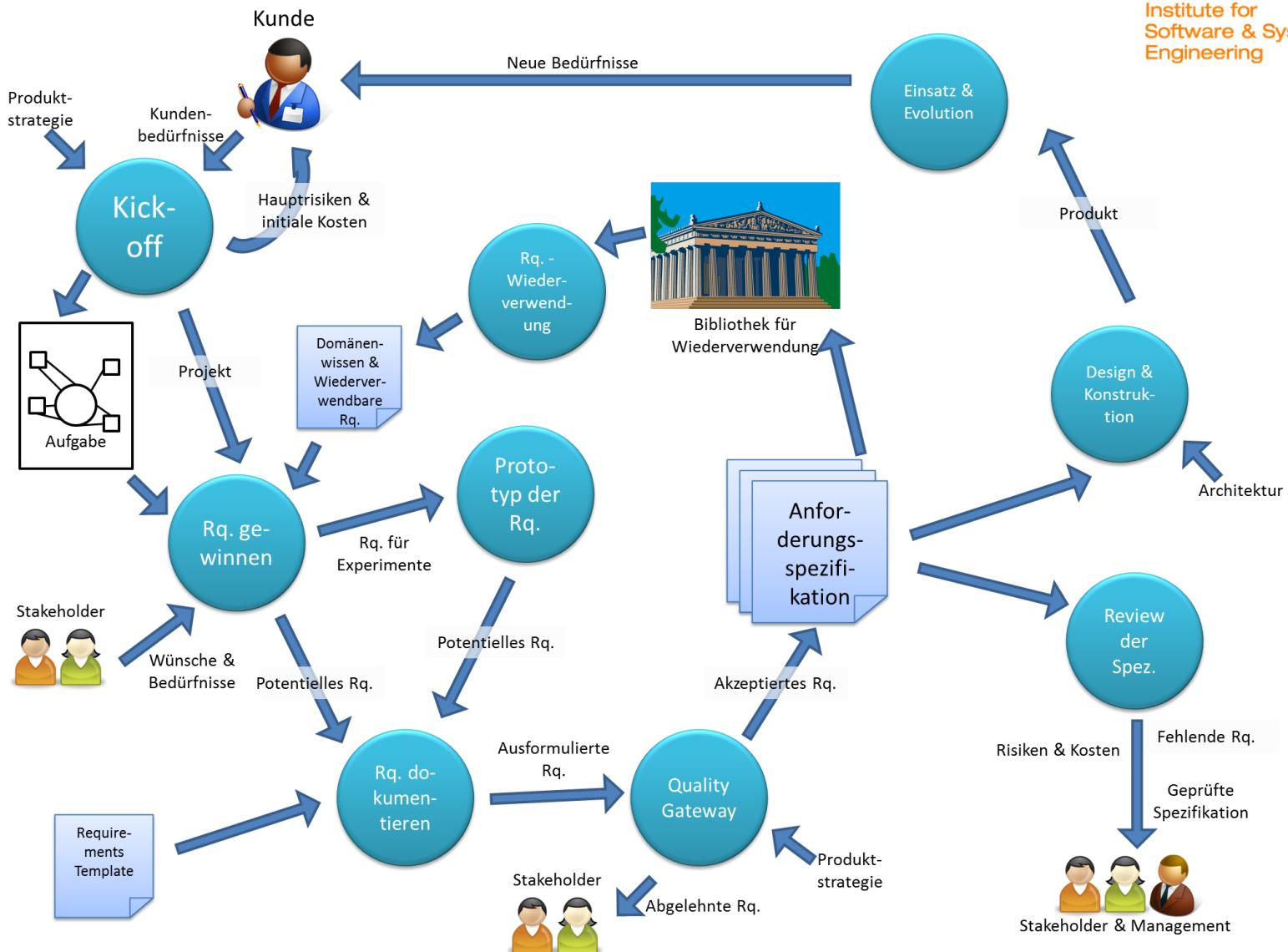
Das Volere Requirements Process Model III



Das Volere Requirements Process Model IV



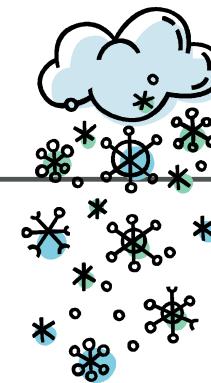
Das Volere Requirements Process Model



Abschnitt 1.7

PROJEKTSTART

Fallstudie „Eisfrei“



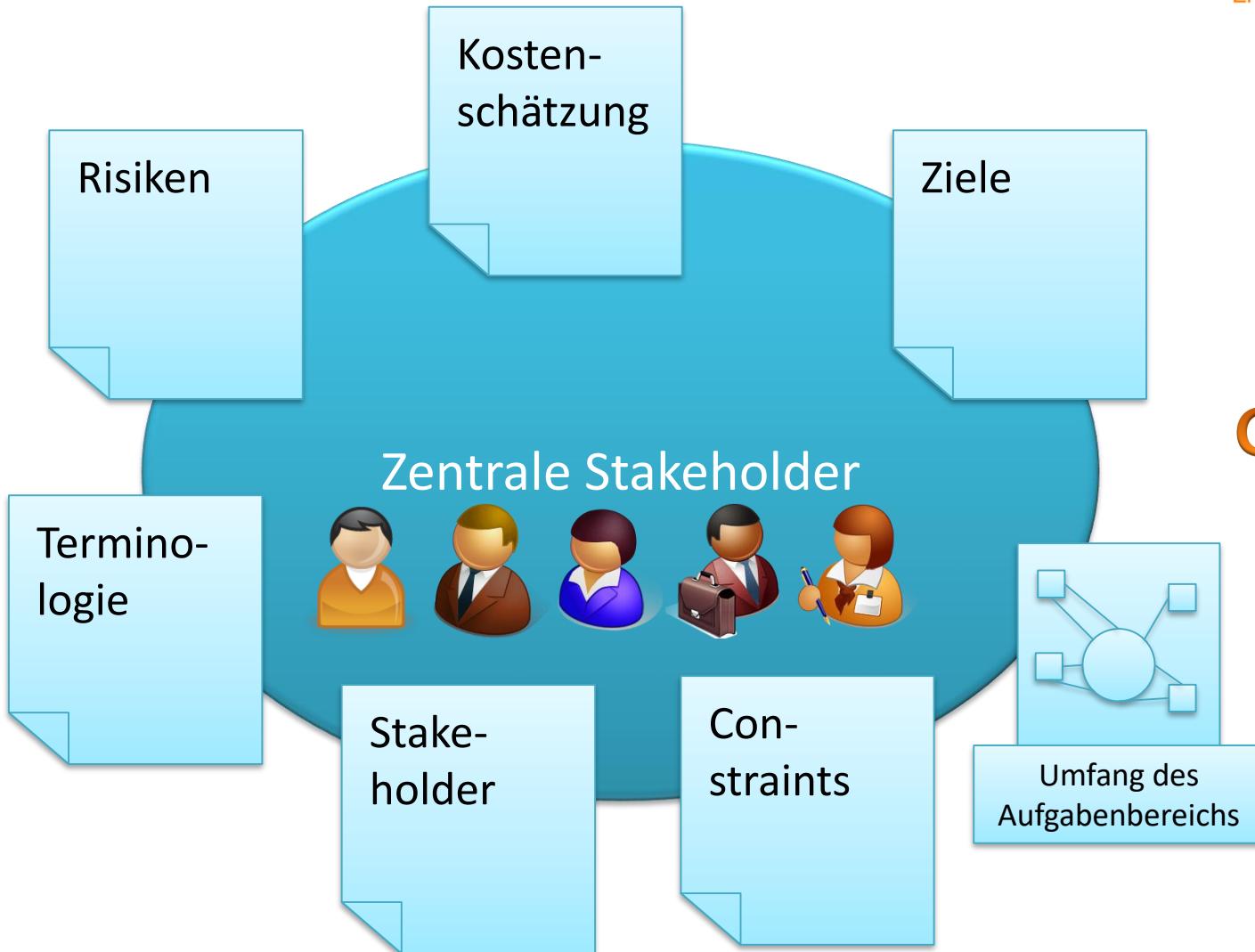
- Eisfrei ist ein System, das vorhersagt, wann sich auf welchen Straßen Eis bildet und Streufahrzeuge disponiert, um die betroffenen Straßen mit Auftausalz zu behandeln.
- Dies ermöglicht der Straßenmeisterei genauere Vorhersagen, präzisere Einsatzpläne und macht so die Straßen sicherer.



Projekt-Kickoff

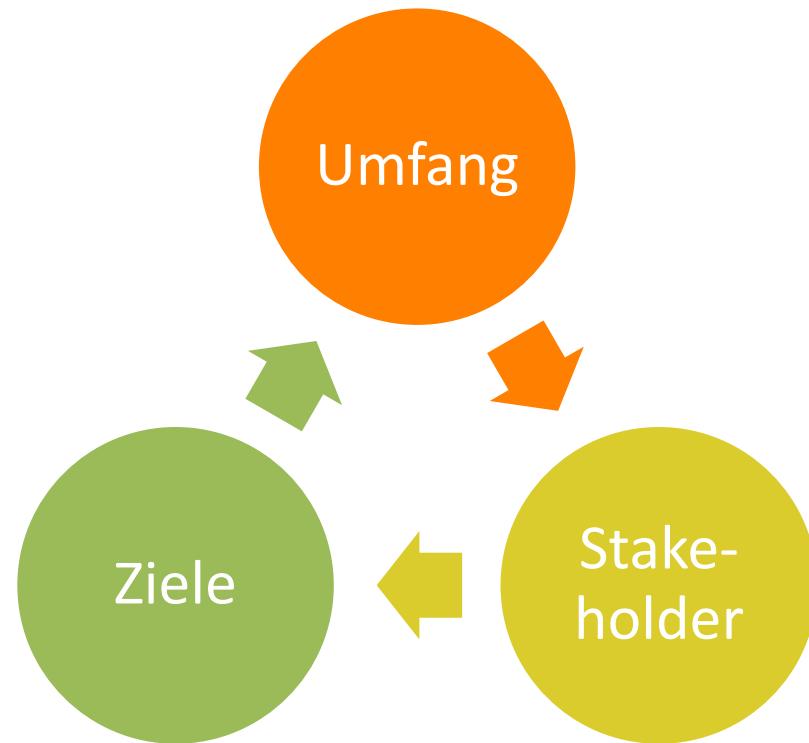
- Projekt vorbereiten und Durchführbarkeit sicherstellen
- Zusammenkunft der zentralen Stakeholder
- Es werden genügend Fakten gesammelt, um sicherzustellen, dass das Projekt
 - einen klar definierten Umfang hat,
 - ein lohnendes Ziel hat,
 - durchführbar ist und
 - die Unterstützung der Stakeholder hat.

Ergebnisse des Kickoff



Go/No go

- Drei Informationen sind entscheidend: Umfang, Stakeholder und Ziele



- Meistens beginnt man mit dem Umfang des Aufgabenbereichs

Abschnitt 1.8

KONTEXT





- Systeme stehen im Allgemeinen nicht allein. Sie sind **eingebettet** in eine (nicht-leere) Umgebung
- In der Umgebung befinden sich in der Regel **Akteure**, die für das System bedeutsam sind
- Die relevanten Akteure aus der Umgebung bilden den (**System-)**Kontext

Aufgabenbereich und Kontext bestimmen

- Durch das Festlegen des **Umfangs des Aufgabenbereichs** trennen wir
 - die Aufgabe, die wir studieren,
von
 - allen Aufgaben, die darum herum existieren.
- Die Grenze ergibt sich anhand von **Informationsflüssen**:
 - Fremde Information, die wir verarbeiten
 - Ergebnisse, die wir nicht selbst verarbeiten

Erster Schritt: Relevante Problemdomänen

- Domänen sind Fachgebiete
- Relevante Domänen sind Fachgebiete, über die wir etwas wissen müssen
- Zu Eisfrei erzählt uns der Kunde:

Straßen überfrieren im Winter und Eis auf der Straße verursacht Unfälle. Wir müssen in der Lage sein, vorherzusagen, wann sich Eis bildet, so dass ein Streufahrzeug eingeplant werden kann, um die Straße rechtzeitig zu behandeln. Wir erwarten, dass ein neues System genauere Vorhersagen der Eislage liefert. Dies ermöglicht zeitnäheres Anwenden von Auftaumaterial als aktuell möglich, wodurch Unfälle reduziert werden. Außerdem wollen wir willkürliche Behandlungen von Straßen eliminieren, welche Auftausalz verschwenden und Umweltschäden verursachen.

Schritt zwei: Angrenzende Systeme in den relevanten Problemdomänen

- Alle relevanten Problemdomänen zu einem Teil unserer Aufgabe zu machen, ist nicht zielführend (und oft einfach unmöglich)
- Stattdessen suchen wir in jeder Problemdomäne nach physischen **Entitäten** (Personen oder Institutionen/Systeme) die **Teilaspekte der Problemdomäne** repräsentieren
- Diese Entitäten sind **potentielle angrenzende Systeme**
- Für jeden Kandidaten ist zu prüfen, ob er Teil unseres Aufgabenbereichs ist, oder Teil einer anderen Aufgabe

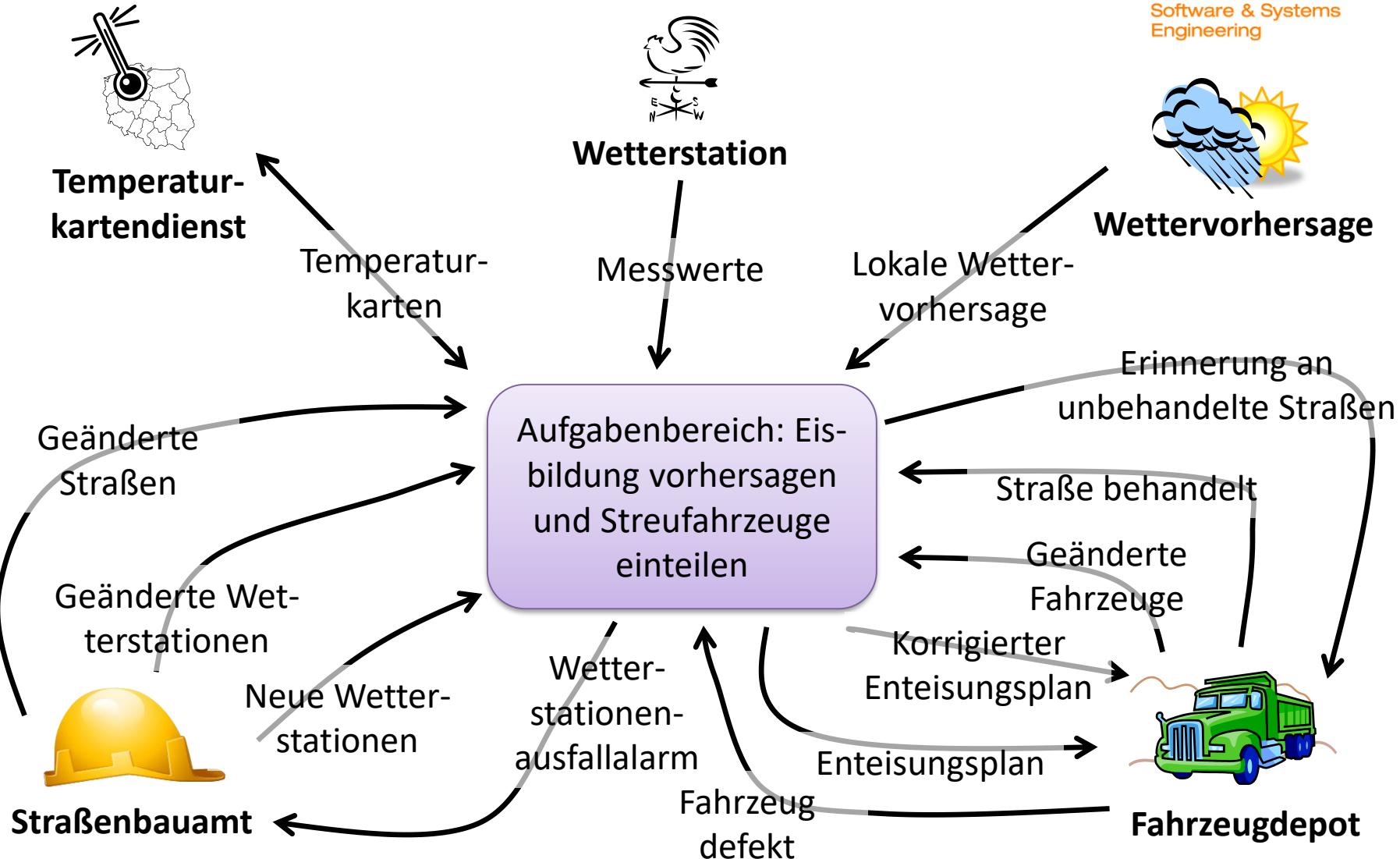
Angrenzende Systeme bei Eisfrei I

Für Eisfrei relevant ist das Wetter. Wolken und Regen sind physische Entitäten dieser Domäne, mit denen wollen wir uns aber nicht auseinandersetzen, da zu weit weg von unserem Aufgabenbereich. Stattdessen gibt es einen [Wetterdienst](#), der uns mit Informationen zum Wetter versorgen kann. Die Wettervorhersage ist aber nicht präzise genug, um Eisbildung adäquat vorherzusagen. [Wetterstationen](#), liefern solche genaueren Daten und können diese an Eisfrei übermitteln. Wetterstationen sind aber teuer, daher kann die Stadt Ulm nicht tausende von ihnen aufstellen. Von der Straßenmeisterei Ulm erfahren wir, dass [Anbieter von Temperaturkarten](#) Informationen über die Temperaturunterschiede auf jedem Meter Straße zwischen den Messstationen liefern können.

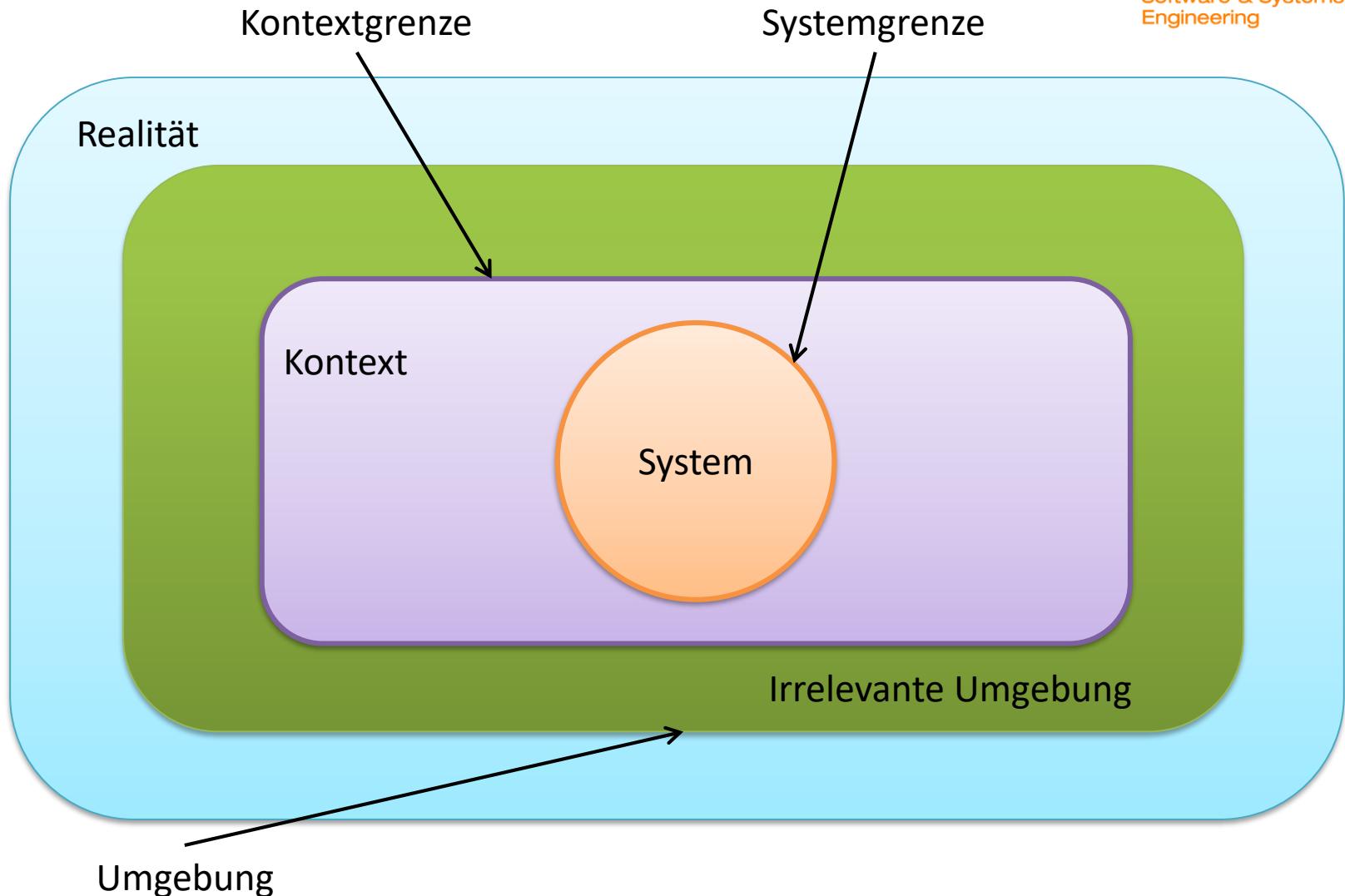
Aus den weiteren relevanten Problemdomänen finden wir weitere angrenzende Systeme:

- Die Domäne Straßen wird für uns durch das **Straßenbauamt** repräsentiert. Dieses baut und wartet Straßen und kann uns Informationen über Straßen liefern.
- Von Vertretern des Kunden erfahren wir außerdem, dass das **Fahrzeugdepot** Streufahrzeuge wartet und als Basis für deren Einsatz dient. Dies ist für uns ein angrenzendes System, das die Domäne Fahrzeuge repräsentiert.

Erster Entwurf eines Kontextdiagramms



Kontext- und Systemgrenze



Realität

Alle materiellen oder immateriellen Dinge der Welt.

Umgebung

Der Teil der Realität, der auf ein in diesem Bereich eingesetztes System Einfluss haben könnte.

Kontext

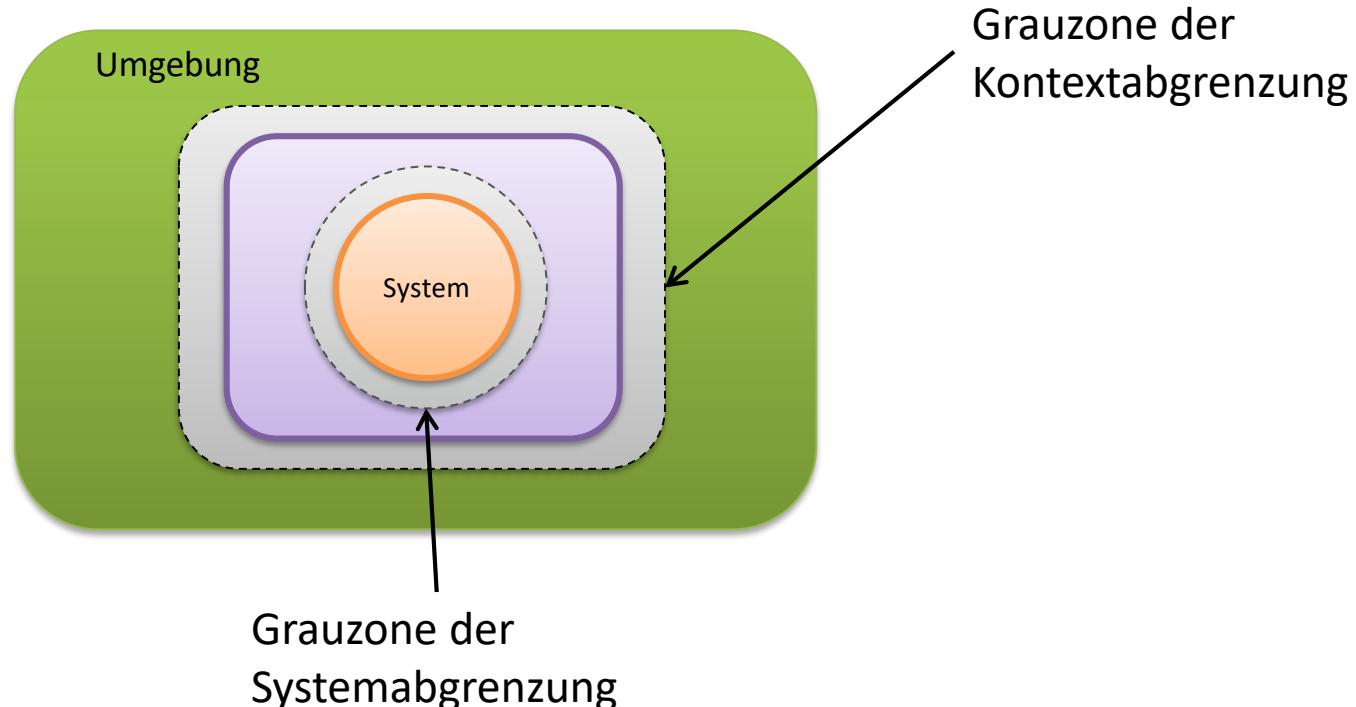
1. Der umgebende Teil einer sprachlichen Einheit. 2. Der inhaltliche (Gedanken-, Sinn)zusammenhang, in dem eine Äußerung steht. 3. (In der Informatik) Die Komponenten eines Anwendungsbereichs (Umgebung), die mit einem System interagieren, aber selbst nicht Teil des Systems sind.

Akteur

Ein Mensch, der Ziele hat und diese verfolgt oder ein Element der Umgebung, das zur Erreichung eines Ziels dient und hierzu handelt oder Informationen verarbeitet.

Dudenverlag: Duden Das Fremdwörterbuch, 2007

Kontext- und Systemgrenze können sich im Laufe des Requirements Engineering verschieben



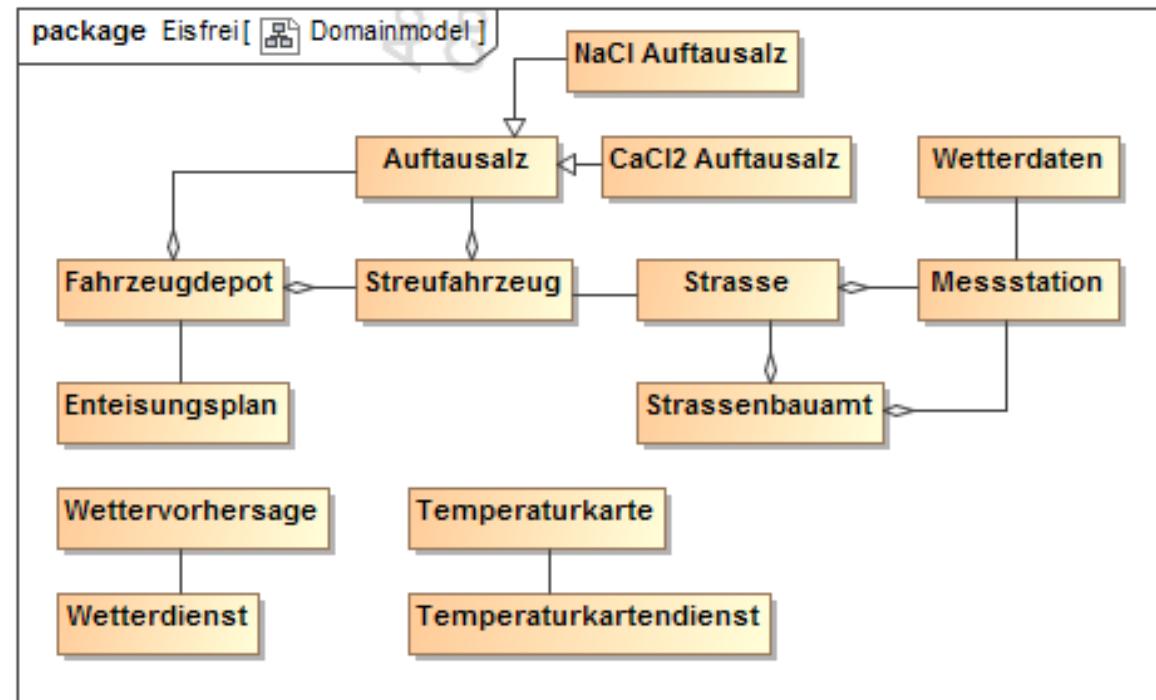
Darstellung des Systems in seinem **Kontext**

- **Andere Betrachtungsebene**
- In der Regel wird das System als **Black-Box** gesehen (keine Interna)
- **Akteure**, die direkt mit dem System interagieren, modellieren
- **Interaktionen** zwischen den Akteuren und dem System modellieren
- **Interaktionen** von Akteuren untereinander modellieren
- Graphische Darstellung

- Entscheidend für gemeinsames Verständnis der Anforderungen durch alle Beteiligten: **Gemeinsames Vokabular**
- Eindeutige Bedeutung für alle Begriffe der Problemdomäne festlegen und dokumentieren
- Graphisch modellieren als **Domänenklassendiagramm**, Detailbeschreibungen der Domänenentitäten im **Glossar**

Domänenklassendiagramm

- Keine Methoden zu den Klassen modellieren
- Fokus auf Beziehungen zwischen den Domänenklassen: Assoziation und Generalisierung



- Alle Dokumente und Informationen mit **Nominalphrasenmethode** untersuchen
- Nomen und Nominalphrasen ergeben Kandidaten für Domänenklassen bzw. deren Attribute
- Duplikate eliminieren
- Unspezifische Begriffe, unnötige Details, usw. eliminieren; ggf. Domänenklassen umbenennen

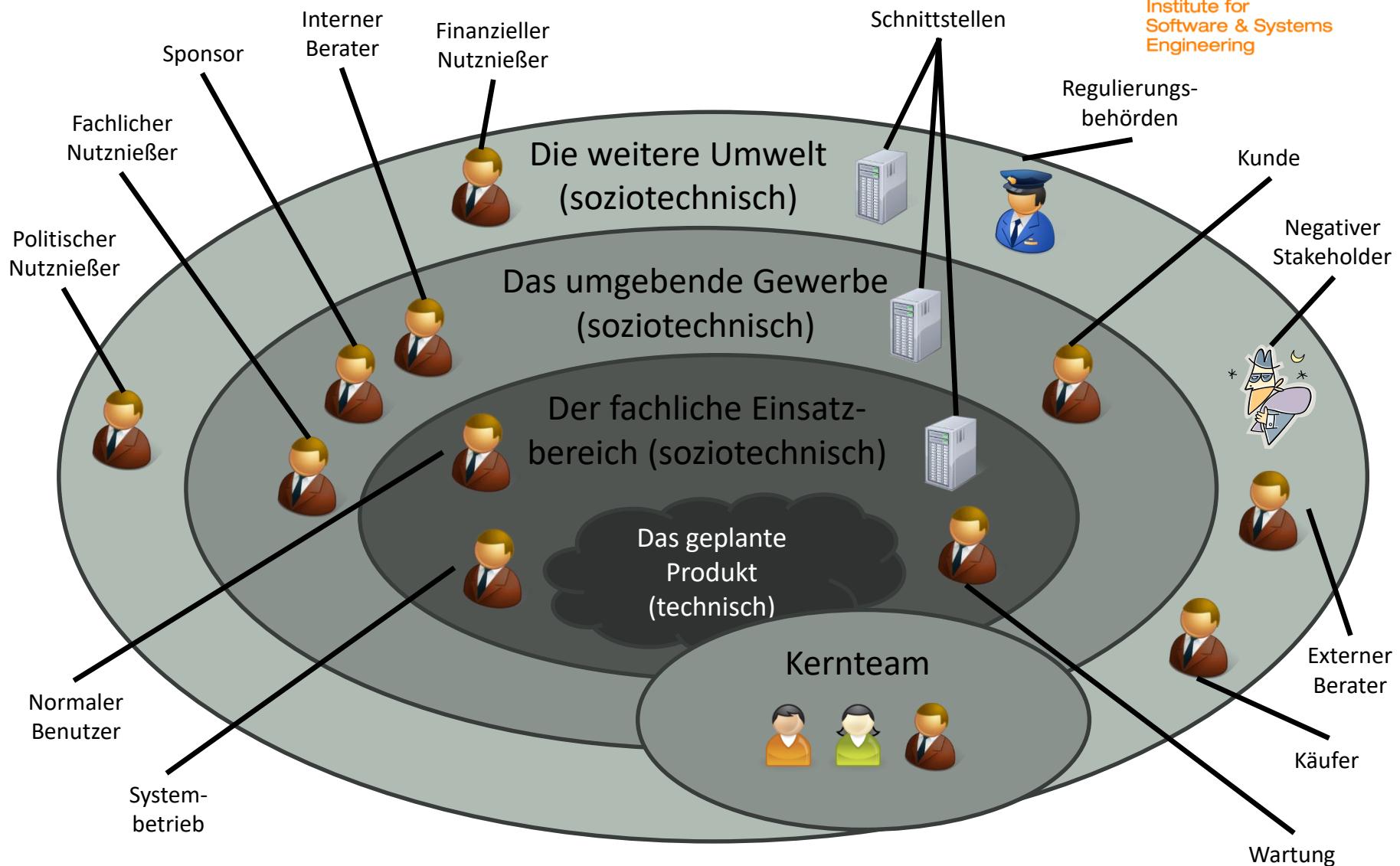
Abschnitt 1.9

STAKEHOLDER UND PROJEKTZIELE

Akteure und Ziele als Quellen für Anforderungen

- Am Anfang steht das Finden von **Akteuren** im Umfeld des Systems, ihren **Zielen** und Handlungen zur Erreichung ihrer Ziele
- Die Akteure, die
 - Aufgaben an das System delegieren
 - dem System Informationen liefern
 - Informationen vom System erhaltenbilden den **Systemkontext**
- Die eigentlichen **Anforderungen** beschreiben, welche Dienstleistungen das System den Akteuren im Kontext anbietet, welche Informationen es benötigt und welche Bedingungen erfüllt sein müssen

Stakeholder identifizieren



- Kunde (auch Sponsor, Auftraggeber): Zahlt für das Projekt. Profitiert vom Produkt, muss letztendlich zufrieden gestellt werden.
- Käufer: Kauft am Ende das Produkt (nicht immer zutreffend)
- Benutzer: Benutzen das fertige System. Wesentlich für das Verständnis der fachlichen Aufgabe.
Wichtig:
 - Fachwissen
 - Technische Erfahrung
 - Intellektuelle Fähigkeiten
 - Einstellung zur Arbeit
 - Bildung
 - Sprachfähigkeit

Der Kunde für das Produkt ist Karl Müller, Geschäftsführer von Salztechnik Systeme. Letztendlich will der Kunde das Produkt an Käufer aus anderen Ländern verkaufen. Der Kunde hat bestätigt, dass er die Entscheidungsgewalt für Änderungen am Produktumfang hat.

Der (erste) Käufer für das Produkt ist die Straßenmeisterei Ulm im Alb-Donau-Kreis, vertreten durch Hauptstraßenmeister Jürgen Meier.

Ziele: Was soll erreicht werden?

- Das Projektziel ist eine übergreifende Anforderung und hat höchste Priorität
- Das Projektziel fokussiert die Anforderungsgewinnung
- Jede spätere Anforderung muss dem Gesamtziel dienen
- Zu Beginn ist das Ziel oft vage oder sehr allgemein formuliert. Ziel beim Kickoff ist die Konkretisierung

Schritt 1: Problembeschreibung

Zuerst wird eine kurze Darstellung des Problems oder des Projekthintergrunds formuliert

Auf Straßen bildet sich im Winter Eis, was zu Unfällen führt, bei denen Menschen ihr Leben verlieren. Wir müssen fähig sein, vorherzusagen, wann sich auf einer Straßen wahrscheinlich Eis bildet, so dass unser Fahrzeugpark rechtzeitig ein Streufahrzeug einteilen kann, um die Straße am Überfrieren zu hindern. Wir erwarten, dass ein neues System durch den Einsatz von Temperaturkarten des Landkreises und durch Temperaturdaten von Wetterstationen an den Straßen als Ergänzung zu Wettervorhersagen genauere Vorhersagen der Eisbildung liefert. Dies ermöglicht eine zeitnahere Behandlung mit Auftausalz, als aktuell möglich, was Unfälle reduziert. Wir wollen außerdem überflüssige Einteisungsbehandlungen vermeiden, da diese Auftausalz verschwenden und Umweltschäden verursachen.

Schritt 2: Projektziel, Nutzen, Erfolgsmaß

- Als Projektziel formuliert man nun, was den größten Beitrag zur Verbesserung der Problemsituation bringt

Projektziel: Präzise Vorhersage des Zeitpunkts, zu dem sich auf einer Straße Eis bildet und Einplanen einer Enteisung.

- Projektziel ist aber nicht nur die Problemlösung sondern das bieten eines Geschäftsvorteils/-nutzens

Nutzen: Reduzierung der Unfälle durch Verhinderung von vereisten Straßen.

Schritt 2: Projektziel, Nutzen, Erfolgsmaß

- Der Geschäftsnutzen sollte messbar sein. So kann der Projekterfolg später beurteilt werden.

Erfolgsmaß: Unfälle, die auf vereiste Straßen zurückgeführt werden, sollen höchstens 15 Prozent aller Unfälle im Winter ausmachen.

- Verbleibende Fragen:
 - Ist das Projektziel vernünftig und angemessen
 - Ist das Projektziel erreichbar
 - Ist das Projektziel realisierbar

Projektziel: Einsparung bei der Straßenwartung im Winter.

Nutzen: Reduzierte Kosten für Enteisungen und Straßenreparaturen.

Erfolgsmaß: Die Kosten für den Streudienst sollen um 25 Prozent des Iststands reduziert werden und Eisschäden an Straßen sollen um 50 Prozent reduziert werden.

Die präzisere Vorhersage der Eisbildung (Ziel 1) ermöglicht es, dass weniger Auftausalz ausgebracht werden muss. Es ergibt sich ein weiterer Nutzen:

Nutzen: Umweltschäden durch unnötig ausgebrachtes Auftausalz reduzieren.

Erfolgsmaß: Die für die Enteisung der Straßen im Bereich der Straßenmeisterei benötigte Menge Auftausalz um 50 Prozent reduzieren.

Relevante Literatur: Thornes, J.E.: Salt of the Earth. Surveyor Magazine, 08.12.1994, S. 16ff

Abschnitt 1.10

ANFORDERUNGEN FINDEN UND DOKUMENTIEREN

Wo kommen denn die Anforderungen her?

- Verschiedene Informationsquellen
- In der Regel viele Beteiligte
- Beteiligte und deren **Ziele** kennen



Informationsquellen



- Welche **Beteiligten** gibt es, von denen Anforderungen kommen können?
- Häufig nicht **Individuen**, sondern **Rollen**
- Typische Rollen: Endbenutzer, Auftraggeber, Betreiber, Entwickler, Projektleitung,...
- Verschiedene **Befragungs- / Interviewtechniken**



- **Wie wird heute gearbeitet?**
- Was ist gut und was soll anders werden?
- Wer verwendet wann wo welche Daten?



- **Ausschreibungstexte, Visionsdokumente, Unternehmensziele**

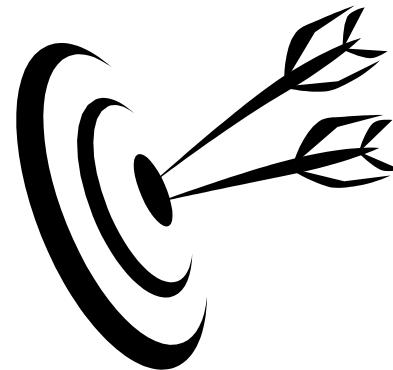
- Betroffene / Beteiligte (Stakeholder, Interessenvertreter) sind die Schlüsselfiguren im Requirements Engineering
- Nicht nur der „Kunde“ im Sinne des Auftraggebers
- Relevante Fragen:
 - Wer hat in welcher Rolle mit dem zu erstellenden System zu tun?
 - Wer kann/soll/darf/muss Anforderungen an das System stellen?
 - Wer stellt Anforderungen an den Projektlauf?
 - Wer kann/soll/darf/muss Randbedingungen vorgeben?



„If you don't know where you're going,
you're unlikely to end up there.“

— Forrest Gump

- Übergeordnete Ziele (Geschäftsziele, Vision) des Vorhabens?
- Welchen Nutzen zieht welcher Beteiligte aus der Erreichung welches Ziels?
- Wie wird die Zielerreichung festgestellt?
 - Abnahmebedingungen
 - Quantifizierbare, messbare Ziele
- Gibt es Zielkonflikte?
- Etwa drei bis sieben übergeordnete Ziele formulieren



- Ziele und Bedürfnisse der Beteiligten erkennen, analysieren und darstellen
- Den Beteiligten Möglichkeiten aufzeigen, wenn diese sie selbst nicht erkennen
- Wo adäquat, zunächst den IST-Zustand erheben
- Bei Produktentwicklung das Marktpotential klären
- Randbedingungen erkennen, analysieren und dokumentieren
- Techniken zur Informationsbeschaffung und Analysemethoden gibt es viele. Es gilt, die für die Situation geeignetsten auszuwählen

Das wichtigste Requirements Engineering-Werkzeug
überhaupt sind die Ohren!

*„You have two ears and one mouth. I suggest that you
use them in that proportion.“*

— Gilbert Keith Chesterton

Abschnitt 1.10.1

ERMITTLUNGSTECHNIKEN

Techniken zur Informationsbeschaffung

Übersicht

Form	Eignung für			
	Wünsche ausdrücken	Möglichkeiten aufzeigen	IST-Zustand erheben	Marktpotenzial klären
Interviews	↑	↓	↑	→
Beobachtung der Benutzer	→	↓	↑	→
Rollenspiele	↑	→	→	↓
Beispiele analysieren	→	↓	↑	↓
Staffagen und Prototypen	→	↑	↓	→
Umfrage / Fragebogen	→	↓	↑	↑
Gemeinsame Arbeitstagungen	↑	→	→	↓
Marktstudien	↓	↓	→	↑
Problemmeldungsauswertung	↑	↓	↓	→
Vergleich mit anderen	→	↑	↓	↑

Erster Schritt bei der Auswahl der Ermittlungstechniken ist die Bestimmung der Risikofaktoren der Anforderungsermittlung

- Menschliche Faktoren
 - Erfahrung von Requirements Engineer und Stakeholder mit der Ermittlungstechnik
 - Soziale, gruppendynamische und kognitive Fähigkeiten der Stakeholder
 - Explizit bewusstes oder selbstverständliches/implizites Wissen gesucht
- Organisatorisches
 - Festpreis- oder Werkvertrag, Neuentwicklung oder Weiterentwicklung, räumlich und zeitliche Verfügbarkeit der Stakeholder
- Fachlich-inhaltliche Einflüsse
 - Bei hoher fachlicher Komplexität empfiehlt sich ein strukturierender Ansatz, beispielsweise die bereits bekannten Business Use-Cases

Abschnitt 1.10.2.1

HERAUSFORDERUNG KOMMUNIKATION

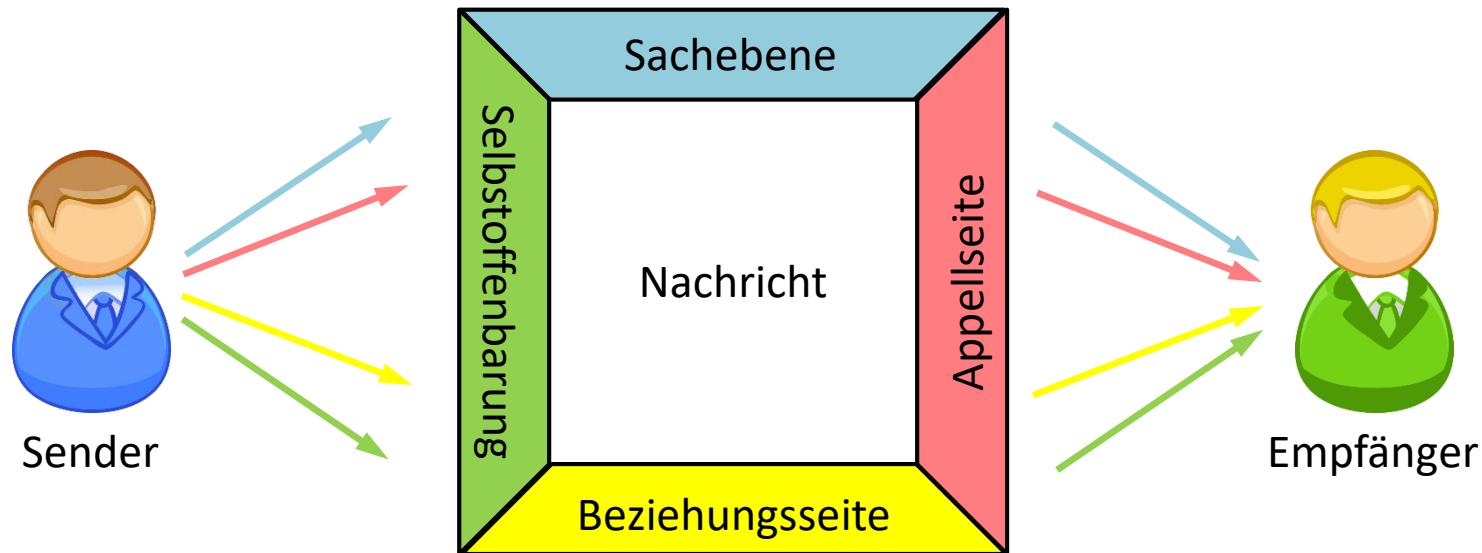
Warum ist Kommunikation schwierig



„Woher soll ich wissen, was Du meinst,
wenn ich höre, was Du sagst?“
— Quelle unbekannt

Vier-Seiten-Modell

- **Vier-Seiten-Modell** (Kommunikationsquadrat oder Vier-Ohren-Modell) nach Friedemann Schulz von Thun
- Modell aus der Kommunikationspsychologie
- Jede Nachricht hat vier Seiten: **Sachinhalt**, **Selbstoffenbarung**, **Beziehung** und **Appell**
- Sender und Empfänger einer Nachricht können die vier Ebenen unterschiedlich gewichten. Das führt zu Missverständnissen



Die vier Seiten einer Nachricht:

- Sachinhalt: Auf die Sache bezogener Aspekt, die beschriebene Sache
- Selbstoffenbarung: Auf den Sprecher bezogener Aspekt, das, was anhand der Nachricht über den Sprecher erkennbar wird
- Beziehungsseite: Auf die Beziehung bezogener Aspekt, was an der Art der Nachricht über den Anderen oder über die Beziehung zum Anderen offenbar wird
- Appell: Auf die beabsichtigte Wirkung bezogener Aspekt, wozu soll der Empfänger veranlasst werden

Beispiel

Durch Missverständnisse gestörte Kommunikation

Ein Mann und eine Frau sitzen beim Abendessen. Der Mann sieht Kapern in der Suppe, das Gespräch beginnt.



Was ist das Grüne
in der Suppe?

Sachebene: *Da ist was Grünes.*
Selbstoffenbarung: *Ich weiß nicht, was es ist.*
Beziehung: *Du wirst es wissen.*
Appell: *Sag mir, was es ist!*

A simple cartoon illustration of a woman with dark hair pulled back and a white collared shirt underneath a green sweater. A grey speech bubble originates from her mouth.

Mein Gott, wenn es dir
hier nicht schmeckt,
kannst du ja woanders
essen gehen!

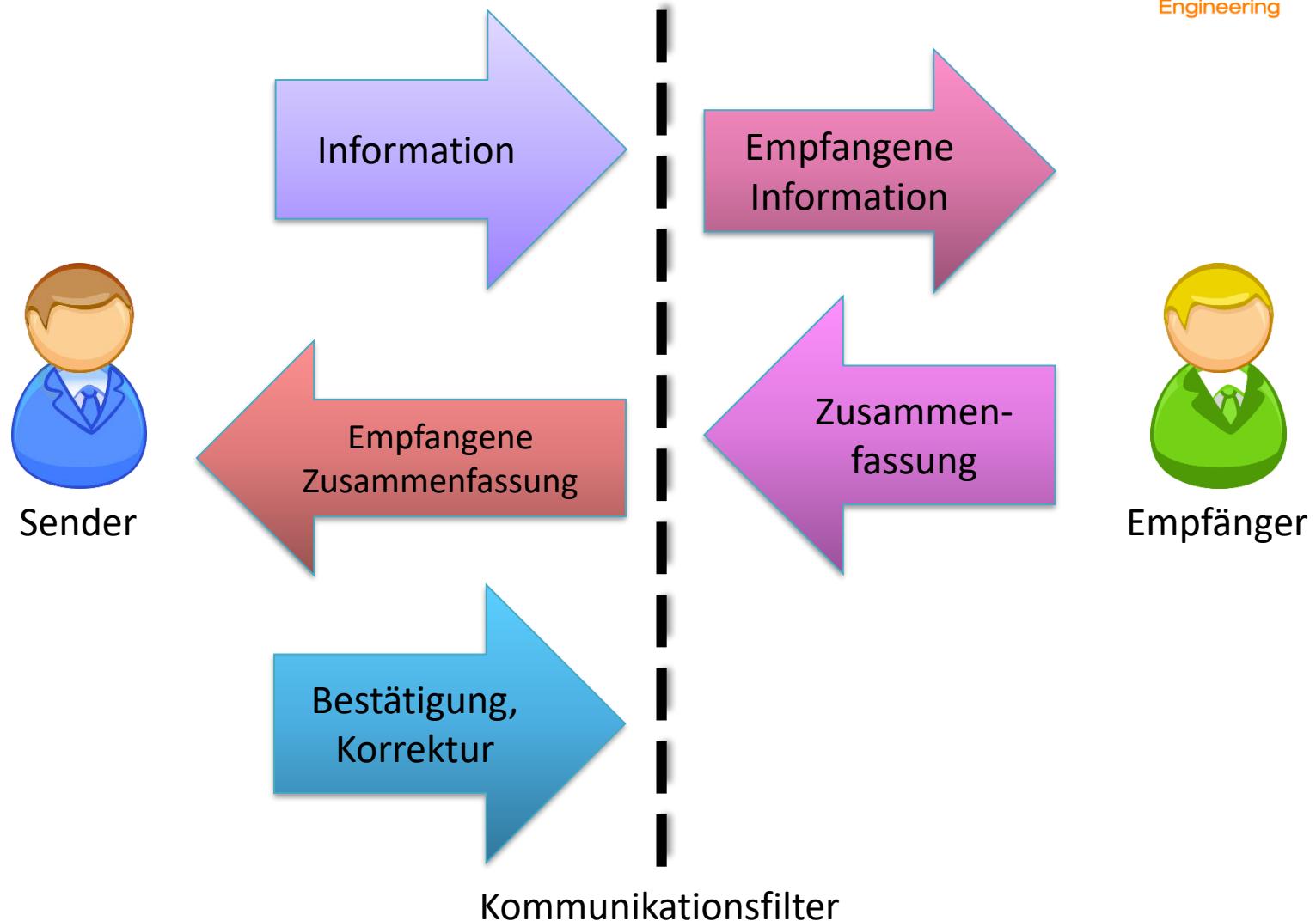
Sachebene: *Da ist was Grünes.*
Selbstoffenbarung: *Mir schmeckt das nicht.*
Beziehung: *Du bist eine miese Köchin!*
Appell: *Lass nächstes Mal das Grüne weg!*

F. Schulz von Thun: Miteinander reden 1: Störungen und Klärungen, rororo, 2010

- Geht auf den Psychologen Dr. Carl Rogers zurück
- Ziele: **Einfühlen** in den Gesprächspartner, **Mitdenken** des Gesprächs, **Aufmerksamkeit** für und **Interesse** am Gesprächspartner
- Hilft Missverständnisse und Kommunikationsverluste zu vermeiden
- Werkzeuge: Paraphrasieren, Ich-Botschaften, geregeltes Feedback

1. Aufnehmendes Zuhören
 - Blickkontakt
 - Aufmerksamkeitslaute
 - Spiegeln der Emotionen des Gesprächspartners
2. Wiederholen mit eigenen Worten
 - Empfänger wiederholt mit eigenen Worten die empfangenen Informationen
 - Am besten in Frageform
3. Verbalisieren mit eigenen Worten
 - Empfänger gibt mit seinen Worten die vermutlichen Stimmungen/Gefühle des Gesprächspartners wieder
 - Nur wenn möglich und erforderlich

Aktives Zuhören: Feedbackschleife



Abschnitt 1.10.2.2

BEFRAGUNGSTECHNIKEN



Interview

- Requirements Engineer und einer oder mehrere Stakeholder
- Vorgegebene Fragen (Katalog), Antworten werden protokolliert
- Geschicktes Nachfragen, um Vollständigkeit zu erreichen und unbewusste Anforderungen aufzudecken
- Hoher Zeitaufwand

Fragebogen

- Offene und geschlossene (z.B. multiple-choice)
- Viele Informationen in kurzer Zeit und zu geringen Kosten
- Vorformulierte Antworten helfen Stakeholdern, die ihr Wissen nicht formulieren können
- Fragt nur nach Dingen, die der Requirements Engineer bereits weiß oder vermutet
- Erstellung eines guten Fragebogens kostet Zeit und erforderte fundiertes Domänenwissen sowie Kenntnisse der Psychologie der Fragebogenerstellung
- Keine unmittelbare Rückkopplung

Abschnitt 1.10.2.3

KREATIVITÄTSTECHNIKEN



Brainstorming

- Gruppe mit ca. 5 bis 10 Personen
- Zu Anfang ungefilterte Sammlung von Ideen für eine vorgegebene Zeit
- Ideen anderer aufgreifen und weiterentwickeln
- Anschließend sorgfältigere Analyse der Ideen
- Beteiligung vieler Stakeholder aus unterschiedlichen Gruppen erhöht Effektivität

Brainstorming paradox

- Brainstorming wird modifiziert
- Es werden Ergebnisse gesammelt, die **nicht** erreicht werden sollen
- Dann werden Maßnahmen entwickelt, wie die negativen Ergebnisse vermieden werden können
- Hilft Risiken zu erkennen und Gegenmaßnahmen zu entwickeln



Perspektivenwechsel

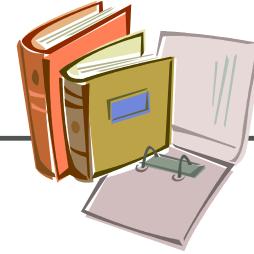
- Perspektivwechsel heißt, das Problem aus unterschiedlichen, klar definierten Blickwinkeln zu betrachten
- Alle Teilnehmer nehmen über die Zeit verschiedene Extrempositionen ein
- Animiert auch Stakeholder, die von ihrer Position überzeugt sind, zu neuen Betrachtungsweisen

Analogietechnik

- Ideen gewinnen durch Brückenschlag aus einer anderen Domäne
- Vertreter sind **Bionik** und **Bisoziation**
- Bionik: Suchen nach ähnlichen Problemen (und Lösungen) in der Natur
- Bisoziation: Analogien auch außerhalb der Natur möglich
- Nötig: Fähigkeit zum analogen Denken, viel Zeit und tiefe Kenntnis des Bereichs, aus dem die Analogie gezogen werden soll

Abschnitt 1.10.2.4

DOKUMENTENORIENTIERTE TECHNIKEN

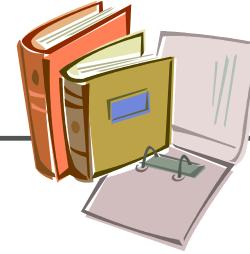


Systemarchäologie

- Informationen aus der Dokumentation oder der Implementierung eines Alt- oder Konkurrenzsystems gewinnen
- Interessant, wenn das Wissen über die implementierte Fachlogik verloren ist
- Erzeugt viele, sehr detaillierte Anforderungen, ist aufwendig
- Einzige Möglichkeit zum vollständigen Transfer der Funktionalität von alt nach neu

Perspektivenbasiertes Lesen

- Text wird aus einer spezifischen Perspektive (z.B. Tester oder Realisierer) gelesen
- Für die Perspektive irrelevante Inhalte werden ignoriert
- Auf speziellen Bereich fokussierte Analyse bestehender Dokumente

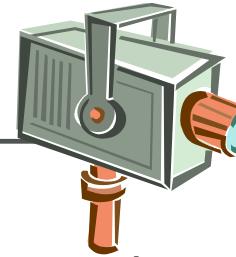


Wiederverwendung

- Einmal erhobene und auf entsprechendem qualitativen Stand dokumentierte Anforderungen können wieder verwendet werden
- Ideal ist die Speicherung der Anforderungen in einer Datenbank mit entsprechenden Sichten und Suchmöglichkeiten

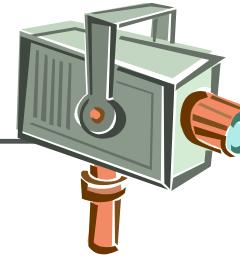
Abschnitt 1.10.2.5

BEOBACHTUNGSTECHNIKEN



- Gut in Situationen, in denen die Fachleute ihr Wissen nicht formulieren können oder keine Zeit haben, dieses an den Requirements Engineer weiterzugeben
- Beobachten der Stakeholder bei ihrer Arbeit, Arbeitsschritte dokumentieren
- Daraus zu unterstützende Arbeitsschritte, mögliche Fehler, Risiken und Fragen ermitteln
- Stakeholder können ihr Wissen aktiv demonstrieren oder nur beobachtet werden
- Beobachtetes hinterfragen! Geht es besser?
- Externer Beobachter hat bessere Chancen Ineffizienz zu erkennen

„Nobody can talk better about what they do and why they do it than they can while in the middle of doing it.“
— Beyer, Holtzblatt: Contextual Design:
Defining Customer-Centered Systems



Feldbeobachtung

- Requirements Engineer ist vor Ort bei den Fachexperten und beobachtet / dokumentiert ihre Arbeitsprozesse, Handgriffe, Arbeitsabläufe
- Video- oder Audio-Aufnahmen können unterstützen
- Gut bei schwer sprachlich zu vermittelnden Prozessen, die Essenz des Prozesses muss aber beobachtbar sein

Apprenticing

- Idee: „In die Lehre gehen“
- Requirements Engineer übernimmt Aufgaben des Stakeholder
- Requirements Engineer als Lehrling muss alles unklare sofort hinterfragen
- So lernt er alle Anforderungen aus erster Hand kennen und erfährt insbesondere Anforderungen, die für den Stakeholder selbstverständlich sind

Abschnitt 1.10.2.6

ERGÄNZENDE TECHNIKEN

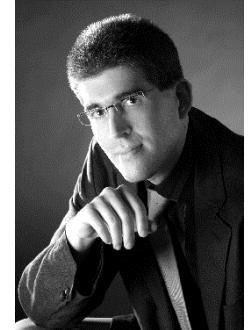
Unterstützende Techniken stammen aus der Moderation, Kommunikationspsychologie, Kognitionswissenschaft aber auch der Softwaretechnik

- **Personas:** Eine erfundene Persönlichkeit, die als Stellvertreter einer nicht verfügbaren Klasse von Stakeholdern dient
- **Workshops:** Viele Stakeholder beisammen, intensive Auseinandersetzung, oftmals für Anforderungsgewinnung der Benutzerschnittstelle. Besonders relevant: Business Use-Case Workshops

- **Persona** = **prototypischer Benutzer**. Modelliert deren Ziele, Verhaltensweisen und Eigenschaften, die im Hinblick auf das zu entwickelnde System relevant sind.
- Ziele
 - Identifikation typischer Benutzergruppen
 - Genaues Verständnis ihres Verhaltens und ihrer Eigenschaften, um so ihre spezifischen Bedürfnisse besser zu verstehen
 - Empathie des Entwicklungsteams für Persona als Prototyp der Benutzergruppe: „Wir bauen das System für Paul Professor und Stefan Student“
- Warum?
 - System, das allen Benutzergruppen gleichermaßen gerecht wird kaum möglich: z.B. Senior (85) vs. Teenager (16)
 - Benutzergruppen haben verschiedene Bedürfnisse: z.B. Anfänger vs. (ewiger) Fortgeschrittener

Personas – Beispiel

- **Name:** Paul Professor
- **Alter und biographische Angaben:** 43, ...
- **Ziele:** Stundenplanung für seine Fakultät, zufriedene Kollegen
- **Aufgaben:** Stundenplan in zwei Tagen fertigstellen
- **Kenntnisse:** Seit 8 Jahren Informatik-Professor, fortgeschrittene Kenntnisse in Excel, Programmiererfahrung in Scala, Java, Ruby,..., Experte für formale Programmanalyse, Erfahrung mit OO-Design und Softwarearchitektur
- **Einstellungen:** Beschäftigt sich gerne mit Software, mag IT-Gadgets, kocht gerne italienisch, betreibt Gesellschaftstanz, hört gerne Verdi-Opern
- **Erwartungen gegenüber dem System:** Paul möchte ohne viel Aufwand gute Ergebnisse, will aber auch Eingriffsmöglichkeiten, um die Stundenplanung zu tunen. Erhofft sich durch Software eine Entlastung



- **Ziele**
 - Was will die Persona (mithilfe des Systems) erreichen?
 - Welche Aufgaben muss die Persona erledigen?
 - Für was ist sie verantwortlich (Rolle, Verantwortungsbereich, Berufsbezeichnung, konkrete Tätigkeiten)?
- **Kenntnisse**

Ausbildung, Computerkenntnisse, Sprachkenntnisse, Wissen über verwandte Produkte und Altsysteme, Berufserfahrung, fachliches Wissen
- **Einstellungen**

Ängste, Sehnsüchte, Vorlieben, Abneigungen
- **Demografische Daten**

Geschlecht, Alter, ethnische Herkunft, ...
- **Erwartungen gegenüber dem System**

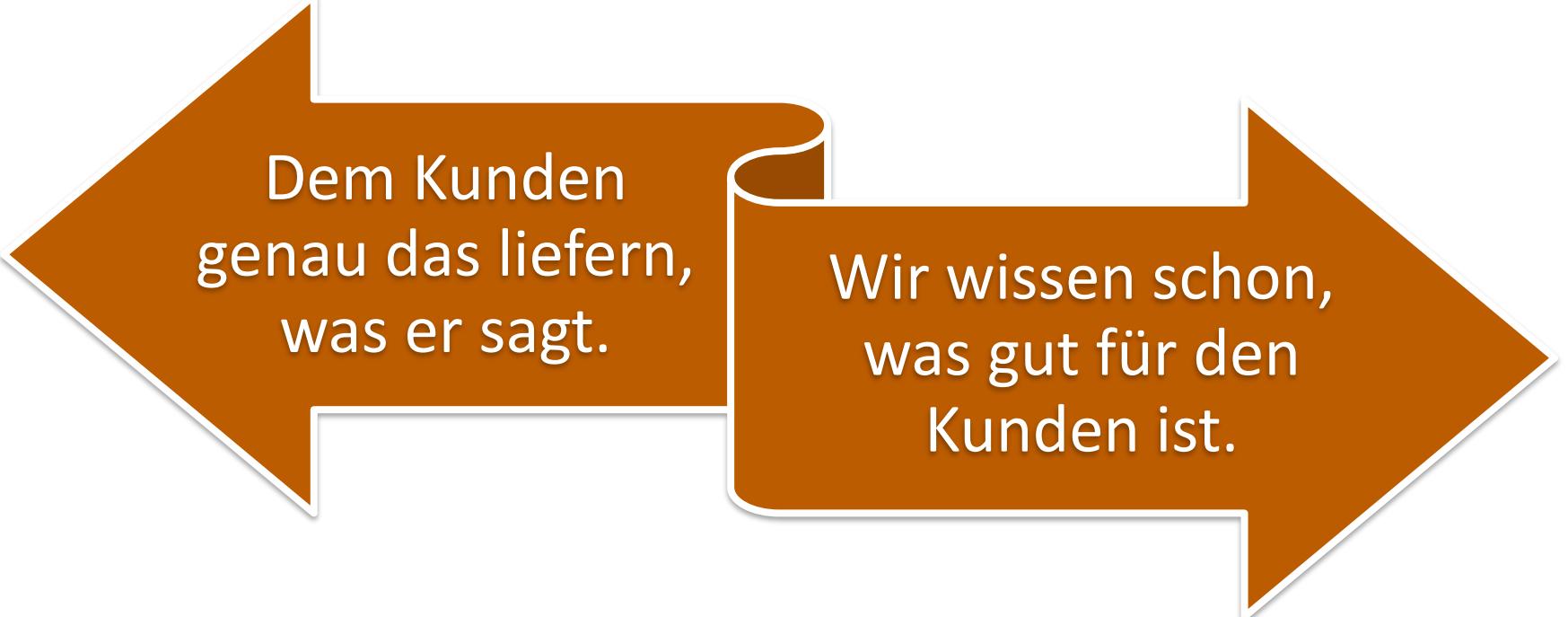
Erwartungen, Wünsche bezüglich des Systems, Einstellung zur Arbeit mit dem System

Abschnitt 1.10.3

INNOVATION ODER „MORE OF THE SAME“

Anforderungen und Innovation





Dem Kunden
genau das liefern,
was er sagt.

Wir wissen schon,
was gut für den
Kunden ist.

Beides falsch

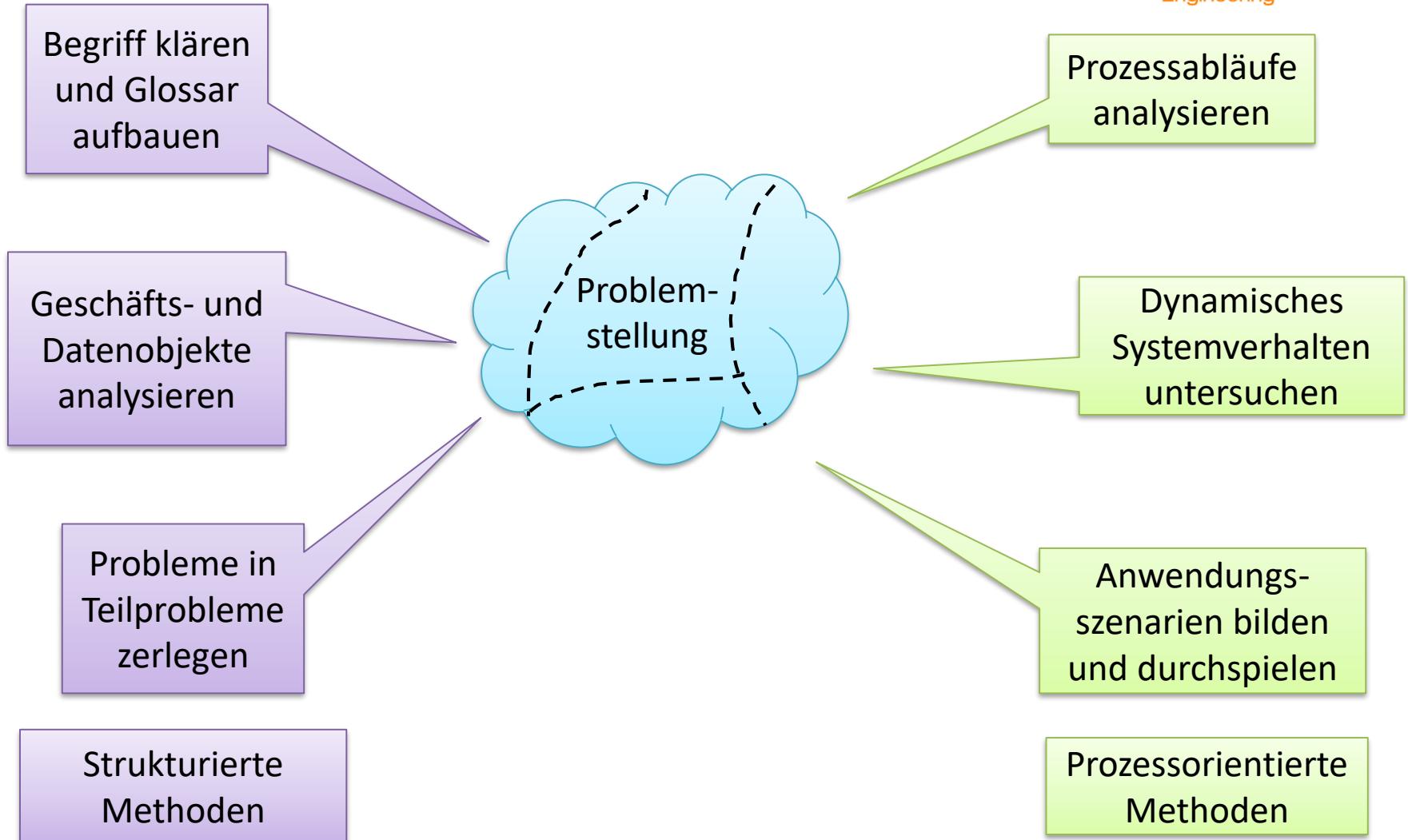
Wie entsteht Innovation?

- Den Beteiligten innovative Lösungen **vorschlagen**
- Kreativität der Beteiligten **anregen**
 - Zukunftsszenarien entwerfen und durchspielen
 - Alle Beschränkungen fallen lassen
 - Metaphern suchen und explorieren
- Anforderungen lassen sich nicht einfach „erheben“

Abschnitt 1.10.4

VORGEHEN UND REGELN DER ANFORDERUNGSGEWINNUNG

Methodisches Vorgehen



- Aussagen aus der Anforderungsermittlung enthalten oft eine Lösungsidee des Stakeholders, nicht sein zugrundeliegendes Problem.
- Der Analyst muss daraus die Essenz des Problems ableiten, um
 - das echte Problem lösen zu können,
 - zu vermeiden, alte Technologieentscheidung zu übernehmen.
- Bei der Suche nach der Essenz der geschäftlichen Aufgabe die aktuelle und die zukünftige Technologie ausblenden.

Problem vs. Lösung: Beispiele

Das Produkt muss läuten und eine blinkende Nachricht auf dem Bildschirm anzeigen, sobald eine Wetterstation keine Messdaten übermittelt.

Das Produkt muss eine Warnung ausgeben, sobald eine Wetterstation keine Messdaten übermittelt.

Der Reisende muss sein Ziel durch Berührung auf einer Karte auf dem Bildschirm auswählen.

Das Produkt muss die Eingabe des Reiseziels ermöglichen.

- Erwartungs- und Begriffs-diskrepanzen bei den Beteiligten
- Stakeholder wissen nicht, was sie wollen
- Stakeholder wissen was sie wollen, können ihre Vorstellungen aber nicht verbalisieren
- Beteiligte verfolgen verdeckte Ziele, die sie absichtlich nicht offenlegen
- Beteiligte sind auf bestimmte Lösung fixiert

Requirements Engineering ist deshalb immer auch

- Aufgabenklärung
- Risikoanalyse
- Konflikterkennung
- Konfliktauflösung
- Konsensbildung
- Anregen der Kreativität der Beteiligten
- Motivation

- Methodische Hilfsmittel für die Analyse von Rohdaten aus der Anforderungsgewinnung und dem Kickoff
- Drei bekanntere Verfahren :
 - Szenarienanalyse
 - Objekt/Nominalphrasenanalyse
 - Ereignis-Reaktions-Analyse

Grundidee: Analyse logischer Interaktionssequenzen zwischen dem Systemkontext und dem Aufgabenbereich

Geeignet für:

- Bildung und Analyse von Anwendungsszenarien
- Analyse des dynamischen Systemverhaltens
- Sekundär auch: Analyse von Prozessabläufen

Liefert:

- Szenarien und Anwendungsfälle
- Prozessablaufmodelle

Grundidee: Analyse der **grammatikalischen Struktur** gegebener Texte

Voraussetzung: **Text vorhanden** (schriftlich oder mündlich)

Geeignet für:

- Analyse von Geschäfts- und Datenobjekten
- Aufbau von Glossaren
- Erkennen von Vorgängen

Liefert:

- Glossar
- Kandidaten für Funktionalität

Problem: Liefert große Menge schwach strukturierte Kandidaten für relevante Begriffe

Durchführung

- Text grammatikalisch analysieren
 - Grammatisches *Subjekt*, grammatisches *Objekt* → Kandidaten für *Objekte*, *Klassen*, *Attribute* oder *Wertebereiche*
 - *Verben* beschreiben Zusammenhänge oder Handlungen:
 - Zusammenhänge → *Beziehungen*, *Attribute*
 - Handlungen → *Funktionalität*, *Verhalten*
 - *Adjektive* präzisieren Aussagen oder schränken sie ein

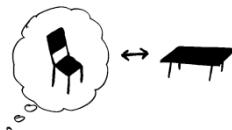
Durchführung II

- Fragmente **klassifizieren, ordnen, vervollständigen**
 - **Synonyme** identifizieren
 - Konkrete Objekte und Attributwerte **abstrahieren** zu Klassen und Attributen
 - Neu gewonnen Information mit bereits vorhandenen Modellteilen **verknüpfen**
- Problem der Abgrenzung von Klassen/Objekten gegen Attribute/Werte:
 - Objekt muss Identität haben
 - Attributwerte sind Daten ohne eigene Identität

Abschnitt 1.11

ANFORDERUNGSDOKUMENTATION

Qualitätskriterien für Anforderungen



Adäquat: Beschreiben, was der Kunde will/braucht



Vollständig: Beschreiben alles, was der Kunde will/braucht



Widerspruchsfrei: Inkonsistente Anforderungen können nicht umgesetzt werden

$$P(m) \Leftrightarrow m \geq 2 \wedge \forall m_0. (m_0 | m \Rightarrow m_0 = 1 \vee m_0 = m)$$

Verständlich: Anforderungen müssen für alle Beteiligten (auch Kunden) verständlich beschrieben sein



Eindeutig: Vermeidet Fehlinterpretationen



Prüfbar: Sonst kann nicht festgestellt werden, ob das System den Anforderungen genügt



Risikogerecht: Je geringer das akzeptierte Risiko, desto umfangreicher die Spezifikation

Abschnitt 1.11.1

DAS ANFORDERUNGSDOKUMENT

Aufbau einer Anforderungsspezifikation

- Keine definitiven Vorgaben aber viele Vorschläge
- Teilweise unternehmensinterne Standards
- IEEE Standard 830-1998 enthält Gliederung, die speziell für die Dokumentation von Softwareanforderungen entwickelt wurde
- Anderer Gliederungsvorschlag ist das Volere Requirements Specification Template

Institute of Electric and Electronic Engineers: IEEE Recommended Practice for Software Requirements Specifications (IEEE Std. 830-1998)

Warum Standardgliederungen?

- Anforderungsdokumente enthalten eine Vielzahl von Informationen, die **für den Leser gut strukturiert** sein müssen
- Standardgliederungen bieten **vordefinierte Kategorien**, in die Informationen eingesortiert werden können
- Standardgliederungen geben die Grobstruktur vor und erläutern, welche Inhalte in den einzelnen Kapiteln stehen sollen

Warum Standardgliederungen?

- Die Festlegung einer standardisierten Gliederung bietet eine Vielzahl von Vorteilen:
 - Leichtere Einarbeitung neuer Mitarbeiter
 - Schnelleres Erfassen ausgewählter Inhalte
 - Ermöglichen selektives Lesen/Überprüfen von Anforderungsdokumenten
 - Ermöglichen automatische Überprüfung gewisser Qualitätsanforderungen, z.B. Vollständigkeit
 - Verbessern die Wiederverwendung von Anforderungsdokumenten

Volere Requirements Specification Template

Projekttreiber

1. Zweck des Projekts
2. Auftraggeber, Kunde, andere Stakeholder
3. Benutzer des Produkts

Randbedingungen des Projekts

4. Einschränkungen
5. Namenskonventionen und Definitionen
6. Relevante Fakten und Annahmen

Funktionale Anforderungen

7. Aufgabenbereich (Scope of the work)
8. Systemgrenzen (Scope of the product)
9. Funktionale und Daten-Anforderungen

Nichtfunktionale Anforderungen

10. Look-and-feel-Anforderungen
 11. Usability-Anforderungen
 12. Leistungsanforderungen
 13. Operationale und Umfeld-Anforderungen
-
14. Wartungs- und Unterstützungsanforderungen
 15. Sicherheitsanforderungen
 16. Kulturelle und politische Anforderungen
 17. Rechtliche Anforderungen
- ## Projektspekte
18. Offene Punkte
 19. Standardlösungen
 20. Neue Probleme
 21. Aufgaben
 22. Migrationstätigkeiten
 23. Risiken
 24. Kosten
 25. Benutzerdokumentation
 26. Zurückgestellte Anforderungen
 27. Lösungsideen

Volere Requirements Resources: www.volere.co.uk

S. Robertson, J. Robertson: Mastering the Requirements Process, Addison-Wesley, 2006

Volere Requirements Specification Template

- **Zweck des Projekts:** Warum wird das Projekt durchgeführt? Welches Problem gibt es im Geschäft des Auftraggebers und wie soll das Produkt bei der Lösung helfen?
- **Auftraggeber, Kunde, andere Stakeholder:** Wer hat ein Interesse an dem Produkt? Welche Bedürfnisse haben die Stakeholder in Bezug auf das Produkt?
- **Benutzer des Produkts:** Wer interagiert später direkt mit dem Produkt? Welche Eigenschaften und Kompetenzen haben diese Leute?

- **Einschränkungen:** Einschränkungen sind global, sie betreffen das Produkt als Ganzes. Das Produkt muss so konstruiert werden, dass es den Einschränkungen genügt. Einschränkungen sind damit auch Requirements, allerdings von außen vorgegeben.
- **Namenskonventionen und Definitionen:** Das spezifische Vokabular des Projekts. Begriff der Problemdomäne, Akronyme und Data Dictionary
- **Relevante Fakten und Annahmen:** Relevante Fakten sind externe Faktoren mit Auswirkungen auf das Projekt, die aber in keinen anderen Abschnitt passen. Die Annahmen sind solche, die das Projektteam macht. Sie dienen als Warnung an das Management über Faktoren, die den Projekterfolg beeinflussen können.

- **Aufgabenbereich:** Wie ist der Aufgabenbereich abgegrenzt? Mit welchen Geschäftsbereich befassen wir uns, wie passt der Aufgabenbereich in die Umgebung?
- **Systemgrenzen:** Was ist der Umfang des Produkts?
Oftmals ein Kontextmodell auf Systemebene oder ein Use-Case-Diagramm plus Liste der Product-Use-Cases
- **Funktionale und Daten-Anforderungen:** Die atomaren Anforderungen, die beschreiben, was das Produkt tun soll
- **Nicht-funktionale Anforderungen (10-17):** Siehe Abschnitt atomare Anforderungen

- **Offene Punkte:** Gibt es Probleme, die bisher ungelöst sind?
- **Standardlösungen:** Gibt es verfügbare fertige Lösungen? Welche Eigenschaften haben sie, inwiefern erfüllen sie die Anforderungen?
- **Neue Probleme:** Änderungen an der bestehenden Ordnung der Dinge können negative Effekte haben. Welche Probleme werden durch die Installation des neuen Produkts geschaffen?
- **Aufgaben:** Welche Schritte sind nötig, um das Produkt auszuliefern?

- **Migrationstätigkeiten:** Welche Tätigkeiten sind erforderlich, um bei der Installation die Einsatzbereitschaft des Produkts herzustellen? Sind z.B. Datenbanken zu migrieren?
- **Risiken:** Alles, was den Projekterfolg gefährdet.
- **Kosten:** Was kostet es, das Produkt herzustellen?
- **Benutzerdokumentation:** Was muss an Unterlagen, Handbüchern, Training, ... für die Benutzer erstellt bzw. vorbereitet werden?

- **Zurückgestellte Anforderungen:** Anforderungen, die nicht in die aktuell zu erstellende Produktversion integriert werden können.
- **Lösungsideen:** Lösungsideen oder Vorschläge, die im RE-Prozess auftauchen oder vorgeschlagen werden, werden in diesem Bereich gesammelt.

Die **Referenzstruktur** für eine **Software Requirements Specification** nach IEEE830-1998 untergliedert das Anforderungsdokument in mehrere Teile, wobei der Hauptteil aus drei Kapiteln besteht.

Table of Contents	3. Specific Requirements
1. Introduction	1. Functional Requirements
1. Purpose	2. External Interface
2. Scope	Requirements
3. Definitions	3. Performance Requirements
4. References	4. Design Constraints
5. Overview	5. Attributes
2. General Description	6. Other Requirements
1. Product Perspective	Appendixes
2. Product Functions	Index
3. User Characteristics	
4. General Constraints	
5. Assumptions and	
Dependencies	

Abschnitt 1.11.2

SZENARIEN UND ANWENDUNGSFÄLLE (USE-CASES)

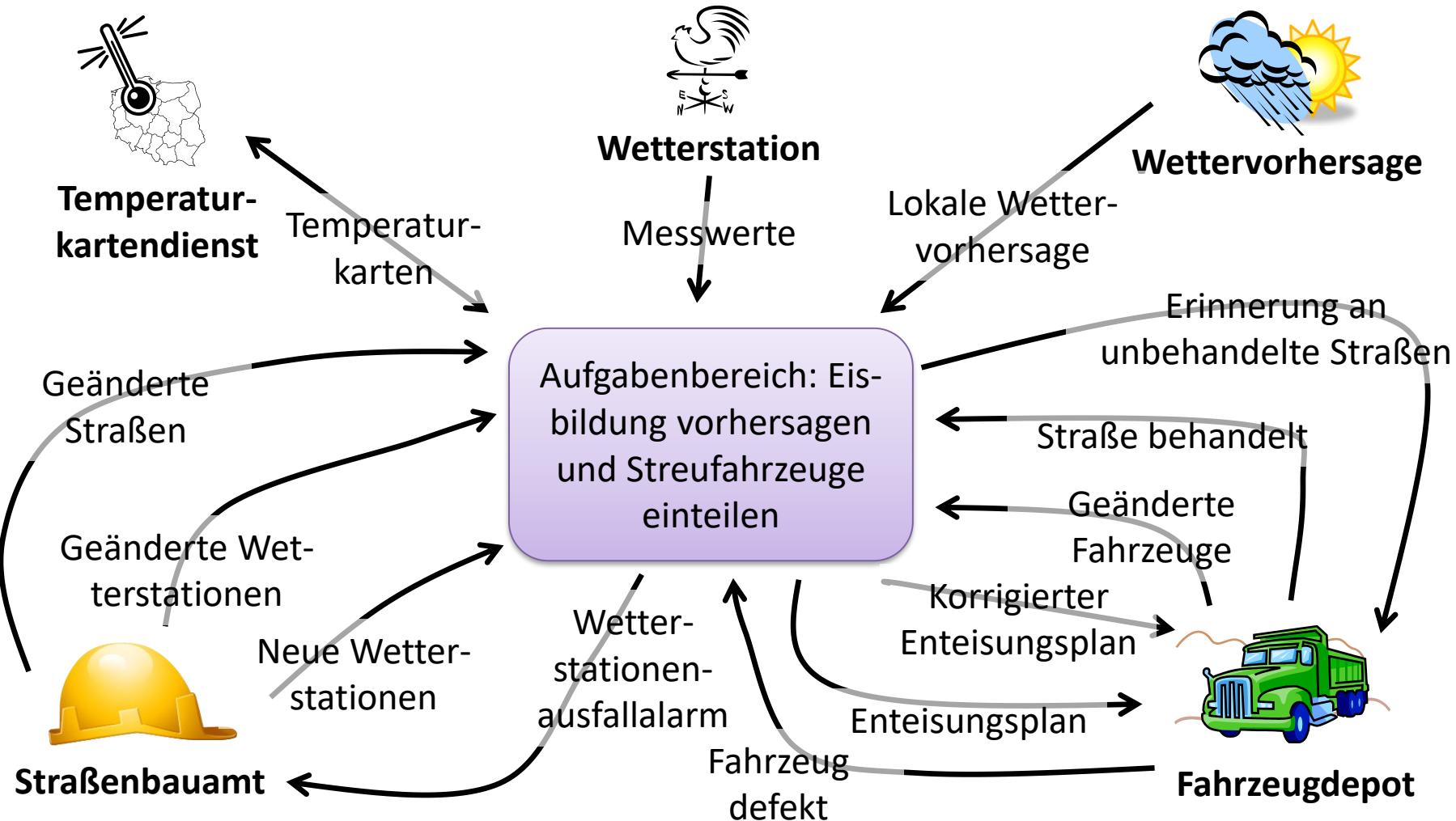
- Ein Ergebnis des Kickoff ist der Umfang des Aufgabenbereichs. Dieser definiert einen Geschäftsbereich, von dem ein Teil durch das Produkt automatisiert werden soll.
- Den gesamten Aufgabenbereich als ganzes zu untersuchen ist üblicherweise nicht möglich.
- Eine Heuristik zur Zerteilung des Aufgabenbereichs in einzeln analysierbare Teile sind **Use-Cases**.
- Der zu untersuchende **Aufgabenbereich ist nicht nur das neue System!** Stattdessen umfasst er alles, was definiert, wie das Geschäft funktioniert, inklusive Menschen, Computern, Maschinen, Telefonen, Fotokopierern, usw.

Für eine Partitionierung suchen wir Teile des Aufgabenbereichs mit folgenden Eigenschaften:

- „Natürliche“ Partitionierung des Aufgabenbereichs. Alle **Teile leisten** einen offensichtlichen und sinnvollen **Beitrag**
- **Minimale Verbindungen** zu anderen Teilen des Aufgabenbereichs
- **Klar umrissener Umfang**
- Haben **Grenzen**, die erkannt und definiert werden können
- Lassen sich in der **Sprache der Fachdomäne benennen**
- Die existierenden solchen Teile können leicht bestimmt werden
- Haben einen oder zwei Stakeholder, die Fachleute für diesen Teil des Aufgabenbereichs sind

Use-Cases finden: Business Events

Zurück zum Kontextdiagramm



Grundidee: Analysieren der auf den Aufgabenbereich einwirkenden **Anstöße** aus dem Kontext und der daraufhin erwarteten **Reaktionen** des Aufgabenbereichs

Geeignet für:

- Analyse das **dynamischen Verhaltens**
- Sekundär auch: Analyse von **Prozessabläufen**

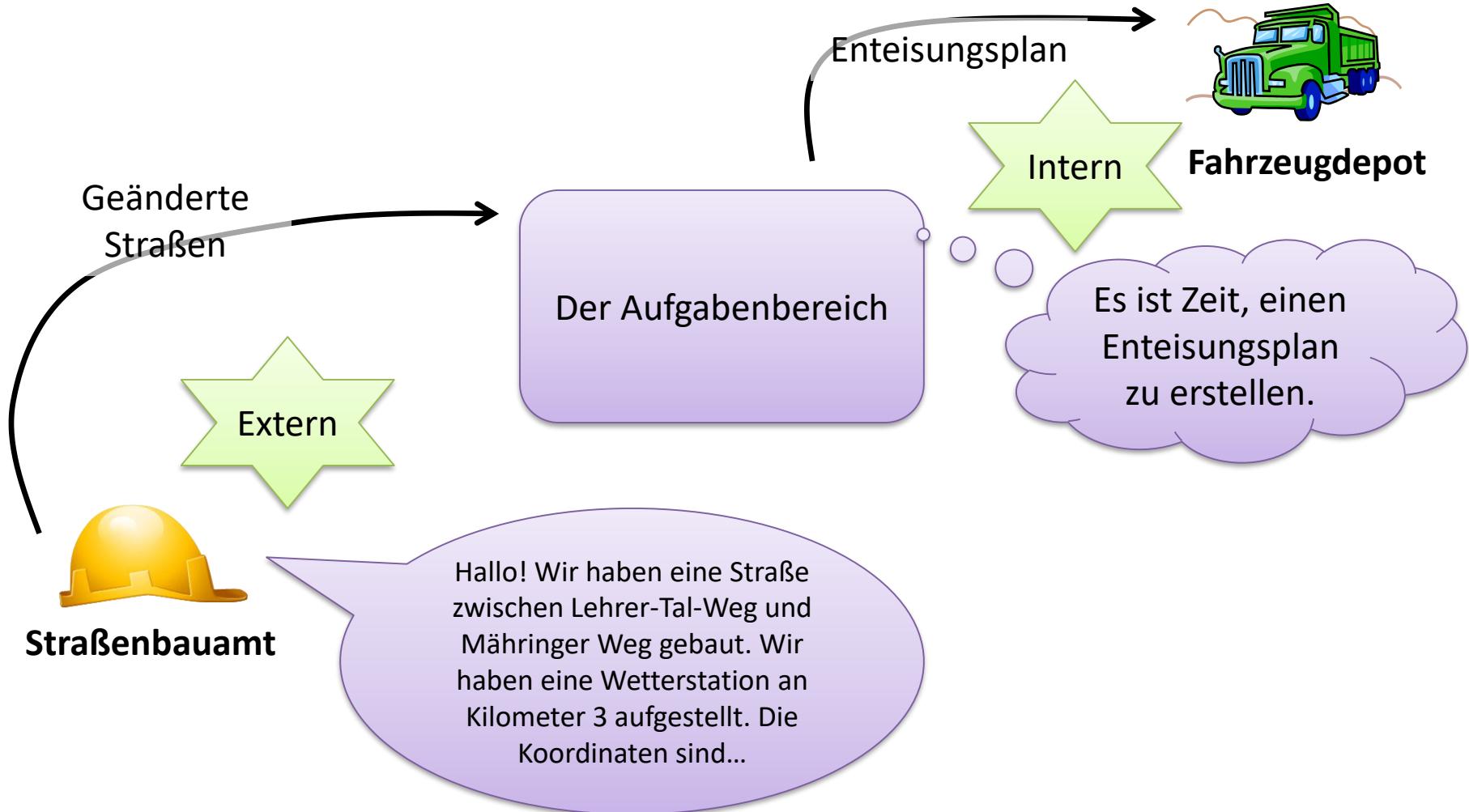
Liefert:

- Szenarien und Anwendungsfälle
- Zustandsbasierte Verhaltensmodelle
- Prozessablaufmodelle

Durchführung

- Alle Ereignisse, die eine Reaktion des Aufgabenbereichs erfordern, auflisten
- Für jedes Ereignis die erforderlichen Reaktionen bestimmen
- Die Reaktion auf ein Business Event heißt ein Business Use-Case

Arten von Business Events



Warum Business Events?

- Eine objektive Partitionierung des Aufgabenbereichs und das Verständnis des Aufgabenbereichs selbst sind entscheidend dafür, die echten Anforderungen zu finden und dies schnell.
- Eine Aufteilung anhand (externer) Stimuli liefert eine solche intersubjektive Zerteilung. Insbesondere ist sie an der Sichtweise des Kunden orientiert.
- Business Events zeigen, welche Dinge zusammengehören und liefern so Partitionen mit minimalen Schnittstellen untereinander.

Warum Business Events?

Man untersucht Business Events und Business Use-Cases, um den Aufgabenbereich als Ganzes zu verstehen, anstatt nur den Teil zu analysieren, der von einem realen oder geplanten System erledigt wird.

Ein Versicherungsmitarbeiter erhält eine Forderung von einem Versicherten und gibt die Forderung in das System ein.

Diese Sichtweise führt zur Analyse der Arbeit des Mitarbeiters und wie man die Daten des Versicherungsfalls eingibt. Das ist aber nicht der ganze Use-Case, sondern nur eine Implementierung. Der **Unfall** ist der eigentliche Auslöser.

Noch ein Beispiel

Ein Anruf geht beim Help-Desk ein. Der Help-Desk-Mitarbeiter beginnt den Vorgang damit, den Anrufer nach seinem Problem zu fragen und den Vorgang speichern.

- Der Use-Case ist dann das Speichern, der Akteur der Help-Desk-Mitarbeiter
- Ist das tatsächlich der Beginn des Geschäftsvorfalls?

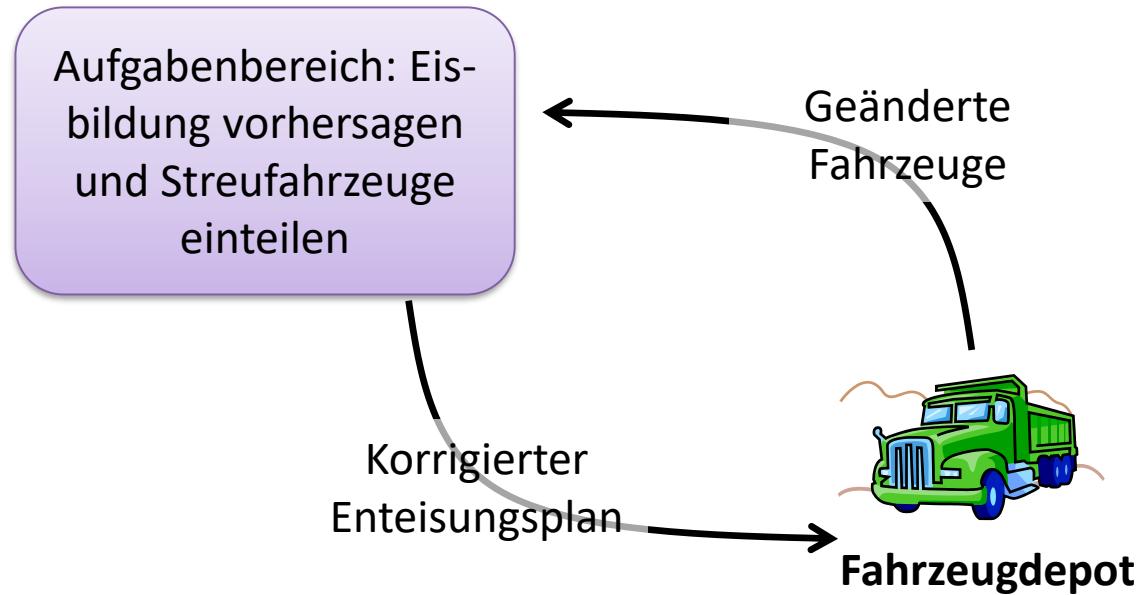
Noch ein Beispiel

Ein Anruf geht beim Help-Desk ein. Der Help-Desk-Mitarbeiter beginnt den Vorgang damit, den Anrufer nach seinem Problem zu fragen und den Vorgang speichern.

- Der Use-Case ist dann das Speichern, der Akteur der Help-Desk-Mitarbeiter
- Ist das tatsächlich der Beginn des Geschäftsvorfalls?
 - Der Beginn könnte der Anruf sein
 - oder die Fehlfunktion des Geräts beim Anrufer

- Business Events sind Dinge die passieren und in unserem Aufgabenbereich eine Reaktion auslösen
- Sie können extern sein oder innerhalb des Umfangs des Aufgabenbereichs
- In beiden Fällen findet eine Kommunikation mit einem angrenzenden System statt
- Diese Informationsflüsse finden sich im Kontextdiagramm

Beispiel: Ein Event, zwei Informationsflüsse



Abschnitt 1.11.2.1

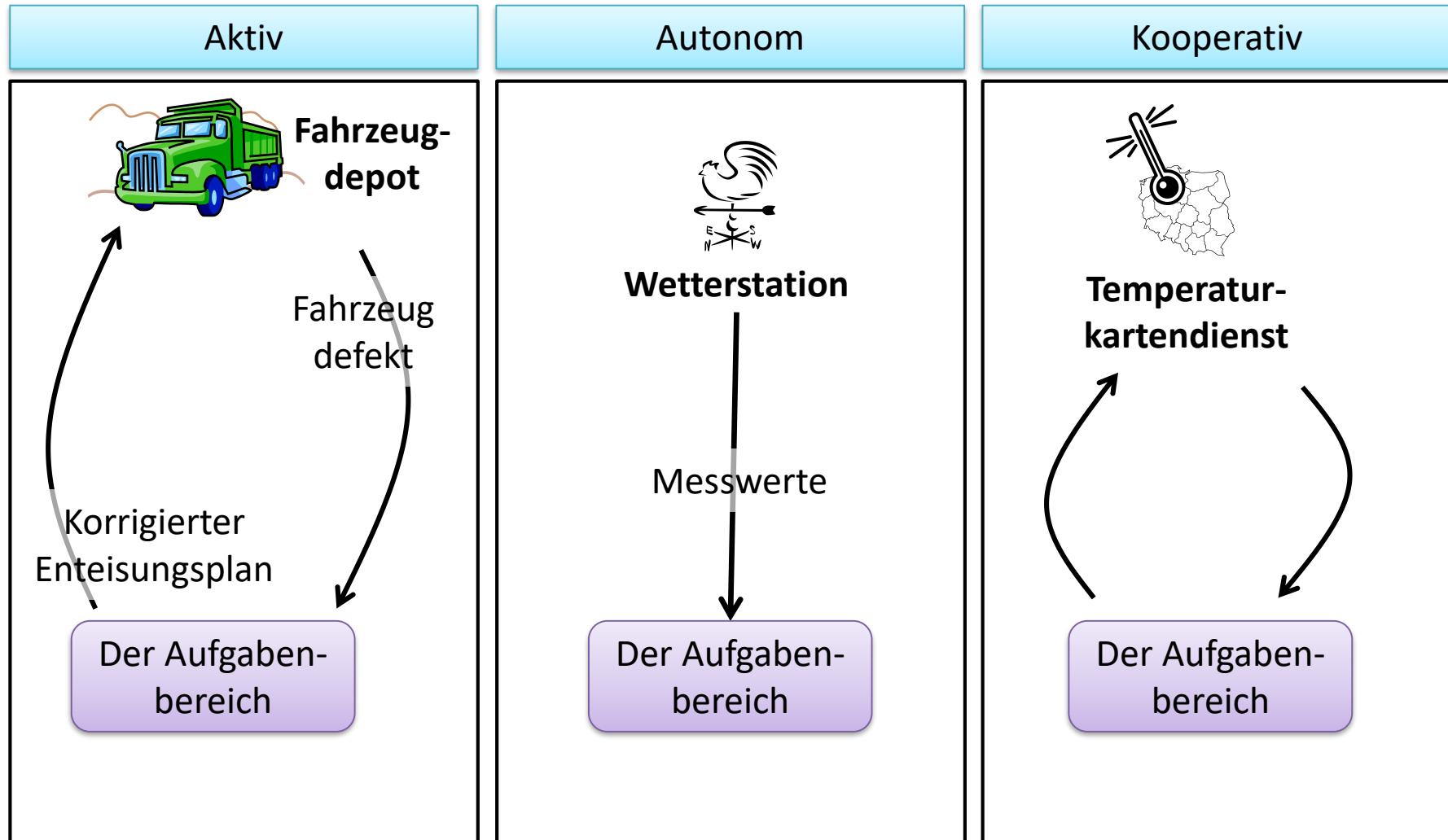
BUSINESS USE-CASES

- Ein **Business Use-Case** ist die Reaktion des Aufgaben-bereichs auf ein Business Event
- Ein Business Use-Case ist ein zusammenhängendes Stück Funktionalität und besteht aus:
 - Einer Menge von identifizierbaren Prozessen
 - Verarbeiteten und gespeicherten Daten
 - Produzierten Ausgaben
 - Versandten Nachrichten
- Einzelne Business Use-Cases können gut unabhängig untersucht werden
- Business Use-Cases werden so erfasst und beschrieben, dass sie mit jeder technischen Lösung sinnvoll sind (auch ohne Systemunterstützung!)

Angrenzende Systeme untersuchen

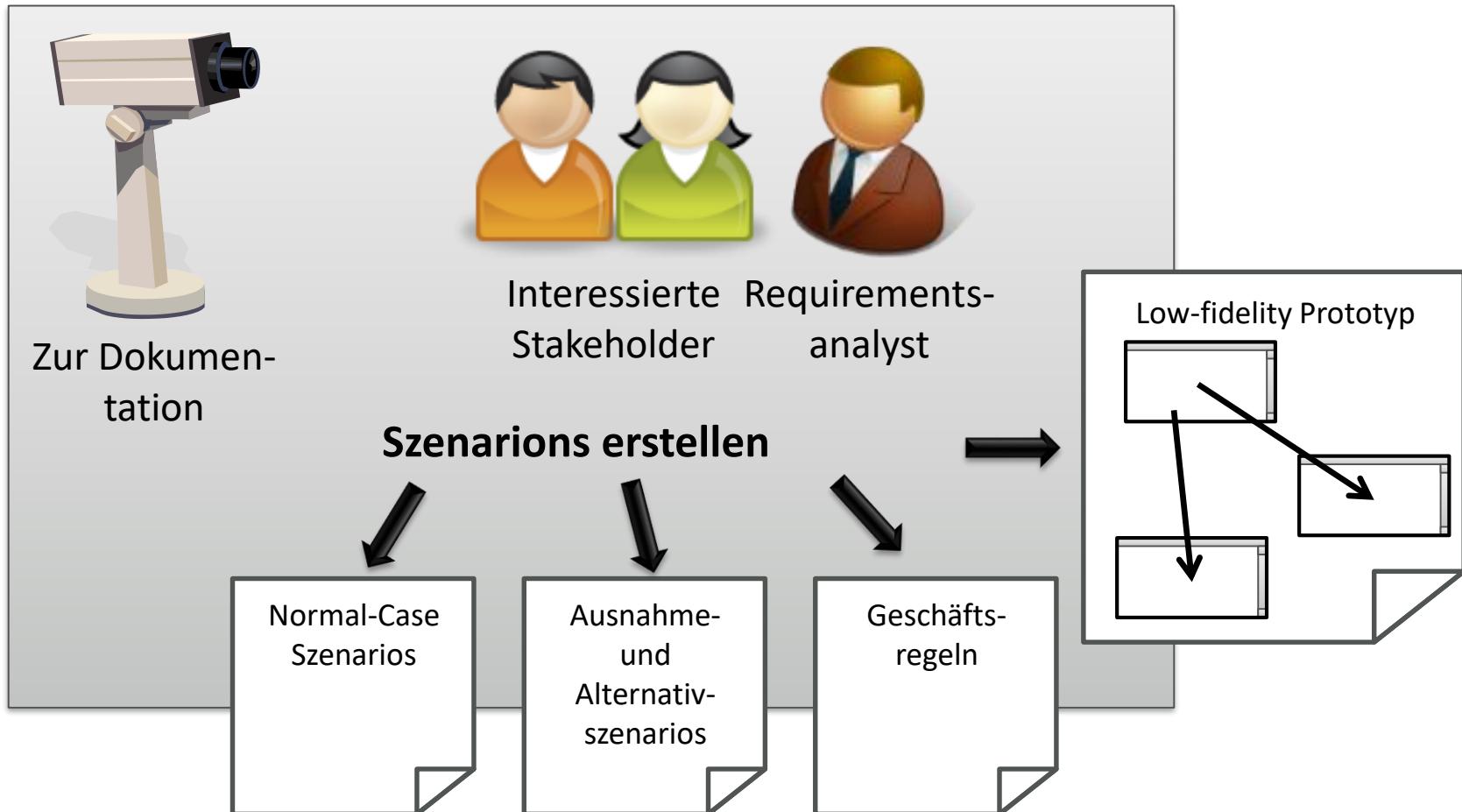
- Die angrenzenden Systeme sind Quellen für Daten oder Empfänger von Ergebnissen unserer Aufgabe.
- Dies kann eine Organisation, ein Individuum, ein Softwaresystem, jedes andere Stück Technik, usw. sein
- Zu klären ist, wie ein angrenzendes System in den von ihm ausgelösten Use-Case involviert werden kann:
 - Welche (technischen) Fähigkeiten hat es
 - Welches Ergebnis strebt es an
 - Was ist aus der Sicht unserer Arbeit das angestrebte Ergebnis

Varianten angrenzender Systeme



- Ziel ist die Formulierung von Szenarien für die Business Use-Cases
- Typisch ist eine kleine Zahl von Stakeholder, um auf deren Use-Cases zu fokussieren
- Es werden Business Use-Case Szenarien formuliert und deren Adäquatheit von den Stakeholder bestätigt
- Sowohl Normal-Case-Szenarien als auch Ausnahme- und Alternativszenarien

Business Use-Case Workshops

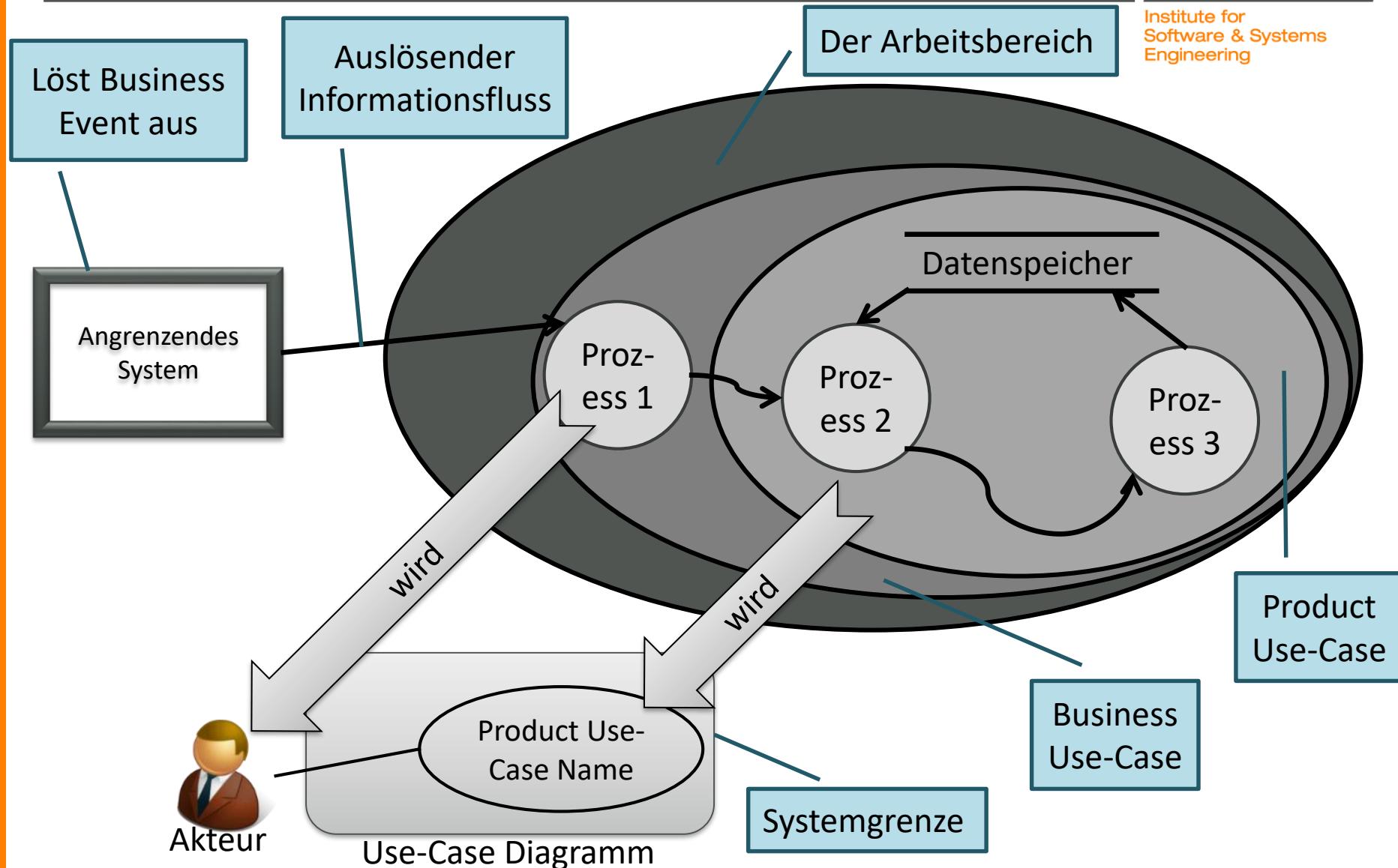


Abschnitt 1.11.2.2

PRODUCT USE-CASES

- Business Use-Case ist die Reaktion unseres Aufgabenbereichs auf ein Business Event
- Untersuchung der Business Use-Cases zielt auf ein möglichst breites Verständnis der zu erledigenden Aufgabe
- Mit diesem Verständnis und im Austausch mit dem Kunden bestimmt man nun den Anteil des Use-Cases, der Teil des Produkts wird. Kann das angrenzende System anders als bisher in die Arbeit einbezogen werden? Welchen Beitrag soll es leisten?
- Der Teil der Reaktion, den das neue System leistet, bildet einen Product Use-Case

Business und Product Use-Case



Abschnitt 1.11.2.3

USE-CASE GRUNDLAGEN

Grundidee: Die **Interaktion** zwischen **systemexternen Akteuren** und dem **System** ins Zentrum der Betrachtungen stellen

Jede Interaktionssequenz wird durch ein **Szenario** beschrieben und beschreibt die Geschehnisse beim Durchlaufen eines Business Use-Case

Liefert **benutzerorientierte Beschreibung** der **Funktionalität**

Szenario

Eine geordnete Menge von Interaktionen zwischen Partnern, in der Regel zwischen einem Aufgabenbereich und einer Menge angrenzender Akteure.

Kann eine konkrete Interaktionsfolge (Beispielszenario) oder eine Menge möglicher Interaktionen (abstraktes Szenario oder Typszenario) sein.

Anwendungsfall

Ein Anwendungsfall (Use-Case) bündelt alle möglichen Szenarien, die eintreten können, wenn ein Akteur versucht, mit Hilfe des betrachteten Systems ein bestimmtes fachliches Ziel zu erreichen.

Akteur

1. Ein Mensch, der Ziele hat und diese verfolgt oder ein Element des Anwendungsbereichs, das zur Erreichung eines Ziels dient und hierzu handelt oder Informationen verarbeitet. 2. Eine Rolle, welche ein externes System oder ein Mensch gegenüber dem System einnehmen kann.

Wikipedia: <https://de.wikipedia.org/wiki/Anwendungsfall>

I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard: Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley, 1992

Abschnitt 1.11.2.4

SZENARIOBESCHREIBUNG

- Freier Text
- Strukturierter Text: In dieser Darstellungsform der wesentliche Bestandteile jedes (ausführlichen) Use-Cases
- UML-Aktivitätsdiagramme
- Interaktionsdiagramme

Szenariobeschreibung mit freiem Text

- Häufig zur Beschreibung konkreter Beispieldaten verwendet.
- Das Ergebnis von Gewinnungstechniken (Interview, Beobachtung) hat oft diese Form.

- Flexibel und ausdrucksmächtig
- Von Anwendungsexperten les- und schreibbar

- Unpräzise, Missverständnisse leicht möglich
- Fehler werden leicht übersehen

Beispiel

„Ich rufe den nächsten wartenden Kunden auf. Wenn er an meinem Schalter angekommen ist, frage ich nach dem Ticket. Wenn der Passagier e-ticket benutzt, benötige ich seine Buchungsnummer. Die meisten Passagiere sind nicht vorbereitet genug, um diese dabei zu haben, deshalb frage ich dann nach ihrem Namen und dem gebuchten Flug. Die meisten kennen die Flugnummer nicht, daher frage ich üblicherweise nach dem Zielort. Den sollten sie kennen!

Ich stelle dann sicher, dass ich den richtigen Passagier und den richtigen Flug habe. Es wäre recht peinlich, den Platz von jemand anders zu vergeben oder einen Passagier zum falschen Ort zu schicken. Wie auch immer, irgendwie finde ich den Flugdatensatz des Passagiers im Computer. Ich prüfe, ob das Passbild wie der Passagier aussieht und ob der Pass noch gültig ist.

Wenn für die Buchung keine Vielfliegernummer hinterlegt ist, frage ich den Passagier, ob er Meilen sammelt. Entweder gibt er mir seine Karte mit der Vielfliegernummer oder ich frage ihn, ob der in Zukunft sammeln möchte. Wenn ja, gebe ich ihm den Mitgliedsantrag. Wir können eine vorläufige Vielfliegernummer mit der Buchung verknüpfen, so dass der Passagier für diese Reise schon Meilen erhält.

Szenariobeschreibung mit freiem Text

Beispiel

Wenn der Computer nicht schon einen Sitz zugewiesen hat, lege ich einen fest. Das heißt üblicherweise, dass ich den Passagier frage, ob er Gang oder Fenster bevorzugt, oder, falls das Flugzeug schon fast voll ist, sage ich ihm, was noch frei ist. Falls der Computer schon einen Sitz vergeben hatte, frage ich natürlich, ob dieser OK ist. Irgendwie einigen wir uns auf einen Sitzplatz und ich bestätige das im Computer. Jetzt kann ich die Bordkarte drucken, aber normalerweise mache ich zuvor das Gepäck.

Ich frage, wie viele Gepäckstücke der Passagier einchecken will und, zur gleichen Zeit, prüfe ich, dass er nicht die Mitnahmegerünen überschreitet. Es ist unglaublich, was manche Leute meinen, in ein doch sehr platzbeschränktes Flugzeug mitnehmen zu können. Ich stelle die Sicherheitsfragen über die Gepäckstücke und bekomme vom Passagier Antwort. Ich drucke die Gepäckstreifen und befestige sie sicher am Gepäck und dann schicke ich das Gepäck auf den Weg.

Als nächstes drucke ich die Bordkarte. Das heißt, dass, soweit es den Computer betrifft, alles erledigt ist. Aber eine Sache gibt es noch: Ich muss sicherstellen, dass der Passagier alles verstanden hat. Ich lese von der Bordkarte vor, wohin er gehen muss, wann der Flug und wann das Boarding ist. Außerdem lese ich vor, wie viele Gepäckstücke eingecheckt wurden und prüfe ob das Ziel auch das Flugziel des Passagiers ist. Ich gebe ihm die Dokumente und wünsche ihm eine gute Reise.“

Szenariobeschreibung mit strukturiertem Text

Am häufigsten als Teil der Beschreibung von Anwendungsfällen

- Der informelle Text aus der Requirementsgewinnung wird auf eine Sequenz von Schritten reduziert
- In der Regel drei bis zehn Schritte
- Man beginnt mit dem normalen Verlauf
- Weitere Angaben im Use-Case, zusätzlich zum Szenario:
 - Auslösendes Business Event, Business Use-Case Name und Nummer
 - Akteure und Auslöser für die Ausführung des Szenarios
 - Vorbedingung
 - Interessierte Stakeholder
 - Ergebnis
 - Angabe möglicher Ausnahmefälle und alternativer Abläufe

Schritt 1: Szenario formulieren

- Business Use-Case auf Sequenz von ca. drei bis zehn Schritten reduzieren
- Mit Stakeholdern klären, ob das kompakte Szenario adäquat ist

1. Ticket oder Buchungsnummer oder Name und Flugnummer des Passagier bestimmen.
2. Stimmen Passagier, Flug und Zielort?
3. Ist der Pass gültig und gehört er zum Passagier?
4. Vielfliegernummer speichern.
5. Einen Sitzplatz bestimmen.
6. Die Sicherheitsfragen stellen.
7. Gepäck für den Flug einchecken.
8. Bordkarte und Gepäckabholzettel ausdrucken und übergeben.
9. „Ich wünschen einen angenehmen Flug.“

Schritt 2: Weitere Angaben zum Use-Case I

- **Business Event-Name:** Auf welches Business Event antwortet dieser Use-Case.

Business Event-Name: Passagier will einchecken.

- **Business Use-Case-Name:** Jeder Use-Case bekommt einen aussagekräftigen Namen

Business Use-Case-Name: Passagier auf Flug einchecken

- **Trigger/Auslöser:** Was löst die Abarbeitung des Use-Case aus? Typischerweise Daten oder eine Anfrage von außerhalb.

Trigger: Check-In-Anfrage mit Ticket, Buchungsnummer oder Identität und Flugnummer des Passagiers

Schritt 2: Weitere Angaben zum Use-Case II

- **Vorbedingungen:** Was muss zu Beginn des Use-Case gelten?

Vorbedingungen: Passagier muss eine Reservierung haben.

- **Interessierte Stakeholder:** Wer hat ein Interesse am Ergebnis des Use-Case?

Interessierte Stakeholder: Passagier, Check-In Agent, Marketing, Gepäcktransport, Reservierungen, Flugbeladungssystem, Security, Immigrationsbehörde im Zielland, Betriebsablauf

Schritt 2: Weitere Angaben zum Use-Case III

- **Aktive Stakeholder:** Wer handelt in diesem Use-Case?

Aktive Stakeholder: Passagier (Auslöser), Check-In Agent

- **Ergebnis:** Der gewünschte Zustand nach dem Use-Case

Ergebnis: Der Passagier ist als für den Flug eingemecheckt registriert, die Gepäckstücke sind dem Flug zugewiesen, ein Sitzplatz ist zugeteilt und der Passagier ist im Besitz der Bordkarte und der Gepäckabholzettel.

Schritt 3: Normal-Case-Szenario finalisieren I

- Die Skizze des Hauptszenarios wird nochmals auf offene Fragen, nicht technologieneutrale Formulierungen, usw. untersucht und ggf. verbessert
- Nicht alles lässt sich vermeiden, manche Vorgaben sind fix
 1. Ticket oder Buchungsnummer oder Name und Flugnummer des Passagier bestimmen.

wird zu

1. Die Reservierung des Passagiers finden.

Schritt 3: Normal-Case-Szenario finalisieren II



Institute for
Software & Systems
Engineering

2. Stimmen Passagier, Flug und Zielort?

wird zu

2. Sicherstellen, dass der Passagier korrekt identifiziert wurde und mit der richtigen Reservierung verknüpft ist.

Schritt 3: Normal-Case-Szenario finalisieren III

3. Ist der Pass gültig und gehört er zum Passagier?

wird zu

3. Ist der Pass gültig und gehört er zum Passagier?
 1. Der Pass muss aktuell sein.
 2. Das Ende der Gültigkeitszeit darf nicht vor Ende der Reise liegen.
 3. Der Pass muss für Reisen in das Zielland gültig sein.
 4. Visas (soweit erforderlich) müssen aktuell sein.
 5. Es dürfen keine „Einreise verweigert“-Stempel vom Zielland vorhanden sein.

oder kurz

3. Ist der Pass gültig und gehört er zum Passagier?
Siehe Verfahrensvorgaben EU-157.

Schritt 4: Ausnahmen und Alternativen

- Typischerweise gibt es Abläufe, die vom normalen Verlauf des Szenarios abweichen
 - Alternative Abläufe: Zulässige und vorgesehene Änderungen am Ablauf, beispielsweise begründet durch Optionen für die Akteure
 4. Vielfliegernummer mit der Reservierung verknüpfen.
A4.1 Wechsel zu einer Vielfliegernummer einer Partnerfluglinie zulassen.
 - Ausnahmen: Unerwünschte aber unvermeidbare Abweichungen
 5. Einen Sitzplatz bestimmen.
E5.1 Der Wunschplatz des Passagiers ist nicht verfügbar.
E5.2 Für den Gate-Agent einen Platzwechselwunsch erfassen.

Ein fertiger Business Use-Case I

Business Event: Passagier will einchecken.

Business Use-Case-Name: Passagier auf Flug einchecken

Trigger: Check-In-Anfrage mit Ticket, Buchungsnummer oder Identität und Flugnummer des Passagiers

Vorbedingungen: Passagier muss eine Reservierung haben.

Interessierte Stakeholder: Passagier, Check-In Agent, Marketing, Gepäcktransport, Reservierungen, Flugbeladungssystem, Security, Immigrationsbehörde im Zielland, Betriebsablauf

Aktive Stakeholder: Passagier (Auslöser), Check-In Agent

Normaler Ablauf:

1. Die Reservierung des Passagiers finden
2. Sicherstellen, dass der Passagier korrekt identifiziert wurde und mit der richtigen Reservierung verknüpft ist.
3. Ist der Pass gültig und gehört er zum Passagier?
Siehe Verfahrensvorgaben EU-157.
4. Vielfliegernummer mit der Reservierung verknüpfen.

Ein fertiger Business Use-Case II

5. Einen Sitzplatz bestimmen.
6. Die korrekten Antworten auf die Sicherheitsfragen bekommen.
7. Gepäck auf den Flug einchecken.
8. Bordkarte und Gepäckabholzettel erzeugen und übergeben.
9. Dem Passagier einen angenehmen Flug wünschen.

Alternativen: ...

Ausnahmen: ...

Ergebnis: Der Passagier ist als für den Flug eingekennzeichnet registriert, das Gepäckstücke sind dem Flug zugewiesen, ein Sitzplatz ist zugeteilt und der Passagier ist im Besitz der Bordkarte und der Gepäckabholzettel.

Szenariobeschreibung mit strukturiertem Text

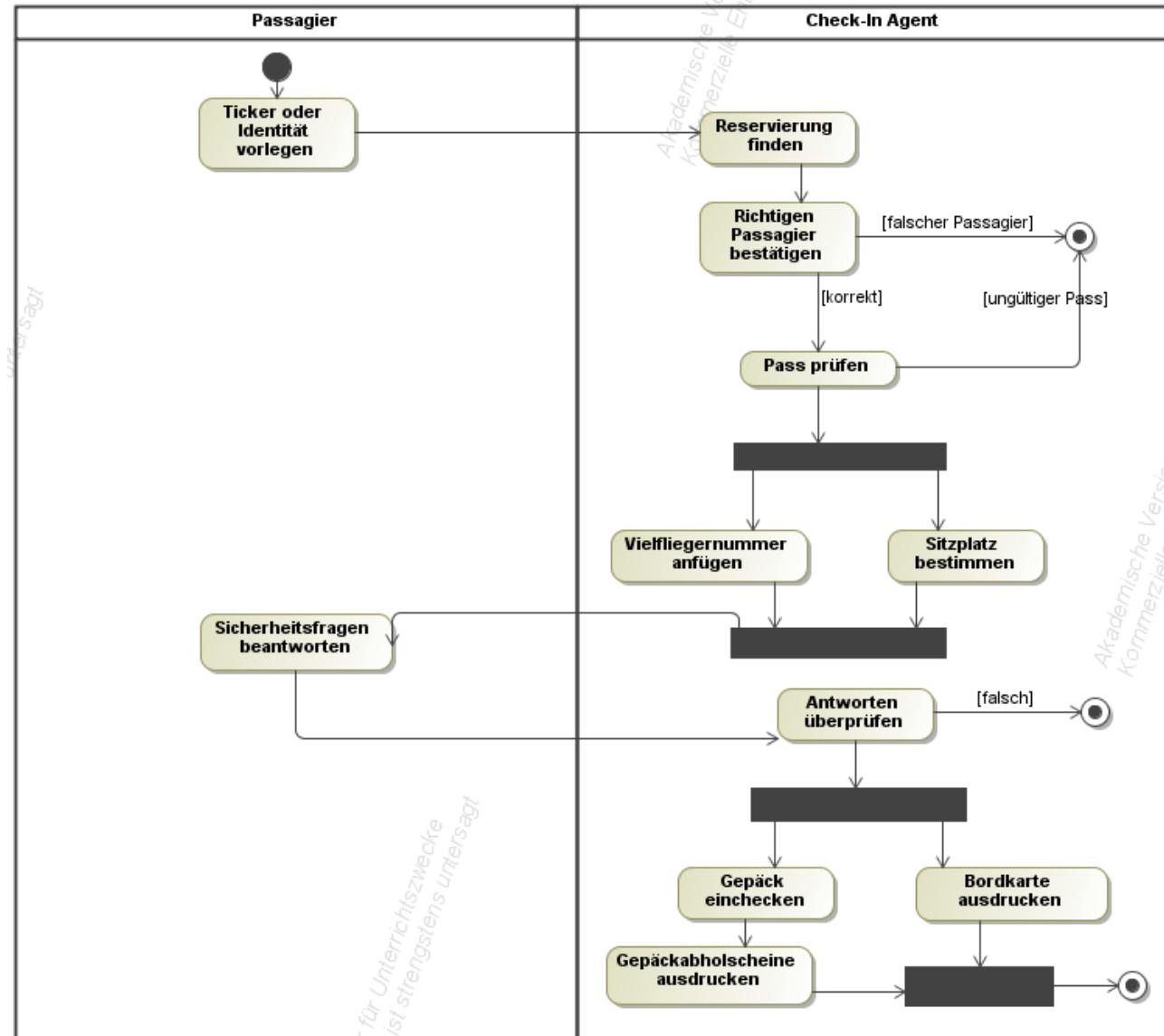


Institute for
Software & Systems
Engineering

- Flexibel und ausdrucksmächtig
- Von Anwendungsexperten les- und schreibbar
- Präziser als freier Text, weniger Auslassungen und Fehler

- Missverständnisse nicht ausgeschlossen
- Zusammenhänge mit anderen Szenarien / Anwendungsfällen werden nicht erfasst

Szenariobeschreibung mit UML-Aktivitätsdiagrammen



Szenariobeschreibung mit UML-Aktivitätsdiagrammen

- Schritte als **actions** modellieren
- Trennung von Aktion (durch Akteur) und **Reaktion** (des Systems) möglich
- Modelliert Normal- und Fehlerfälle

Szenariobeschreibung mit UML-Aktivitätsdiagrammen

- Aussagekräftig
- Wählbarer Grad an Präzision (bei der Beschreibung der Ereignisse und Aktionen)

- Für Anwendungsexperten verständlich (erfordert aber Unterstützung oder Ausbildung)

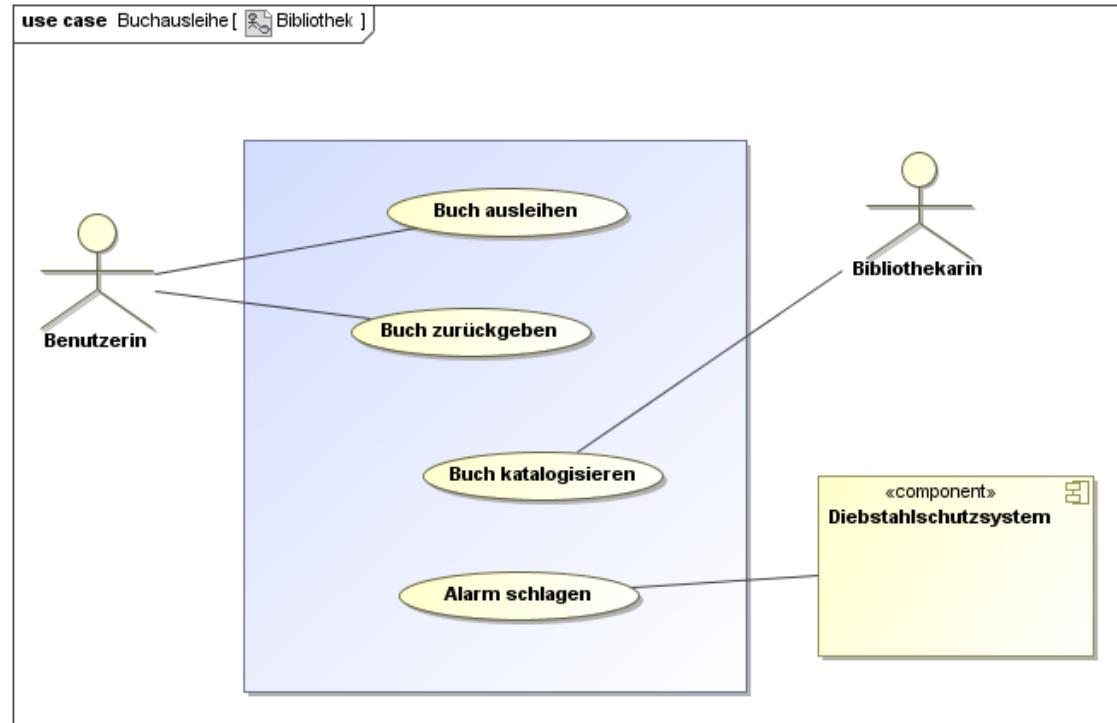
- Braucht Modellierungsexperten für die Erstellung

Abschnitt 1.11.2.5

USE-CASE DIAGRAMM

Übersicht über die Use-Cases

- UML Use-Case Diagramm
- Überblick über alle **Anwendungsfälle** (Typszenarien) eines Systems
- Ist eine Art **Kontextdiagramm**



UML Use-Case Diagramm

- Zeigt, welche Akteure in welchen Anwendungsfällen mit dem System interagieren
- Bei kleinen Modellen übersichtlich

- Modelliert Zusammenhänge zwischen Anwendungsfällen nur rudimentär
- Modelliert die Inhalte der Anwendungsfälle nicht
- Kennt keine Zerlegung

Abschnitt 1.11.2.6

VERFEINERUNG ZUM PRODUCT USE-CASE

- Ausgehend vom Business Use-Case wird der **Product Use-Case** bestimmt
- Dazu gilt es, die Constraints zu bewerten, denen das Projekt unterliegt
- Die Entscheidung erfolgt im Konsens mit den Stakeholdern
- Faktoren sind
 - Geschäftsziele des Projekts
 - Technische Auswirkungen der Alternativen
 - Auswirkungen der Alternativen auf geschäftliche Abläufe
 - Projektrandbedingungen

Product Use-Case ist ein Ausschnitt des Business Use-Case und wird mit einem Product Use-Case Szenario beschrieben

Product Use-Case-Name: Passagier auf Flug einchecken

Trigger: Name des Passagiers und seine Passnummer

Vorbedingungen: Passagier muss eine Reservierung haben.

Interessierte Stakeholder: Passagier, Check-In Agent, Marketing, Gepäcktransport, Reservierungen, Flugbeladungssystem, Security, Immigrationsbehörde im Zielland, Betriebsablauf

Akteur: Check-In Agent

Normaler Ablauf:

1. Die Reservierung des Passagiers finden
2. Vielfliegernummer mit der Reservierung verknüpfen.

5. Einen Sitzplatz bestimmen.
6. Gepäck auf den Flug einchecken.
7. Bordkarte und Gepäckabholzettel erzeugen und übergeben.

Alternativen: ...

Ausnahmen: ...

Ergebnis: Der Passagier ist als für den Flug eingemeindet registriert, das Gepäckstücke sind dem Flug zugewiesen, ein Sitzplatz ist zugeteilt und der Passagier ist im Besitz der Bordkarte und der Gepäckabholzettel.

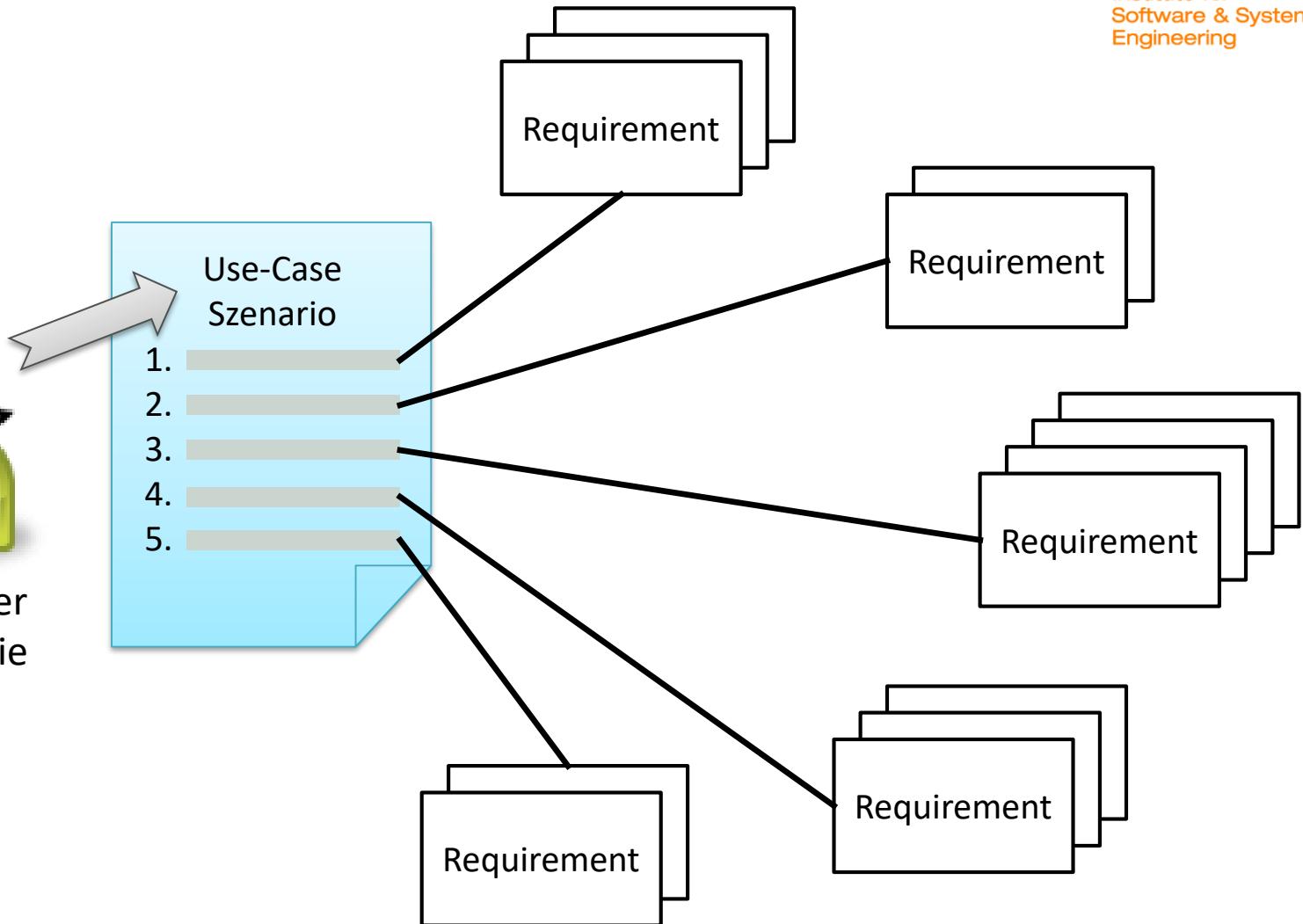
Abschnitt 1.11.3

ATOMARE ANFORDERUNGEN

Atomare Anforderungen



Stakeholder
erklären die
Arbeit



Vom Product Use-Case zu atomaren Anforderungen

- Product Use-Case Szenario analysieren
- Was muss das Produkt können, damit ein Schritt des Szenarios umgesetzt wird? Dies ergibt **die atomaren funktionalen Anforderungen**
- Jede notwendige Aktivität des Produkts ergibt eine atomare Anforderung
- Formulieren als **ein Satz mit einem Verb**

Das Produkt muss den ausgewählten Artikel zum Einkaufswagen hinzufügen.

- Auch **Qualitätsanforderungen** erfassen! Müssen in dem Schritt bspw. Reaktionszeiten eingehalten werden?

Beispiel: Enteisungsplan erstellen

Szenario:

1. Bediener wählt Planungsdatum und Gebietsbezeichner aus
2. Produkt wählt die relevanten Temperaturverteilungskarten aus

Atomare Requirements für 1:

- Das Produkt muss die Eingabe des Planungsdatums akzeptieren
- Das Produkt muss einen gültigen Gebietsbezeichner akzeptieren

Die Einzelschritte werden ggf. noch verfeinert, bis Unklarheiten beseitigt sind.

Atomare Requirements für 1:

- Das Produkt muss die Eingabe des Planungsdatums akzeptieren
- Das Produkt muss warnen, wenn das Planungsdatum weder heute noch morgen ist
- Das Produkt muss einen gültigen Gebietsbezeichner akzeptieren
- Das Produkt muss überprüfen, ob das Gebiet in dem Bereich liegt, für dessen Enteisung die Einrichtung zuständig ist
- Das Produkt muss sicherstellen, dass das Gebiet dasjenige ist, das der Bediener bearbeiten will

- Nicht-funktionale Anforderungen sind **oft** (aber nicht immer) nicht direkt an einen Schritt eines Szenarios gebunden.
- Sie beschreiben **oft** Eigenschaften **des Produkts als Ganzes**.
- Suche durch Prüfung verbreiteter Klassen von nicht-funktionalen Anforderungen.

Kategorien nicht-funktionaler Anforderungen

- **Look and Feel:** Stil und Stimmung der optischen Erscheinung.

Das Produkt muss mit den Firmenstandards zur Markenoptik verträglich sein.

- **Benutzbarkeit und menschliche Faktoren:** Usability-Requirements sichern, dass das Produkt den Fähigkeiten und Erwartungen an die Verwendbarkeit entspricht.

Das Produkt muss für Mitglieder der Allgemeinheit, die möglicherweise nicht Deutsch sprechen, leicht zu bedienen sein.

Beispiel: Motivation und Erfolgsmaß

Das Produkt soll einfach zu lernen sein.



Das Produkt soll für ein Mitglied der Allgemeinheit ohne Training bei der ersten Verwendung leicht zu benutzen sein.

Begründung: Potentielle Benutzer haben möglicherweise diese Art Produkt nie vorher verwendet.

Erfolgsmaß: 90% einer für die Allgemeinheit repräsentativen Testgruppe muss innerhalb von 45 Sekunden nach der ersten Begegnung mit dem Produkt erfolgreich ein Ticket gekauft haben.

Kategorien nicht-funktionaler Anforderungen

- **Leistung:** Zeit, Geschwindigkeit, Durchsatz, Genauigkeit bei der Leistungserbringung.

Das Produkt muss die Kapazität für 5000 Straßen besitzen.

Begründung: Die maximale Zahl von Straßen im Zuständigkeitsbereich aller denkbaren Kunden.

- **Betriebs- und Umweltbedingungen:** Wo und wie muss das Produkt betrieben werden können.

Das Produkt muss mit der Datenbank des Thermalkartendienstes zusammenarbeiten.

Das Produkt muss einen Sturz aus 1m Höhe überstehen.

Kategorien nicht-funktionaler Anforderungen

- **Wartung und Betriebssupport:** Anforderungen aus zukünftiger Wartung oder Weiterentwicklung sowie Anforderungen des Produktsupports.

Das Produkt muss leicht auf Linux portiert werden können.

Begründung: Linux wurde als möglicher Zukunftsmarkt erkannt.

- **Sicherheit:** Vertraulichkeit, Integrität und Verfügbarkeit von Daten.

Das Produkt muss sicherstellen, dass seine Straßentemperaturdaten den von den Wetterstationen übermittelten Werten entsprechen.

- **Kulturelle und politische Aspekte:** Vermeiden, dass das Produkt wegen sprachlichen, religiösen oder kulturellen Aspekten, Tabus, Vorurteilen oder Bräuchen in manchen Regionen abgelehnt wird.

Das Produkt darf keine Wörter oder Symbole verwenden, die irgendwo auf der Welt beleidigend sind.

- **Rechtliche Requirements:** Gesetze und Vorschriften, die den Einsatz des Produkts betreffen.

Das Produkt muss den Vorschriften des Behindertengleichstellungsgesetzes genügen.

- Mit Metriken erreicht man Messbarkeit für abstrakte Begriffe wie Performanz, Benutzbarkeit oder Wartbarkeit
- Nicht alle Metriken lassen sich vollautomatisch berechnen

Metrik

Nach IEEE 1061 ist eine Metrik „*a function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality.*“

Institute of Electrical and Electronics Engineers: IEEE Standard for a Software Quality Metrics Methodology (IEEE Std 1061-1998)

- Funktionale und nicht-funktionale Anforderungen brauchen ein Erfolgsmaß.
- Erfolgsmaß ist ein objektiv prüfbares Kriterium.
- Wichtig für Abnahmetests aber auch für die Entwickler, da diese dadurch einen Standard haben, den sie mit der Lösung anstreben.
- Formulieren bei funktionalen Anforderungen eher einfach, bei nicht-funktionalen schwierig.
- Formulieren des Erfolgsmaßes hilft gerade bei nicht-funktionalen Anforderungen, diese überhaupt erst zu verstehen.

Abschnitt 1.12

EINZELANFORDERUNGEN DOKUMENTIEREN

Aspekte bei der Formulierung von Anforderungen

Darstellungsaspekte

- Drei Perspektiven für funktionale Anforderungen: Funktionalität, Daten und Verhalten
- Attribute: Spezielle Qualitäten, Leistung und Randbedingungen

- Entscheidungsspielraum:
 - Beschreibungsmittel
 - Konstruktive oder deskriptive Darstellung
 - Strukturierung / Aufbau
 - Präzision
 - Detaillierung

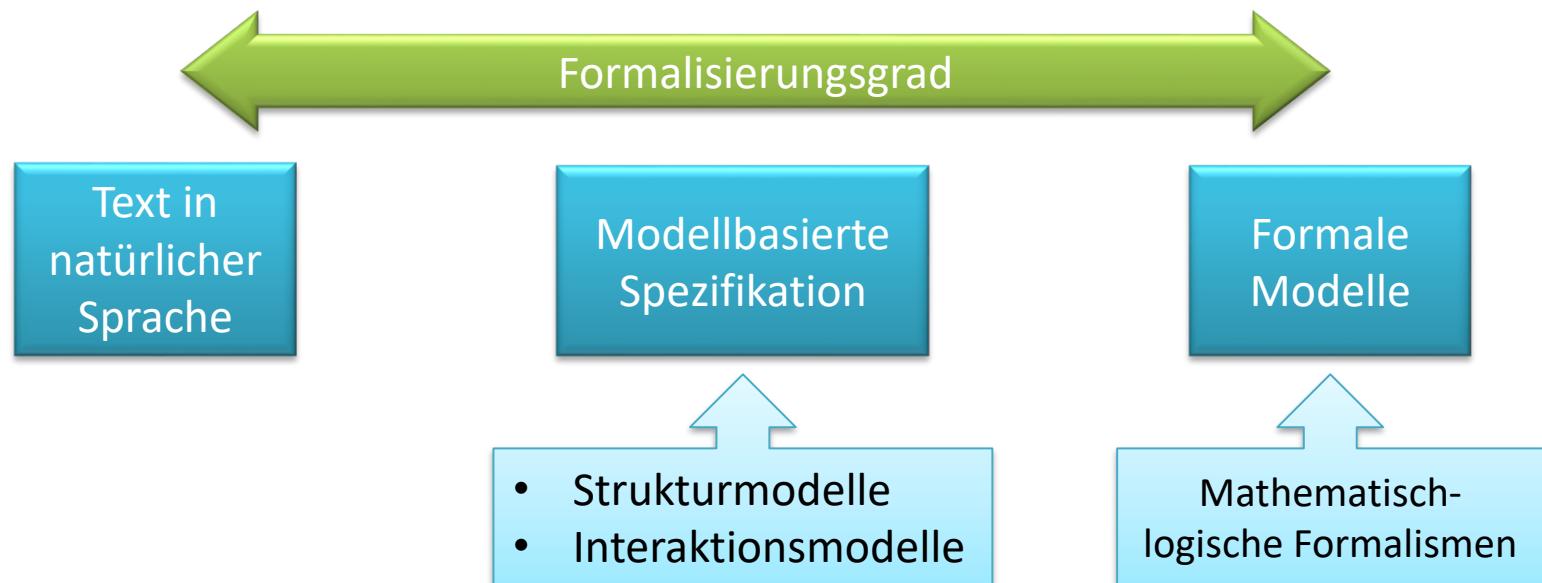
K. Pohl, C. Rupp: Basiswissen Requirements Engineering, dpunkt.verlag,
2009

Welche Informationen sind darzustellen?

Funktionale Anforderungen

- **Daten**: Statische Struktur der Daten, Verwendung, Erzeugung, Speicherung, Änderung
- **Funktionen**: Manipulation der Daten im System, Eingabe, Verarbeitung und Ausgabe von Daten
- **Verhalten**: Verhalten im Sinne der Reaktion auf externe Stimuli, sichtbares dynamisches Systemverhalten

- Große **Bandbreite** an Beschreibungsmitteln von informellem Text bis mathematisch-logische Notation
- Ein wesentlicher Unterschied: Unterschiedlicher **Grad an Formalität**



- Anforderungen **sprachlich präzise** ausdrücken, z.B.
 - Eindeutige Bezeichner
 - Definierte Subjekte, keine Passivkonstruktionen, kein „man“, kein „es wird“
 - Festgelegte Semantik für Verben wie „kann“, „muss“, „soll“,...
- Anforderungen **inhaltlich präzise** beschreiben
 - Eindeutige Formulierungen
 - Ausreichend Details
 - Prüfbar
 - Testbar
 - Messbar, bei quantifizierten Angaben

Beispiele

Vage Anforderungen

„schnell“

„Für alle Kurse müssen rechtzeitig Teilnehmerlisten verfügbar sein.“

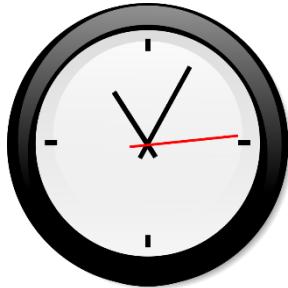
Präzise Anforderungen

„unter 5 ms“

„10 Arbeitstage vor Kursbeginn erstellt das System für jeden Kurs eine Teilnehmerliste.“

Abschnitt 1.12.1

ANFORDERUNGEN IN NATÜRLICHER SPRACHE



Time flies like an arrow;
fruit flies like a banana



- Bei der Wissensdarstellung kann es durch **Darstellungstransformationen** zu Diskrepanzen zwischen dem mentalen Modell des Sprechers und der sprachlichen Darstellung kommen
- Man unterscheidet
 - **Tilgung**: Indikator für unvollständige Informationen
 - **Generalisierung**: Indikator für falsche Verallgemeinerungen
 - **Verzerrung**: Indikator für verfälschende Aussagen
- Werden solche sprachlichen Effekte nicht beseitigt, können die formulierten Anforderungen unvollständig oder missverständlich sein

Qualität natürlichsprachlicher Spezifikationen

Die Qualität natürlichsprachlicher Anforderungsspezifikationen kann systematisch verbessert werden durch:

- Geeignete **Strukturierung** des Dokuments
- Regeln zur **sprachlichen Formulierung** von Anforderungen
- Kontrollierten Umgang mit **Redundanz**
- Konsequente Verwendung eines **Glossars**

- Regel: Anforderungen im **Aktiv** formulieren, mit definiertem Subjekt
- Regel: Aktivitäten (Prozesse) durch ein **Vollverb** beschreiben mit eindeutiger/definierter Bedeutung
- Regel: **Nominalisierungen** hinterfragen und wenn möglich auflösen

Nominalisierungen **sind** aus Verben gebildete Nomen. Eine Nominalisierung macht aus einem (länger dauernden) Prozess ein singuläres Ereignis. Dies kann zu Verständnisproblemen führen.

- Regel: **Funktionsverbgefüge auflösen**

Funktionsverbgefüge sind Verbkonstruktionen, deren semantischer Inhalt weitgehend durch ein Substantiv bestimmt wird und in denen das Verb nur als Funktionsverb fungiert.

- Regel: Einzelanforderungen in **kleine Einheiten fassen**:
Pro Einzelanforderung (Prozesswort) einen Satz
formulieren **Beim Formulieren helfen Satzschablonen!**
- Regel: Fehlende Informationen zum Vollverb finden
und ergänzen

Stelle typische W-Fragen: Was? Wie? Wem? Wann? Wie oft?

- Regel: Prüfen, ob Informationen zu beschriebenen **Eigenschaften** fehlen

Eigenschaften treten als Adjektive oder Adverbien auf. Diese werden mit den typischen W-Fragen untersucht.

- Regel: Eigenschaften müssen mess- und testbar formuliert sein. Für eine vollständige Beschreibung benötigt man einen **Bezugspunkt**

- Regel: Nicht-funktionale Aspekte in eigenständigen Anforderungen formulieren, wenn der nicht-funktionale Aspekt eigenständig behandelt werden soll (getestet, gewartet, entwickelt, ...) oder er übergreifend für mehrere Funktionen gefordert wird

- Regel: **Universalquantoren** (Zahl- und Mengenwörter) hinterfragen

All-Quantifizierungen („jeder“, „alle“, „immer“, ...) und Ausschlüsse („nie“, „keiner“, „entweder-oder“, ...)

- nach Ausnahmen hinterfragen
- gefundene Ausnahmen als Anforderungen spezifizieren

- Regel: **Fehlende** Zahl- und Mengenwörter klären
- Regel: **Nomen mit unspezifischer Bedeutung** („die Daten“, „die Anzeige“, „der Kunde“, ...) hinterfragen

- Regel: Anforderungen, die bestimmtes Verhalten als **möglich** oder **unmöglich** festlegen, müssen hinterfragt werden

Wird nur gefordert, dass ein Verhalten möglich oder unmöglich sein soll wurde möglicherweise der Teil der fachlichen Logik getilgt, der diese Forderung realisiert.

- Regel: Nebensätze mit für die Anforderung **nicht essentiellen Informationen in eigene Sätze extrahieren**
- Regel: **Redundanz und ausschmückende Floskeln vermeiden**

Überflüssiger Text vergrößert die Spezifikation und bedeutet mehr Aufwand (Erstellung, Prüfung, ...) ohne Mehrwert.

- Regel: **Abweichungen** vom **Normalverhalten** untersuchen

In der Regel muss auch beschrieben werden, was geschehen soll, wenn das gewünschte Verhalten nicht sichergestellt werden kann.

- Regel: **Unvollständige Bedingungsstrukturen** untersuchen

Verhalten unter einer Bedingung erfordert mindestens auch die Angabe, was passieren soll, wenn die Bedingung nicht zutrifft.

- Regel: **Implizite Annahmen** untersuchen

Oft stecken in Beschreibungen oft nur beiläufig erwähnte Annahmen, die wahr sein müssen, damit die Beschreibung sinnvoll ist.

- Satzschablonen sind ein einfacher, leicht zu erlernender Ansatz, sprachliche Effekte zu reduzieren
- Satzschablonen unterstützen den Autor, semantische Eindeutigkeit zu erreichen

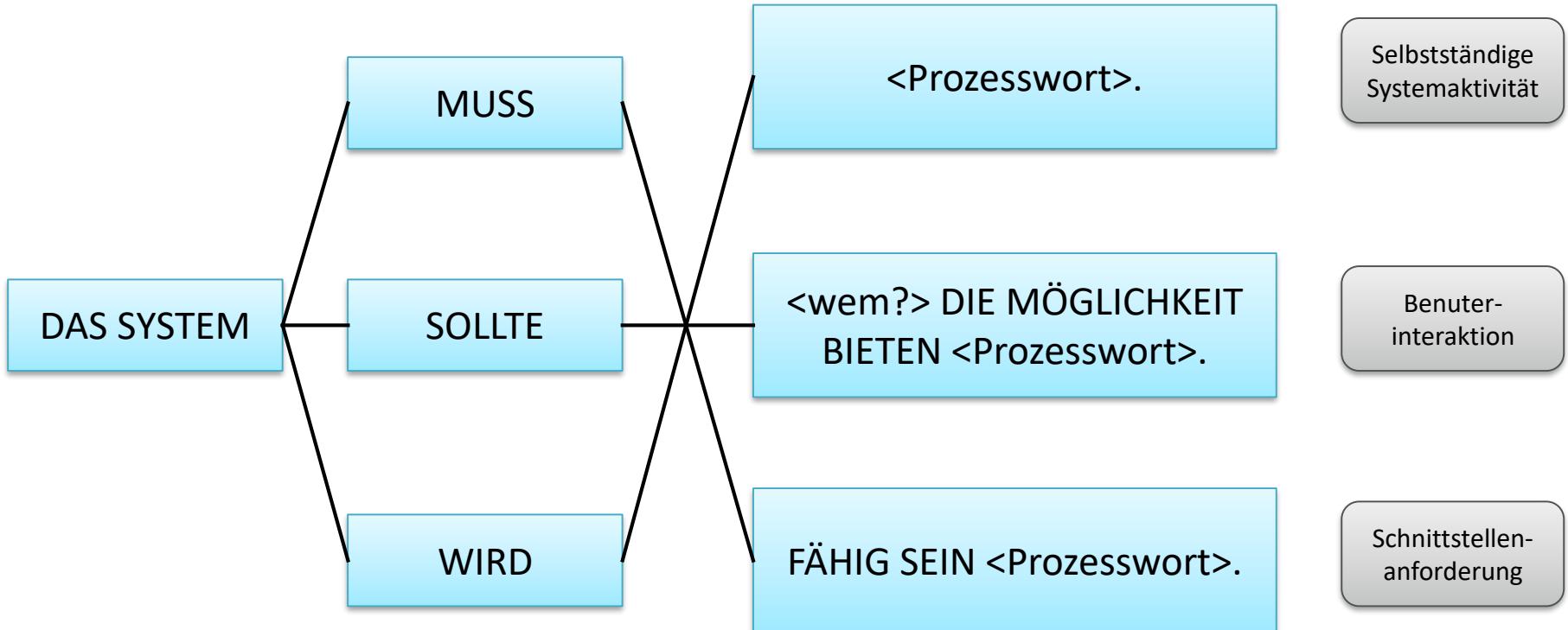
Satzschablone

Eine Satzschablone oder Requirements Template ist ein Bauplan für die syntaktische Struktur einer einzelnen Anforderung.

Einsatz

1. Festlegen der **rechtlichen Verbindlichkeit**: Dies ergibt Modalverben *muss*, *soll* oder *wird*
2. **Kern der Anforderung** (Prozess / Vorgang) ermitteln und durch entsprechendes **Verb** beschreiben
3. **Art der Systemaktivität** bestimmen:
 - Selbständige Systemaktivität
 - Benutzerinteraktion
 - Schnittstellenanforderung

Satzschablonen

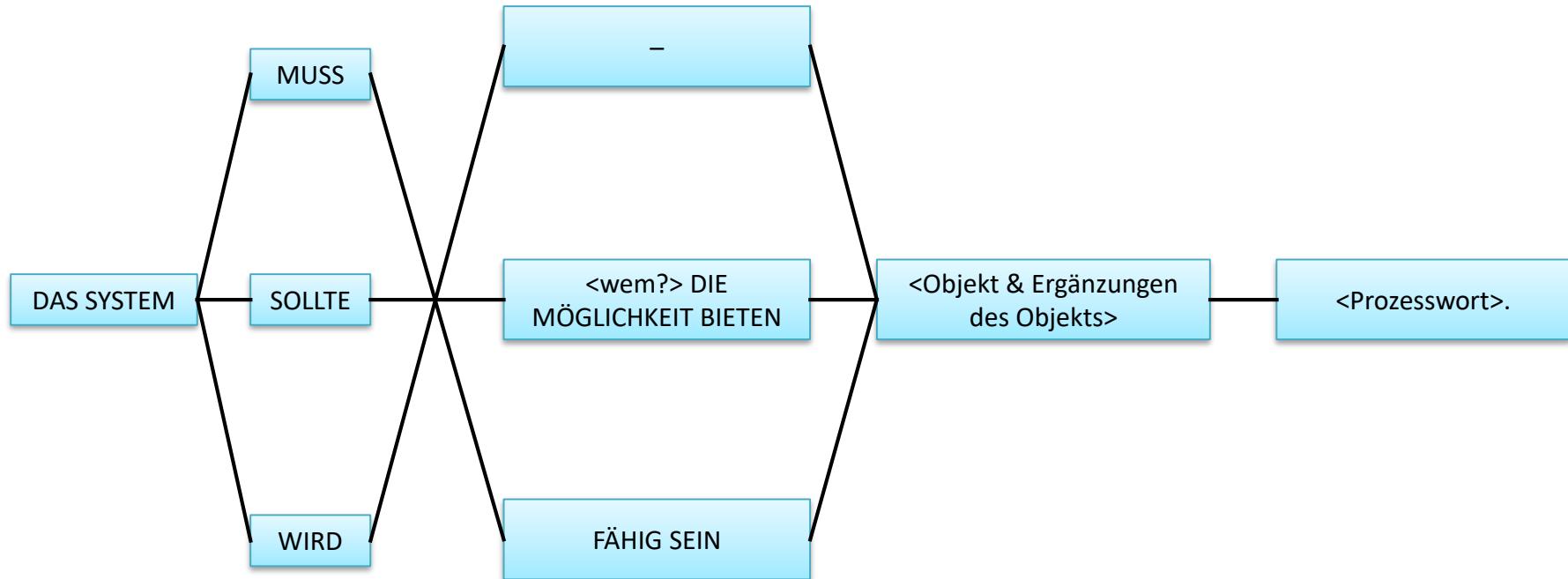


Einsatz

1. Festlegen der **rechtlichen Verbindlichkeit**: Dies ergibt Modalverben *muss*, *soll* oder *wird*
2. **Kern der Anforderung** (Prozess / Vorgang) ermitteln und durch entsprechendes **Verb** beschreiben
3. **Art der Systemaktivität** bestimmen:
 - Selbständige Systemaktivität
 - Benutzerinteraktion
 - Schnittstellenanforderung
4. **Objekte** einfügen

Satzschablonen

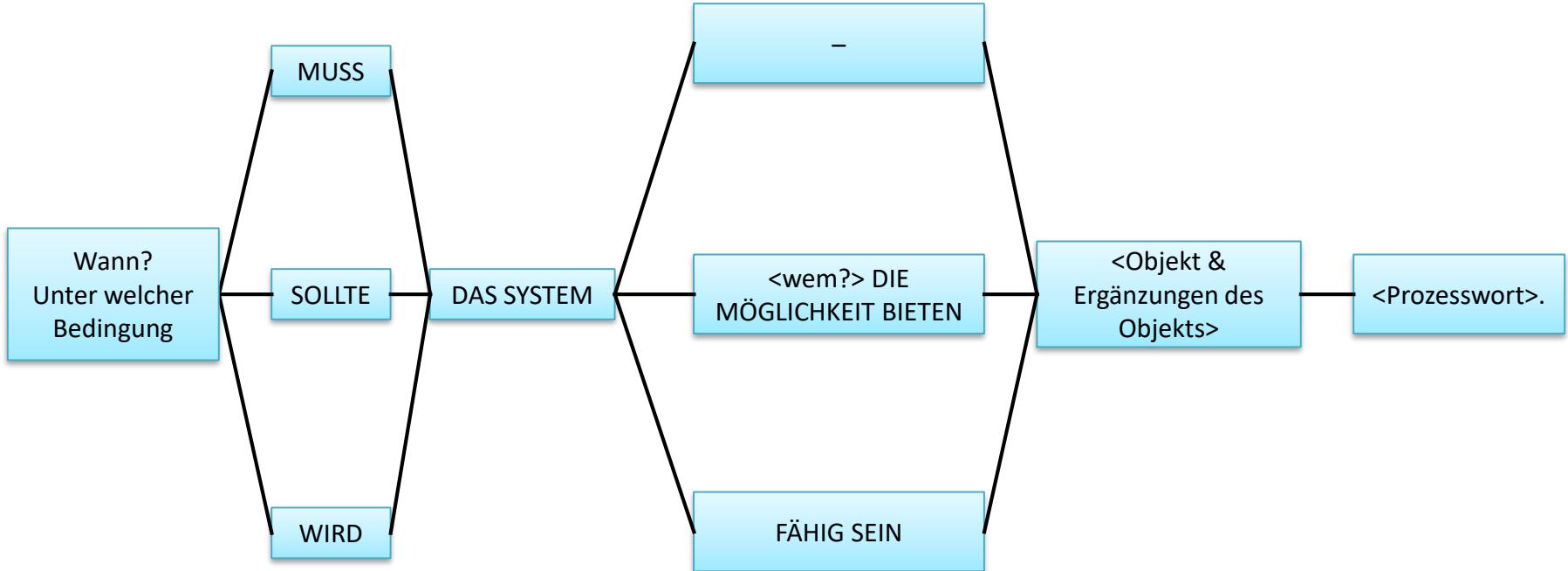
Objekte



Einsatz

1. Festlegen der **rechtlichen Verbindlichkeit**: Dies ergibt Modalverben *muss*, *soll* oder *wird*
2. **Kern der Anforderung** (Prozess / Vorgang) ermitteln und durch entsprechendes **Verb** beschreiben
3. **Art der Systemaktivität** bestimmen:
 - Selbständige Systemaktivität
 - Benutzerinteraktion
 - Schnittstellenanforderung
4. **Objekte** einfügen
5. **Bedingungen** ergänzen

Bedingungen



- Die Satzschablonen implizieren, dass eine Anforderung mit genau einem Satz formuliert werden kann. Anforderungen sind aber oft vielschichtiger, benötigen ergänzende Erläuterungen oder Beschreibungen
- Festgelegte Semantik für „muss“, „soll“ und „wird“ stellt implizite Priorisierung dar. Prioritäten sind aber Eigenschaften von Anforderungen, nicht Teil davon. Prioritäten sollten in den Metadaten einer Anforderung verwaltet werden.

Rahmen für Einzelrequirement

Rq-Nr.:

Rq-Typ:

Event/Use-Case-Nr.:

Beschreibung:

Begründung:

Ursprung:

Erfolgsmaß:

Kundenzufriedenheit:

Kundenmissfallen:

Konflikte:

Priorität:

Unterstützende Dokumente:

Entwicklung:

Beispiel

Rq-Nr.: 75

Rq-Typ: 9

Event/Use-Case-Nr: 6

Beschreibung: *Das Produkt muss einen Fehler melden, falls eine Wetterstation keine Daten übermittelt.*

Begründung: *Ausfall einer Datenübertragung kann ein Hinweis sein, dass die Wetterstation defekt ist und gewartet werden muss, und dass die Daten zur Eisbildungsvorhersage unvollständig sein können.*

Ursprung: *Hans Meier, Straßenbauamt*

Erfolgsmaß: *Liegt die Anzahl der übertragenen Wetterdaten in einer Stunde unter der vom Hersteller spezifizierten minimalen Anzahl von Datenübermittlungen, muss ein Fehler gemeldet worden sein.*

Kundenzufriedenheit: 3

Kundenmissfallen: 5

Konflikte: *Keine*

Priorität: *Hoch*

Unterstützende Dokumente: *Spezifikation der WeMe Wetterstation*

Entwicklung: *Gefunden durch GBS, 28.7.2011*

Regeln für das Formulieren von Anforderungen – Top 5



Institute for
Software & Systems
Engineering

- Nominalisierungen auflösen
- W-Fragen stellen
- Unpräzise Substantive hinterfragen
- Unvollständige Bedingungsstrukturen untersuchen
- Implizite Annahmen untersuchen

- Ausdrucksmächtig
- Ohne zusätzliche Ausbildung les- und schreibbar
- Viele Anforderungen nur mit natürlicher Sprache ausdrückbar

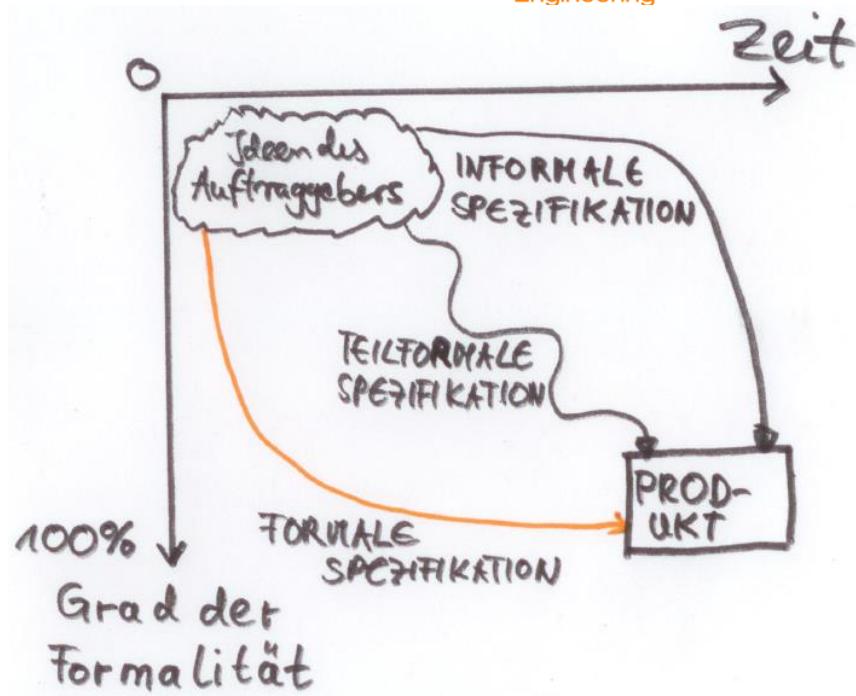
- Unübersichtlich
- Fehlerträchtig und schwierig zu prüfen
- Mit zunehmender Größen wachsen Probleme überproportional

Heute das am häufigsten verwendete Mittel zur Formulierung von Anforderungen

Abschnitt 1.12.2

FORMALE SPEZIFIKATION

- Das Problem analysieren
- Anforderungen formal spezifizieren
- Implementierung durch korrektheitserhaltende Transformation
- Pflege der Anforderungs-spezifikation, nicht des Codes



Was bedeutet „formal“?

- **Formale Sprache**, d.h. Verwendung einer Spezifikationssprache mit
 - formal definierter **Syntax** und
 - formal definierter **Semantik**
- Primär zur Spezifikation funktionaler Anforderungen

Welche Ausprägungen gibt es?

- Rein deskriptiv, zum Beispiel **Algebraische Spezifikationen**
- Rein konstruktiv, zum Beispiel **Petrinetze**
- Modellbasierte Mischformen, zum Beispiel **OCL**, **VDM** und **Z**

- Mathematisches Modell des **Systemzustands** und seiner **Veränderungen**
- Basierend auf **Mengen**, **Relationen** und **logischen Ausdrücken**

Beschreibung von:

- Grundmengen
- Zusammenhängen zwischen Mengen (Relationen, Funktionen)
- Invarianten (Prädikate)
- Zustandsveränderungen (Relationen, Funktionen)
- Zusicherungen für Zustände

Bekannte Vertreter:

- **VDM** (Vienna Development Method; Björner und Jones 1978)
- **Z** (Spivey 1992)
- **OCL** (ab 1997; OMG 2006)
- **Alloy** (Jackson 2002)
- **B** (Abrial 2009)

- Hier nur ein grober Überblick und einige Beispiele
- Viel einschlägige Literatur verfügbar
- Heute wahrscheinlich die in der Praxis am häufigsten verwendete formale Spezifikationssprache

- J.M. Spivey: The Z Notation: A Reference Manual; Prentice Hall, 1992. Online unter: <http://spivey.orient.ox.ac.uk/~mike/zrm/>
- J. Woodcock, J. Davies: Using Z — Specification, Refinement, and Proof; Prentice Hall, 1996. Online unter: <http://www.usingz.com/>
- J. Jacky: The Way of Z: Practical Programming with Formal Methods; Cambridge University Press, 1997

Die Grundelemente von Z

- Z basiert auf Mengen
- Eine Spezifikation besteht aus Mengen, Typen, Axiomen und Schemata

Basic Types sind Grundmengen: $[Name]$ $[Datum]$ \mathbb{N}

Variablen (und Ausdrücke) haben einen Typ:

Personen: P Name Zähler: \mathbb{N}

Axiome definieren globale Variablen und deren (invariante) Eigenschaften:

$string: seq\ CHAR$

$\#string \leq 64$

\mathbb{N} Menge der natürlichen Zahlen seq Sequenz von Elementen

$P M$ Menge aller Teilmengen von M $\#M$ Anzahl Elemente der Menge M

Die Grundelemente von Z

- Z basiert auf Mengen
- Eine Spezifikation besteht aus Mengen, Typen, Axiomen und Schemata

Basic Types sind Grundmengen: $[Name]$ $[Datum]$ \mathbb{N}

Variablen (und Ausdrücke) haben einen Typ:

Personen: P Name Zähler: \mathbb{N}

Axiome definieren globale Variablen und deren (invarianten) Eigenschaften:

$string: seq \text{ CHAR}$

$\#string \leq 64$

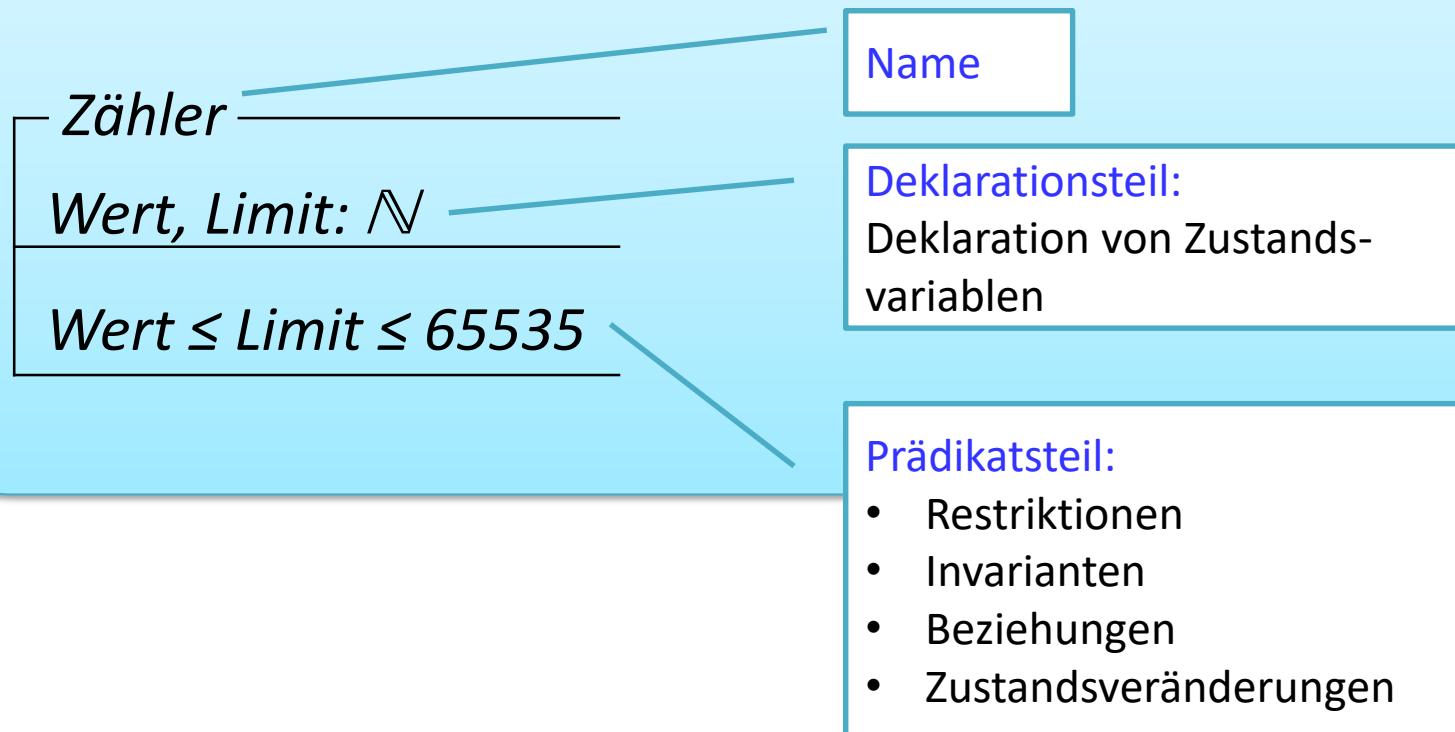
Deklaration

Invariante

\mathbb{N} Menge der natürlichen Zahlen seq Sequenz von Elementen

$P M$ Menge aller Teilmengen von M $\#M$ Anzahl Elemente der Menge M

Schemata gliedern eine Z-Spezifikation und bilden jeweils einen eigenen Namensraum:



Relationen und Funktionen sind Mengen geordneter Tupel:

Bestellung: $\mathcal{P}(Teil \times Lieferant \times Datum)$

Eine Teilmenge aller geordneten Tripel (t, l, d) mit $t \in \text{Teil}$,
 $l \in \text{Lieferant}$, $d \in \text{Datum}$

Geburtstag: Person \rightarrow Datum

Eine Funktion, die jeder Person genau ein Datum als Geburtstag zuordnet

Zustandsveränderungen durch **Operationen**:

— Inkrementieren —

Δ Zähler

$Wert < Limit$

$Wert' = Wert + 1$

$Limit' = Limit$

ΔS Die in S definierten Variablen werden geändert

M' Zustand der Variablen M nach der Operation

Logische Gleichheit;
keine Zuweisung

Spezifikation eines Bibliothekssystems

Die Bibliothek hat einen Bestand an Büchern und eine Menge von Personen als Benutzer.

Bücher aus dem Bestand können von Benutzern ausgeliehen sein.

Bibliothek ——————

Bestand: \mathcal{P} Buch

Benutzer: \mathcal{P} Person

ausgeliehen: Buch \rightarrow Person

dom ausgeliehen \subseteq Bestand

ran ausgeliehen \subseteq Benutzer

→ Partielle Funktion
dom Definitionsbereich...
ran Wertebereich...
...einer Relation

Beispiel II

Spezifikation eines Bibliothekssystems

Nicht ausgeliehene Bücher aus dem Bestand können von Benutzern ausgeliehen werden.

Ausleihen

$\Delta \text{Bibliothek}$

$\text{auszuleihendesBuch?}: \text{Buch}$

$\text{Ausleiher?}: \text{Person}$

$\text{auszuleihendesBuch?} \in \text{Bestand} \setminus \text{dom ausgeliehen}$

$\text{Ausleiher?} \in \text{Benutzer}$

$\text{ausgeliehen}' = \text{ausgeliehen} \cup \{\text{auszuleihendesBuch?}, \text{Ausleiher?}\}$

$\text{Bestand}' = \text{Bestand}$

$\text{Benutzer}' = \text{Benutzer}$

$x?$	x ist Eingabeveriable
$a \in X$	a ist Element der Menge X
\setminus	Mengendifferenzoperator
\cup	Vereinigungsoperator

Beispiel III

Spezifikation eines Bibliothekssystems

Es kann erfragt werden, ob ein bestimmtes Buch ausgeliehen ist.

AnfrageObAusleihbar —————

$\exists \text{Bibliothek}$

angefragtesBuch?: Buch

istAusleihbar!: {ja, nein}

angefragtesBuch? \in Bestand

istAusleihbar! = **if** *angefragtesBuch?* \notin **dom** ausgeliehen

then ja **else** nein

x!

x ist Ausgabevariable

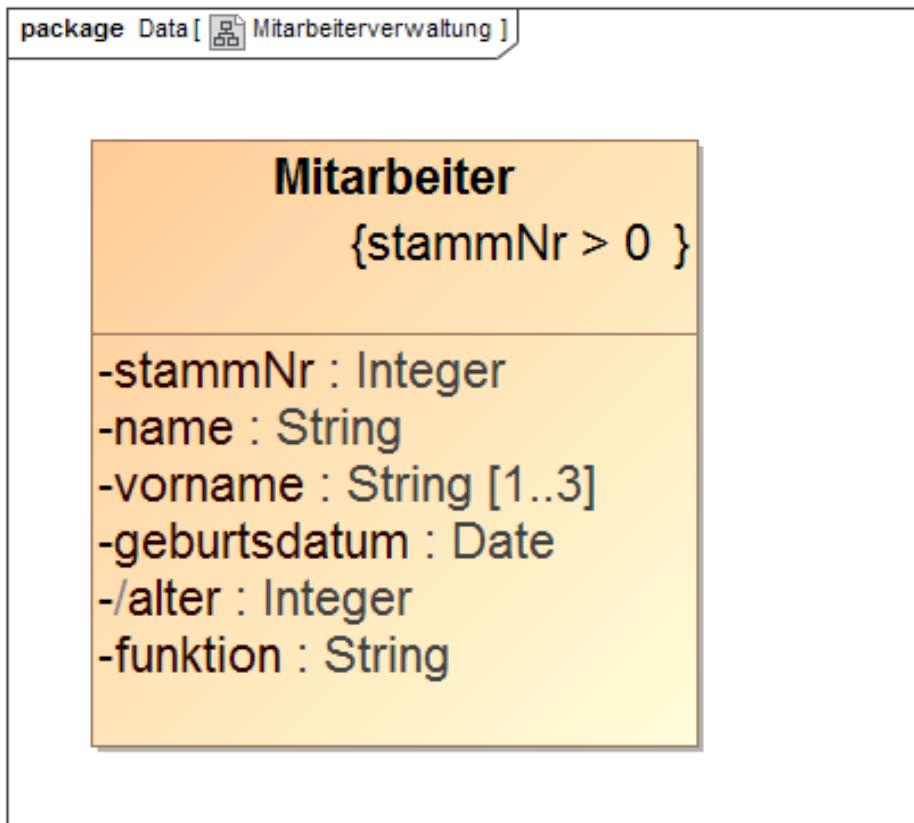
OCL Grundlagen

- OCL ist eine **textuelle formale Sprache**
- Dient der **Präzisierung** von UML-Modellen
- Jeder OCL-Ausdruck steht im **Kontext** eines UML-Modellelements
- Ursprünglich bei IBM als Sprache zur formalen Formulierung von Integritätsbedingungen entwickelt
- 1997 in UML 1.1 integriert
- Aktuell ist **Version 2.4**

Verwendung der OCL

- Spezifikation von **Invarianten** (i.e. zusätzliche Restriktionen) auf UML-Modellen
- Spezifikation der **Semantik von Operationen** in UML-Modellen
- Auch verwendbar als **Anfragesprache** auf UML-Modellen

Invarianten

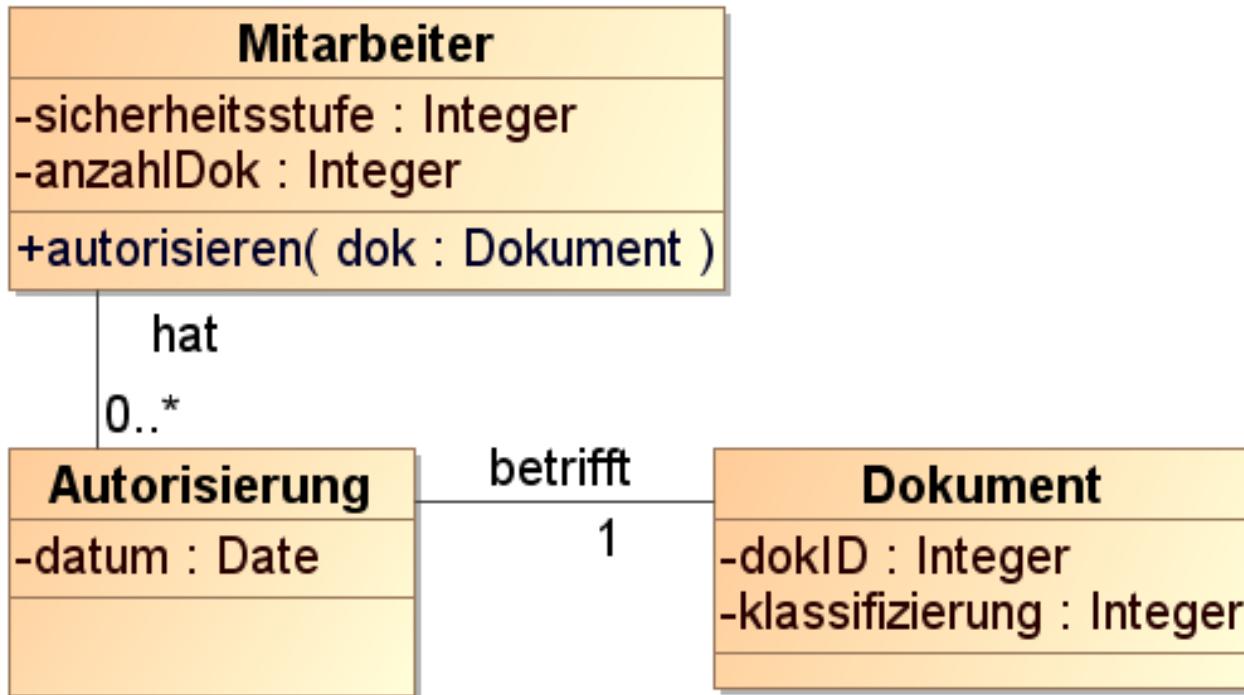


context Mitarbeiterverwaltung::Mitarbeiter **inv:**
self.funktion = "Fahrer" **implies** self.alter ≥ 18

- OCL-Ausdruck ist Bestandteil eines UML-Modellelementes
- Kontext für OCL-Ausdruck ist implizit gegeben

- OCL-Ausdruck wird separat aufgeschrieben
- Kontext muss explizit spezifiziert werden

Semantik von Operationen



context Mitarbeiter::autorisieren (dok: Dokument)

pre self.sicherheitsstufe \geq dok.klassifizierung

post anzahldok = anzahldok@pre + 1 **and**
self.hat->exists (a:Autorisierung | a.betrifft = dok) **and**
self.sicherheitsstufe = self.sicherheitsstufe@pre

Navigation, Aussagen über Mengen in OCL

Personen mit Sicherheitsstufe 0 können für keine Dokument autorisiert werden:

context Mitarbeiter inv: self.sicherheitsstufe = 0 implies

`self.hat -> isEmpty()`

Navigation vom aktuellen Objekt zu einer Menge assoziierter Objekte

Anwendung einer Funktion auf eine Resultatmenge

Weitere Beispiele

Die Anzahl der Dokumente für jeden Mitarbeiter muss mit der Menge der bestehenden Autorisierungen übereinstimmen:

```
context Mitarbeiter inv: self.hat->size() = self.anzahlDok
```

Die für eine Person autorisierten Dokumente sind alle voneinander verschieden:

```
context Mitarbeiter inv: self.hat->forAll(a1, a2: Autorisierung |  
a1<>a2 implies a1.betrifft.dokID <> a2.betrifft.dokID)
```

Es gibt maximal 1000 Dokumente:

```
context Dokument inv: Dokument.allInstances->size() ≤ 1000
```

- Art und Kontext: **context, inv, pre, post**
- Prädikatenlogische Ausdrücke: **and, or, not, implies, exists, forAll**
- Fallunterscheidung: **if...then...else**
- Operationen auf Resultatmengen: **size(), isEmpty(), sum()**
- Funktionen zur Modellreflexion:
z.B. `self.oclIsTypeOf(Mitarbeiter)` liefert im Kontext von Mitarbeiter als Ergebnis WAHR

- Aussagen über alle Instanzen einer Klasse: allInstances
- Navigation: Übliche Punktnotation: self.hat.datum = ...
- Anwendung auf Mengen: Pfeilnotation self.hat->size()
- Zustandsveränderung: @pre-Notation für den Wert im Startzustand anzahlDok = anzahlDok@pre + 1
- Versand von Nachrichten:
chef^Aautorisieren(arbeitsvertrag)

- Werden Anforderungen durch formale Modelle beschrieben, so möchte man wissen, ob das Modell gewisse Eigenschaften aufweist.
- Formale Spezifikationen erlauben,
 - die Gültigkeit von Eigenschaften **zu beweisen**
Verfahren: Mathematisch-logisches Schließen, unterstützt durch **Theorembeweiser**
 - die Gültigkeit von Invarianten **automatisiert zu testen** und ggf. Gegenbeispiele zu finden
Verfahren: Systematisches, automatisiertes Explorieren des gesamten Zustandsraum der Spezifikation und Prüfen der gewünschten Eigenschaft in jedem Zustand (**Model Checking**)
- Eigenschaften, deren Gültigkeit man beweisen oder testen möchte, sind zum Beispiel **sicherheitskritische Invarianten**

Vor- und Nachteil

- Immer eindeutig (da Semantik formal definiert)
- Erfüllung wichtiger Eigenschaften beweisbar / automatisch testbar
- Lösungsneutral
- Formale Verifikation von Programmen möglich
- Modelle simulierbar/ animierbar (z.B. Petrinetze, ASM)

- Erstellung sehr aufwendig
- Prüfung / Nachweis der Vollständigkeit wird nicht einfacher
- Nicht ohne intensive Ausbildung lesbar → Prüfung auf Adäquatheit schwierig
- Große Spezifikationen auch für Fachleute schwer zu verstehen
- Aspekte wie Benutzerschnittstellen sind praktisch nicht modellierbar
- Beschreibung von Ausnahmefällen schwierig
- Zum Teil muss perfekte Technologie angenommen werden



WARUM ES SO SCHWER IST EIN PROGRAMMIERER ZU SEIN

Meine Mutter sagte:

„Bist du so lieb und gehst für mich zum Supermarkt.
Kaufe dort eine Flasche Milch. Wenn sie Eier haben,
bringe sechs mit.“

Nun, ich kam mit sechs Flaschen Milch zurück.

Meine Mutter sagte:

„Warum zum Teufel, bringst du mir sechs Flaschen Milch.“

Meine Antwort:

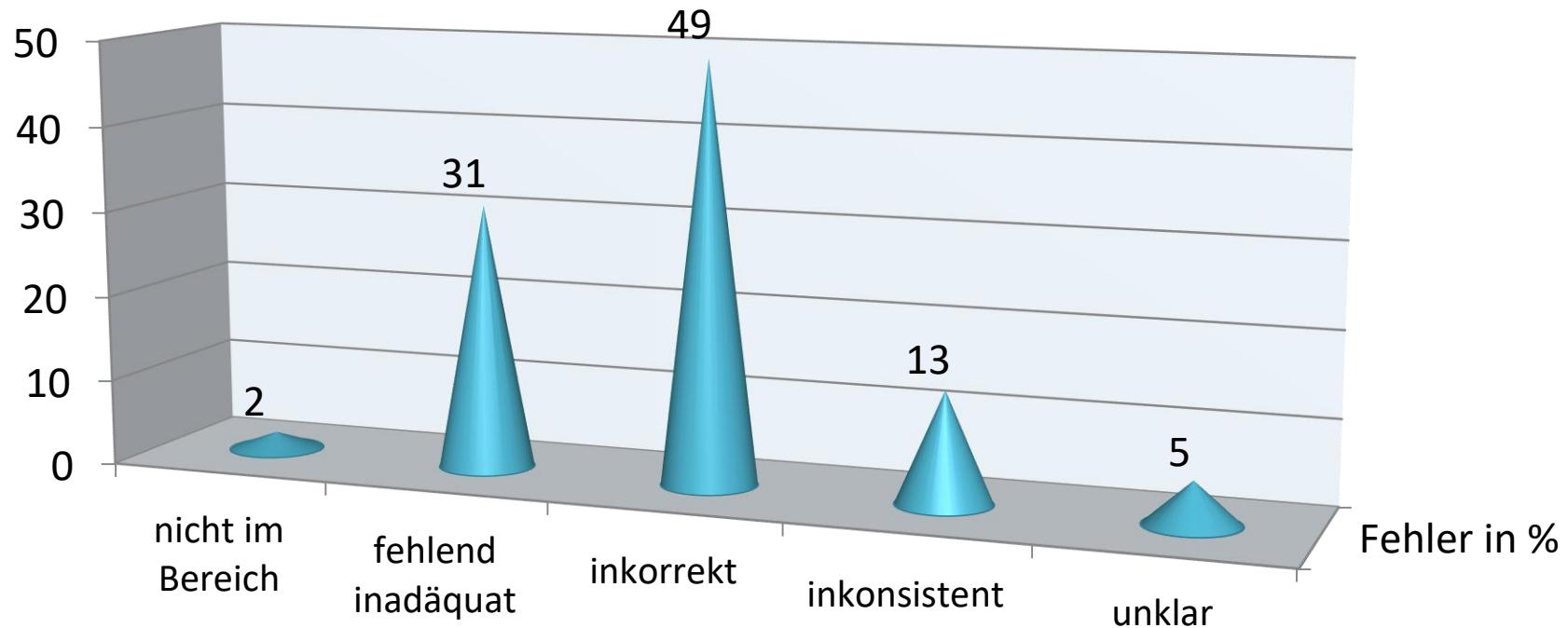
„Weil sie EIER hatten!“

Obersetzung Markus Brandl

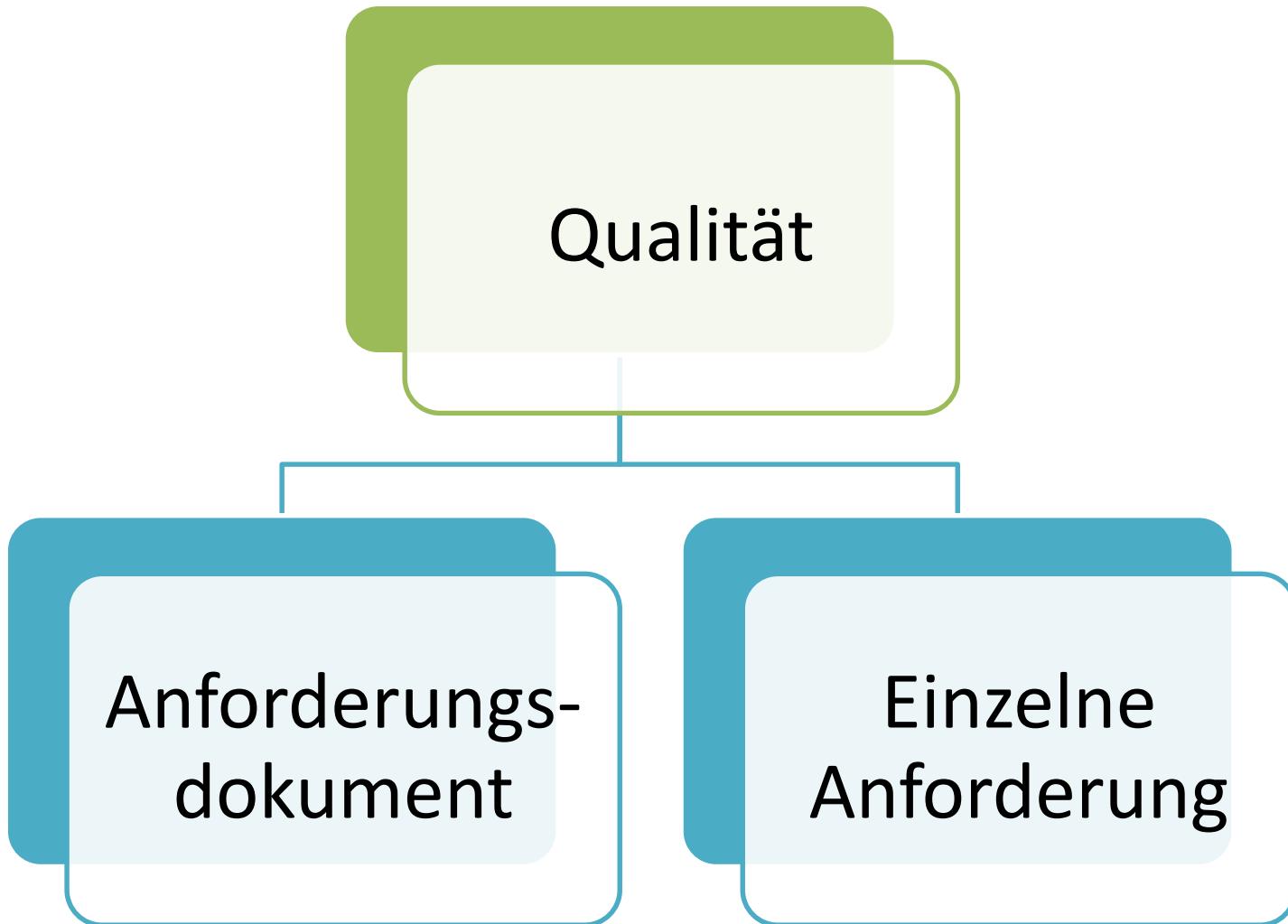
Abschnitt 1.13

QUALITÄT VON ANFORDERUNGEN

Auch Anforderungen können Schwächen haben



B. Boehm: Software Engineering Economics, Prentice-Hall, 1981



Qualitätskriterien Anforderungsdokument

Eindeutigkeit und Konsistenz:

Alle Einzelanforderungen in sich eindeutig und konsistent, Einzelanforderungen widersprechen sich nicht untereinander. Einzelanforderungen und Anforderungsdokumente eindeutig identifizierbar

Verfolgbarkeit:

Beziehungen zwischen Anforderungsdokument und anderen Dokumenten müssen nachverfolgbar sein

IEEE 830

Vollständigkeit:

Alle relevanten Anforderungen sind enthalten, die Einzelanforderungen sind vollständig dokumentiert.

Modifizierbarkeit und Erweiterbarkeit:

Dokumente müssen erweiterbar sein, Struktur und Aufbau sollten leicht änderbar und erweiterbar sein. Dokumente sollten der Versionsverwaltung unterliegen

Klare Struktur:

Angemessener Umfang, klar strukturiert, damit für alle Stakeholder lesbar

- **Bewertet:** Anforderung ist nach Priorität oder Wichtigkeit, Stabilität und rechtlicher Verbindlichkeit bewertet
- **Eindeutig:** Anforderung kann nur auf eine Art und Weise verstanden werden, alle Leser kommen zu einem einzigen, konsequenten Verständnis
- **Korrekt:** Spiegelt die Vorstellung des Stakeholders adäquat wieder
- **Konsistent:** Einzelanforderung für sich enthält keine Widersprüche, die Einzelanforderungen untereinander sind widerspruchsfrei (unabhängig vom Abstraktionsgrad oder der Dokumentationsform)

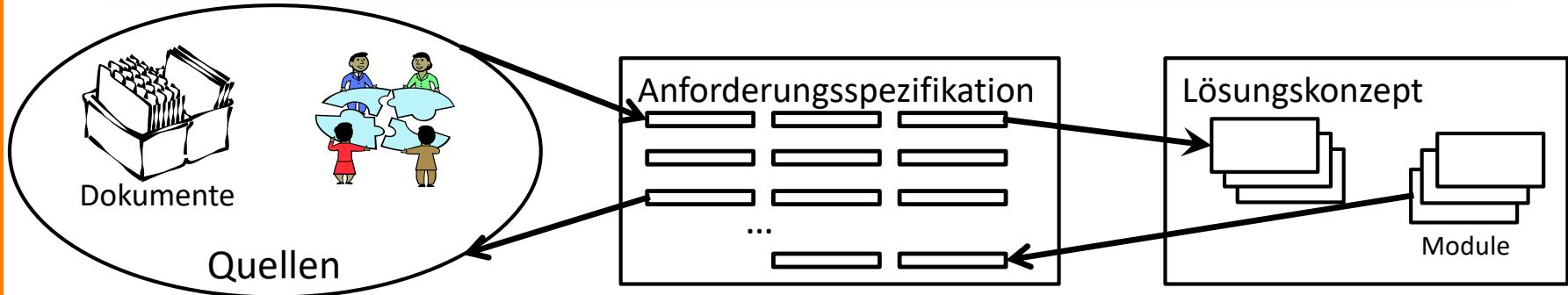
- **Prüfbar:** Die geforderte Produkteigenschaft muss sich durch Test oder Messung nachweisen lassen
- **Verfolgbar:** Sowohl der Ursprung einer Einzelanforderung als auch deren Umsetzung und die Beziehungen zu anderen Dokumenten sind nachvollziehbar
- **Vollständig:** Geforderte und zu liefernde Funktionalität vollständig beschrieben.

Ergänzende Kriterien

- **Abgestimmt**: Anforderung ist für alle Stakeholder korrekt und von allen als gültige Anforderung akzeptiert
- **Gültig und aktuell**: Gegebenheiten im Systemkontext werden so wiedergegeben, dass Anforderung hinsichtlich der aktuellen Gegebenheiten im Systemkontext gültig ist
- **Realisierbar**: Anforderung muss innerhalb der organisatorischen, rechtlichen, technologischen und finanziellen Randbedingungen umsetzbar sein
- **Verständlich**: Anforderung muss für alle Stakeholder verständlich sein

Verfolgbarkeit (traceability)

- Rückwärts: Wo kommt welche Anforderung her?
- Vorwärts: Wo ist welche Anforderung entworfen bzw. implementiert?
- Wie hängen Anforderungen voneinander ab?



- Aufwand und Ertrag für Verfolgbarkeit gegeneinander abwägen
- Rückverfolgungsbeziehungen pflegen, sonst sind sie nutzlos
- Benötigt Werkzeugunterstützung

Abschnitt 1.14

PRÜFUNG UND ABNAHME

Grundsätze

- Spezifikation **validieren**: Die richtigen Anforderungen spezifiziert?
 - Adäquat?
 - Vollständig?
- Spezifikation **verifizieren**: Richtig spezifiziert?
 - Verständlich?
 - Eindeutig?
 - Widerspruchsfrei?
 - Prüfbar?
 - Risikogerecht?
- Fehler möglichst früh finden und beheben

Review (Durchsicht)

- Das **Mittel der Wahl** zur Prüfung von Spezifikationen
- **Walkthrough**: Autor führt durch das Review
- **Inspektion**: Gutachter prüfen eigenständig, tragen in Sitzung Befunde zusammen
- **Autor-Kritiker-Zyklus**: Kunde liest und kritisiert, bespricht Befunde mit Autor

- C. Rupp et al.: *Requirements-Engineering und –Management*; Hanser; 2014
- S. Robertson, J. Robertson: *Mastering the Requirements Process*; Addison-Wesley, 2013
- U. Hammerschall, G. Benekean: *Software Requirements*; Pearson, 2013
- K. Pohl, C. Rupp: *Basiswissen Requirements Engineering (4. Auflage)*; dpunkt.verlag, 2015
- Institute of Electric and Electronic Engineers: *IEEE Standard Glossary of Software Engineering Terminology* (IEEE Std. 610.12-1990)
- Institute of Electric and Electronic Engineers: *IEEE Recommended Practice for Software Requirements Specifications* (IEEE Std. 830-1998)
- J. Woodcock, J. Davies: *Using Z — Specification, Refinement, and Proof*; Prentice Hall, 1996. Online unter: <http://www.usingz.com/>
- M. Glinz: *Folien zur Vorlesung Requirements Engineering I*; Universität Zürich