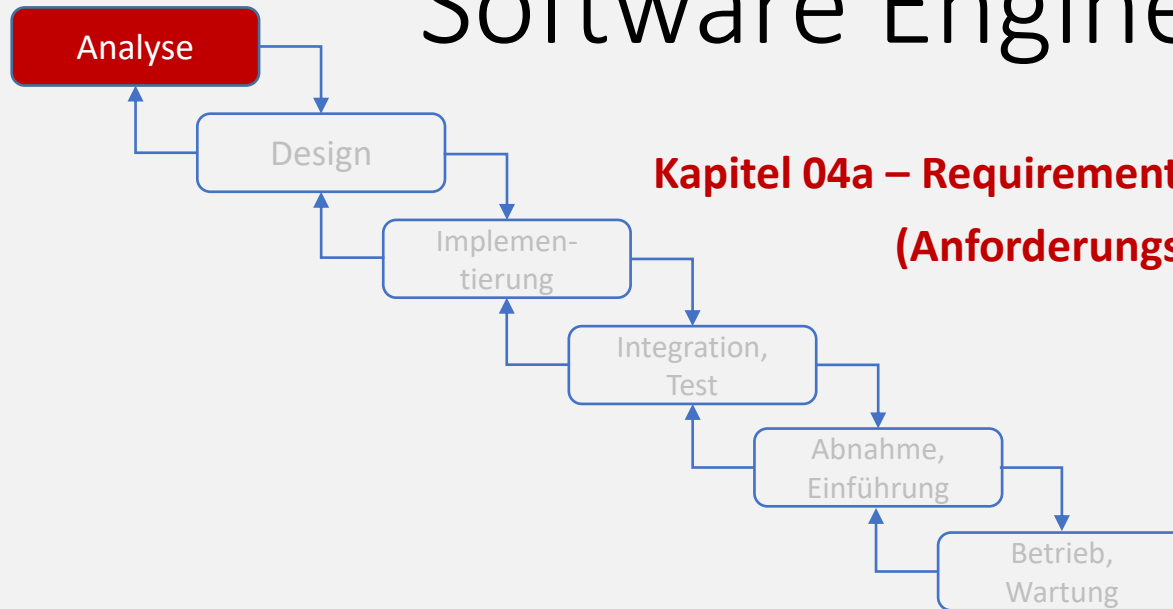




Software Engineering



Kapitel 04a – Requirements Engineering (Anforderungsanalyse)

Gliederung

- Motivation/Überblick
- Bedeutung der Analysephase, Bedeutung des RE
- Hauptaufgaben des RE
- Visionen, Ziele und Rahmenbedingungen
- Anforderungen und Stakeholder
- Anforderungen
 - erfassen,
 - dokumentieren,
 - prüfen,
 - abstimmen,
 - verwalten
- Risiken des RE und Best Practices
- Zeitliche Änderungen von Anforderungen
- Erfassung von Anforderungen mithilfe von Use Cases

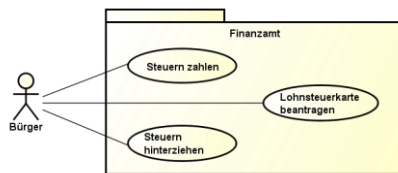
Lernziele

- Was ist Requirements Engineering?
- Warum betreiben wir Requirements Engineering?
- Welche Arten von Anforderungen gibt es?
- Welche Anforderungen müssen erfasst werden?
- Was sind Lasten- und Pflichtenheft?
- Wie erhebt und beschreibt man Anforderungen?

Inhalt

Analyse-Phase ist gegliedert in:

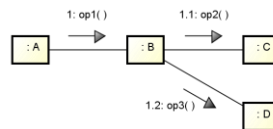
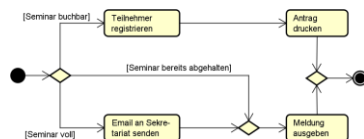
- Requirements Engineering



Attribut	Bedeutung
Identifikation	Kurze, eindeutige Identifikation einer Anforderung
Name	Eindeutiger, charakterisierender Name
Beschreibung	Beschreibt in maximal 3 Sätzen die Anforderung
Verweis	Verweis auf andere Dokumente
Version	Aktueller Versionsstand
Autor	Benennt den Autor
Quelle	Benennt die Quelle
Begründung	Warum ist diese Anforderung wichtig
Abnahmekriterium	Eine messbare Testanweisung welche die Überprüfung der Anforderung erlaubt
Stabilität	Wie stabil in Bezug auf Änderungen ist die Anforderung. (fest, gefestigt, volatil)
Kritikalität	Abschätzung von Schadenshöhe + Eintrittswahrscheinlichkeit bei nicht Erreichen
Priorität	„notwendig, gewünscht, optional“

Thema dieses Kapitels 04a

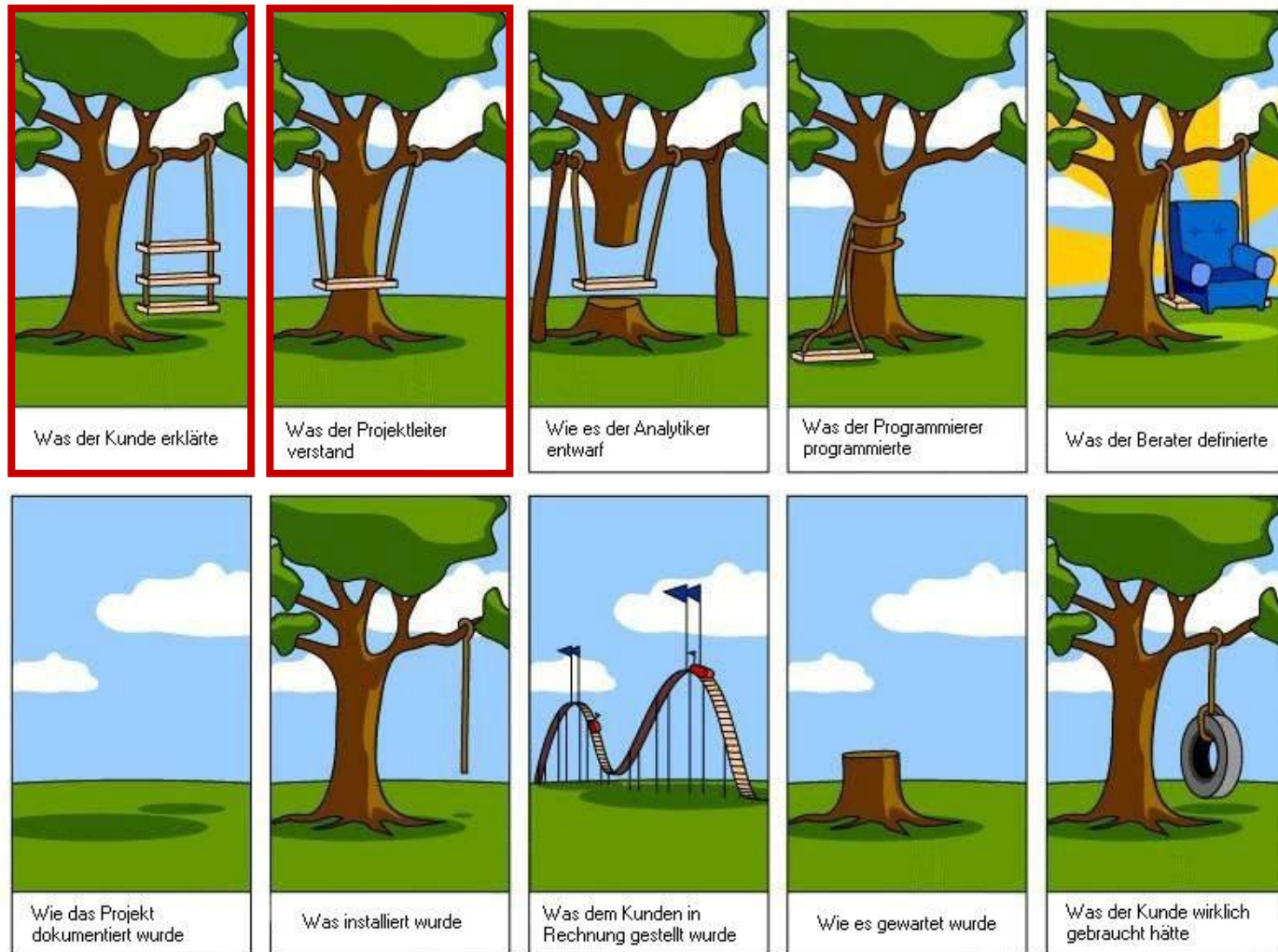
- Anforderungsmodellierung
(Objektorientierte Analyse)



Thema des nächsten Kapitels 04b

Motivation und Überblick

Motivation: Was soll realisiert werden?



Beispiel 1:

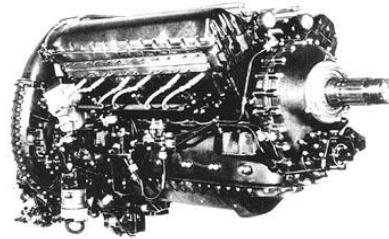
“Ich will den schnellsten Oldtimer...”

- Ergebnis:



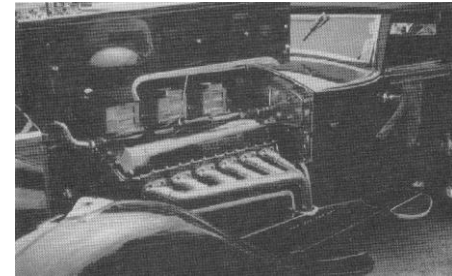
RR Phantom II
Baujahr: 1929-1936

+



12-Zylinder Rolce-Royce
Hubraum: 27.04 l
Leistung: 1565 PS

=



Auto mit
Flugzeugmotor

- Verbrauch 115 Liter/100km...
- Beim unbedachten Gasgeben kann die Kupplung in Flammen aufgehen oder sich die Antriebswelle vom Getriebe abdrehen...
- Das Auto ist praktisch unbenutzbar...
- Lektion: **Erst die Funktionalität** finden, **dann die Lösung!**

Beispiel 2:

... wie es nicht geht

Ein realistischeres weiteres kleines Beispiel: (Ist-Zustand)

- Zur Stundenerfassung und Abrechnung werden von den Projektmitarbeitern spezielle Excel-Tabellen jeden Freitag ausgefüllt und am Montag vom Projektleiter bei der Verwaltung abgegeben.
- Der zuständige Sachbearbeiter überträgt dann die für den Projektüberblick relevanten Daten manuell in ein SAP-System. Dieses System generiert automatisch eine Übersicht, aus der die Geschäftsführung ablesen kann, ob die Projekte wie gewünscht laufen.
- Dieser Bericht liegt meist am Freitag der Folgewoche vor. Die Bearbeitungszeit ist der Geschäftsführung zu lang, deshalb soll der Arbeitsschritt automatisiert werden.

Beispiel 2:

... wie es nicht geht

Ein realistischeres weiteres kleines Beispiel: (Projektplanung)

- Es wird ein Projekt „Projektberichtsautomatisierung“ (ProAuto) beschlossen.
- Der Leiter der hausinternen IT-Abteilung wird über die anstehende Aufgabe informiert. Er erhält eine Beschreibung der Excel-Daten und der gewünschten SAP-Daten.
- Der Leiter stellt fest, dass seine Abteilung das Know-how und die Kapazität hat, das Projekt durchzuführen und legt der Geschäftsführung einen Projektplan mit einer Aufwandsschätzung vor.
- Die Geschäftsführung beschließt, das Projekt intern durchführen zu lassen und kein externes Angebot einzuholen.

Beispiel 2:

... wie es nicht geht



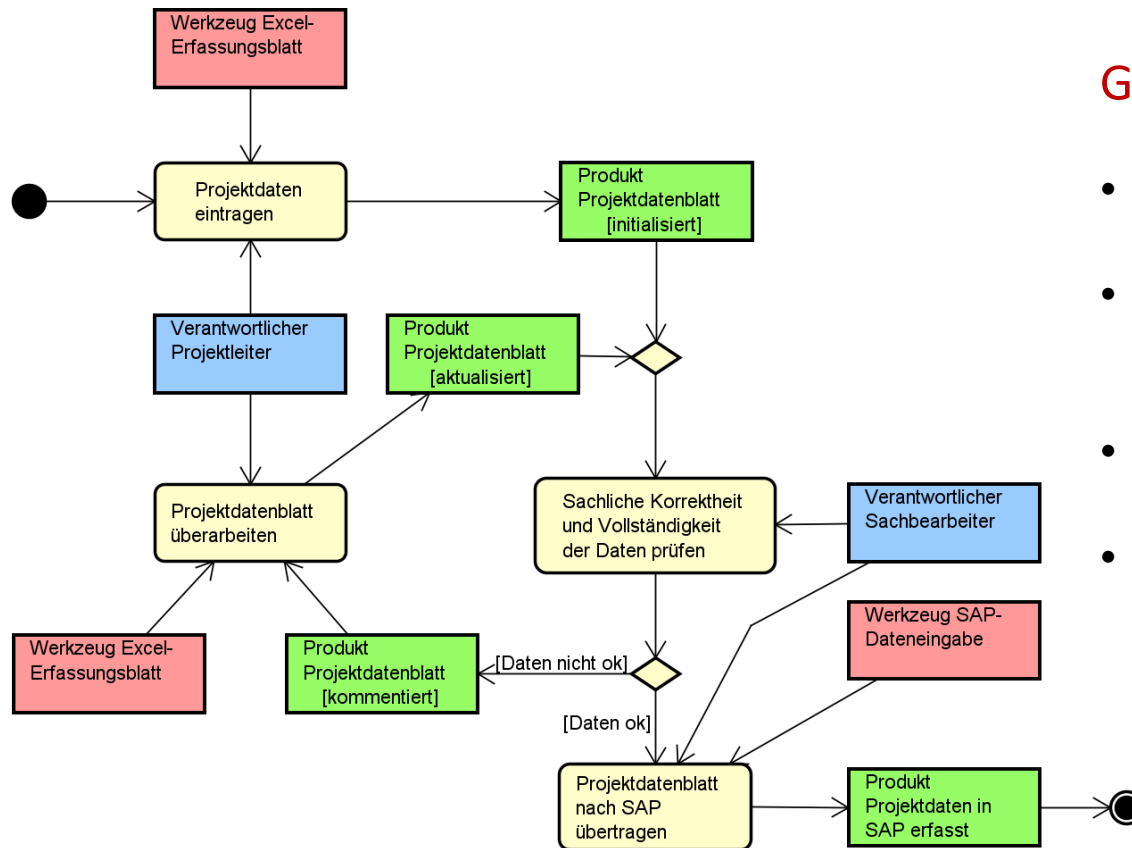
Die Schritte zum Projektmisserfolg:

- Die IT-Abteilung analysiert die Excel-Daten und wie die Daten in das SAP-System eingefügt werden können.
- Kurz nach dem geschätzten Projektende liegt eine technisch saubere Lösung vor. Excel wurde um einen Knopf erweitert, so dass die Projektleiter per Knopfdruck die Daten nach SAP überspielen können.
- Vier Wochen nach Einführung des Systems wird der Leiter der IT-Abteilung entlassen, da die Daten zwar jeden Montag vorliegen, sich aber herausgestellt hat, dass sie nicht nutzbar sind und die erzürnte Geschäftsleitung falsche Entscheidungen getroffen hat. Das Projekt wird an eine Beratungsfirma neu vergeben.

Was könnten Probleme gewesen sein?

Beispiel 2:

... wie es **besser** geht: Geschäftsprozessanalyse



Geschäftsprozessanalyse:

- Ist-Zustand modellieren
- Ist-Zustand analysieren und optimieren
- Soll-Zustand modellieren
- Prozess implementieren und trainieren

Analysephase (RE + OOA)

Ziel:

- Klärung, **was** realisiert werden soll

Aktionen:

- Klärung der **Visionen und Ziele**
- Definition der **Stakeholder** (Systembetroffene)
- Definition des **Systemkontextes** (Einsatzfeld, Benutzer, Systemumgebung)
- Analyse des **Ist-Zustandes**
- Analyse der **Machbarkeit** (Risiko, Kosten)
- Bestimmung der **Anforderungen** (funktional vs nicht-funktional)

Ergebnis:

- Pflichtenheft
- Konzept der Benutzeroberfläche (frühes Feedback)
- Analysemodell (statisch und dynamisch)

Requirements Engineering als erster Schritt der Analysephase

“Requirements Engineering” frei nach Sommerville

Requirements Engineering ist der Prozess des Herausfindens, Analysierens, Dokumentierens und Überprüfens der Dienste und Beschränkungen eines zu erstellenden Systems.

Requirements Engineering ist kooperativer, iterativer, inkrementeller Prozess mit Ziel:

- Alle relevanten Anforderungen sind bekannt und
- In erforderlichem Detaillierungsgrad verstanden
- Involvierte Stakeholder stimmen über Anforderungen überein
- Anforderungen sind konform zu Dokumentationsvorschriften dokumentiert bzw. zu Spezifikationsvorschriften spezifiziert

Aufgaben des Requirements Engineering (Anforderungsanalyse)

- **Bestimmung aller Anforderungen** an die zu erstellende Software bzw. an das zu erstellende DV-System
- Eigenschaften von Anforderungen: sie müssen ...
 - **vollständig**
 - **notwendig** ("WAS statt WIE")
 - **eindeutig**
 - **richtig** ("abgestimmt als Teil einer Zielhierarchie") sein.
- Bemerkung zur Ablauforganisation:
 - Anforderungen müssen nicht notwendig in einer Phase vor Beginn des Entwurfs vollständig bestimmt werden
 - siehe z.B. iterative Vorgehensmodelle! (Kapitel 08)

Bedeutung der Analysephase

Bedeutung der Analysephase

Statistiken belegen:

- 60% der Softwareprojekte scheitern durch Analysefehler
- 50% der Ausfälle im industriellen Sektor sind auf Software-Fehler zurückzuführen

Grundsatz: Frühzeitig Fehlererkennung/-behebung spart Zeit, Kosten, Ärger

Zur Erinnerung:

- 99% Sicherheit (1% Fehlerquote) hätte im Alltag gravierende Auswirkungen (s. Kap 1):
 - jeden Monat eine Stunde verschmutztes Trinkwasser
 - jeden Tag zwei unsichere Flugzeuglandungen am Frankfurter Flughafen
 - jede Stunde 22.000 Schecks die von falschen Bankkonten abgebucht werden
 - jede 1.000ste Bremsung eines Fahrzeuges würde völlig versagen

Bedeutung des Requirement Engineerings

Viele Softwareprojekte scheitern

Hauptgründe für Projektabbruch:

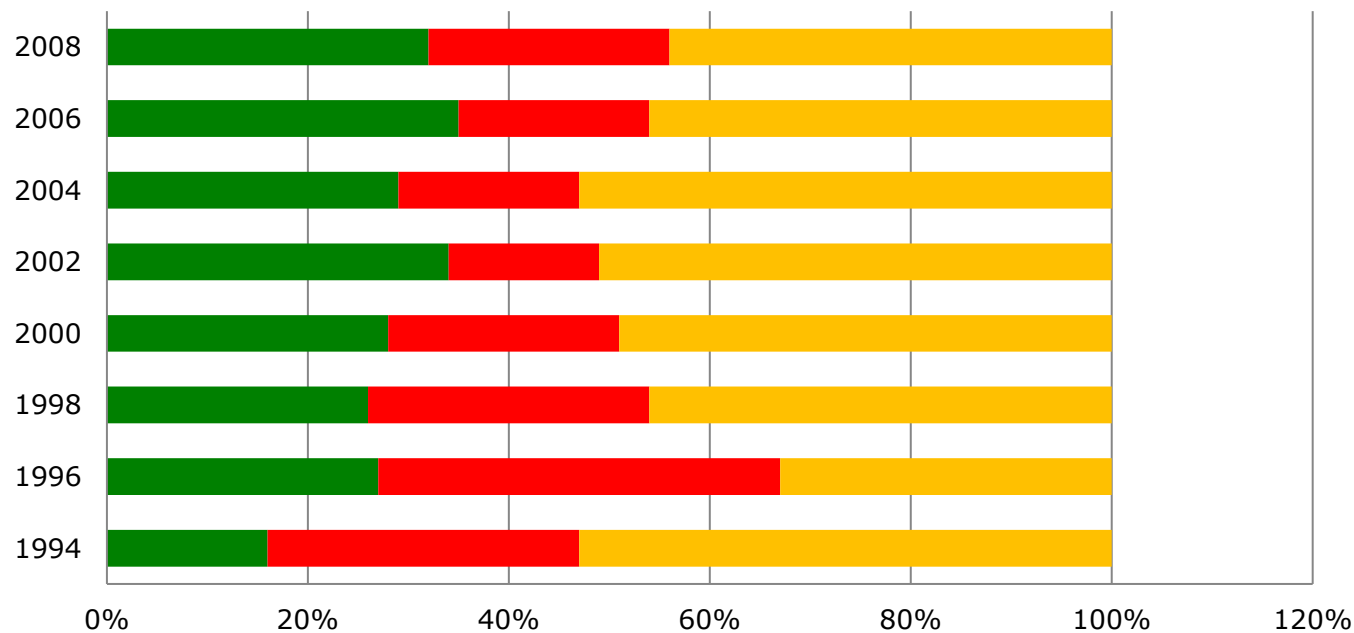
1. Änderung der Anforderungen und des Umfangs
2. Mangelnde Einbindung des höheren Managements
3. Engpass im Budget
4. Fehlende Projektmanagement-Fähigkeiten

→ **Technische Probleme sind meist keine Gründe!**

Schlechtes Image von Softwareprojekten

- Jedes 5. Projekt scheitert

Projekterfolg ist selten



	1994	1996	1998	2000	2002	2004	2006	2008
■ Successful	16%	27%	26%	28%	34%	29%	35%	32%
■ Failed	31%	40%	28%	23%	15%	18%	19%	24%
■ Challenged	53%	33%	46%	49%	51%	53%	46%	44%

[Standish Group - Chaos Report]

Hauptaufgaben des Requirement Engineerings

Hauptaufgaben des Requirement Engineerings

Anforderungen müssen:

- ermittelt:
 - Systemkontext festlegen, Anforderungen ermitteln
- dokumentiert:
 - Adäquat beschreiben
- geprüft und abgestimmt:
 - Prüfen sowie abstimmen und dadurch qualitätssichern
- verwaltet werden:
 - Änderungen sind nachzuverfolgen

Aktivitäten des RE

Hauptaufgaben des Requirement Engineerings

Ergebnis:

- Anforderungsspezifikation (Lasten-, Pflichtenheft):

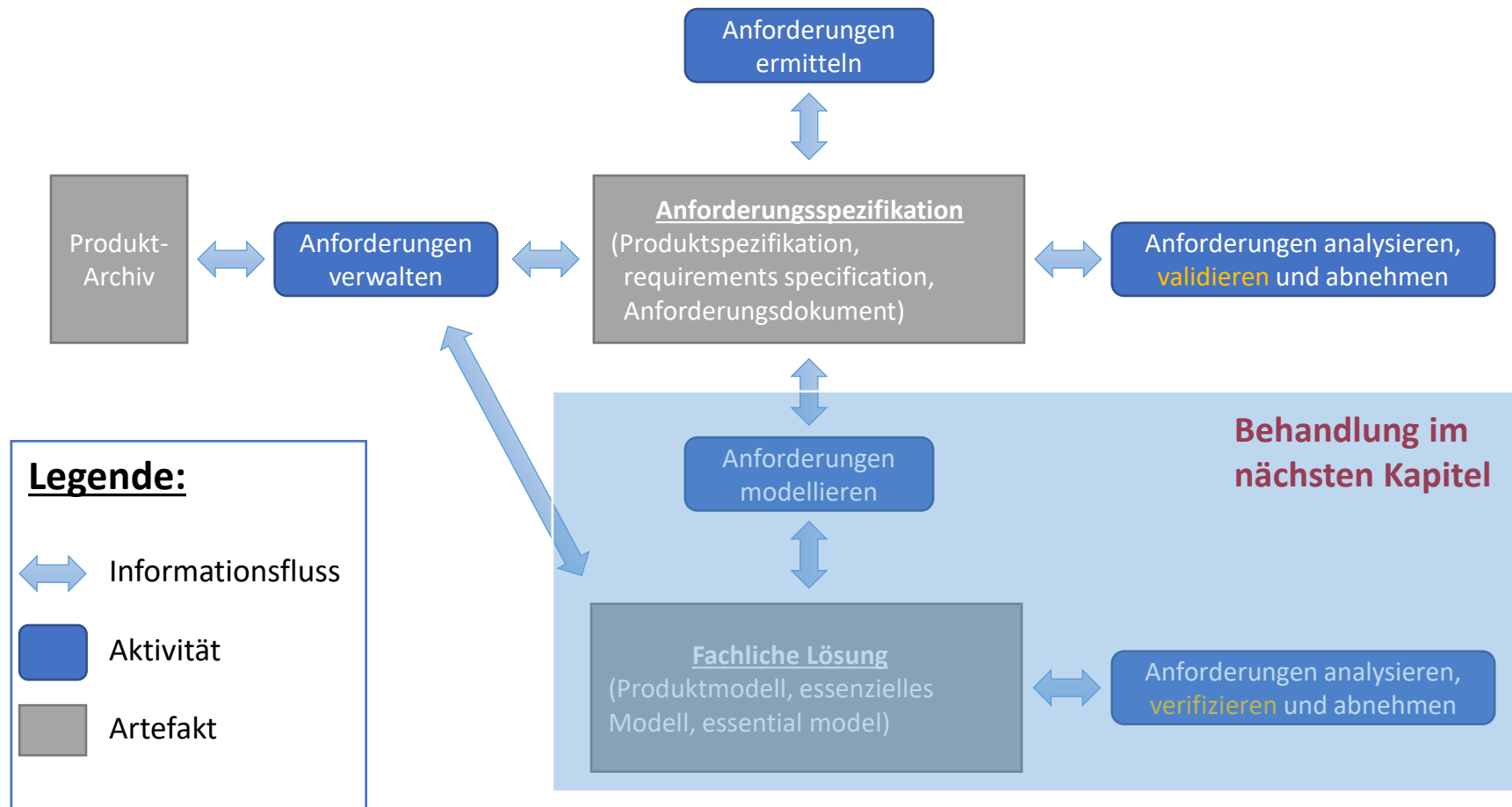
Enthält alle ermittelten, spezifizierten, analysierten und validierten Anforderungen aus Kundensicht.

- Fachliche Lösung bzw. Produktmodell:

Beschreibt die analysierte und verifizierte fachliche Lösung aus Auftragnehmersicht

Artefakte des RE

Requirements Engineering: Aktivitäten und Artefakte



Aufbau eines Requirement Dokumentes nach IEEE 830-1998

1. Einleitung

- a) Zielsetzung (Vision)
- b) Produktziele
- c) Definitionen, Akronyme, Abkürzungen
- d) Referenzen
- e) Überblick

2. Übersichtsbeschreibung

- a) Produkt-Umgebung
- b) Produkt-Funktionen
- c) Benutzer-Eigenschaften
- d) Restriktionen
- e) Annahmen und Abhängigkeiten

3. Spezifische Anforderungen

- a) Externe Schnittstellenanforderungen
- b) Funktionale Anforderungen
- c) Leistungsanforderungen
- d) Entwurfsrestriktionen
- e) Eigenschaften des Softwaresystems
- f) Andere Anforderungen



Visionen, Ziele und Rahmenbedingungen

Visionen und Ziele

Bevor die Anforderungen und Rahmenbedingungen aufgenommen werden, sollten **Visionen und Ziele formuliert** werden.

Vision:

- beschreibt, **was** erreicht werden soll, aber **nicht wie**
- hat **geringe Detailtiefe**; dient als **Leitgedanke**
- wird im Rahmen eines Projektauftrags formuliert, der genehmigt werden muss

Dadurch können Anforderungen später immer mit den Visionen und Zielen abgeglichen werden.

Denkfalle des Auftraggebers:

Visionen und Ziele werden sich im Laufe der Anforderungserstellung schon ergeben.

Beispiel: Visionen

- **Vision für ein Türsteuergerät für einen Fensterheber:**

/V20/ Die Fensterheberkomponente soll das komfortable Heben und Senken der Seitenfenster eines Fahrzeugs ermöglichen.

- **Vision für eine Kundendatenbank:**

/V20/ Die Kundendatenbank soll als single point of truth alle Stammdaten der Kunden enthalten.

Visionen und Ziele

Ziel:

- ausgehend von einer Vision dienen Ziele dazu, die **Vision** zu **verfeinern** und zu **operationalisieren**

Beispiel: Ziel für den Fensterheber

/Z20/ *Die erwarteten Stückzahlen betragen 20.000 Einheiten p.a.*

Regeln zur Zielformulierung:

1. Kurz und prägnant formulieren
2. Aktivformulierung
3. Überprüfbare und realistische Ziele formulieren
4. Nicht überprüfbare Ziele verfeinern
5. Mehrwert eines Ziels hervorheben
6. Begründung für das Ziel liefern
7. Keine Lösungsansätze formulieren

Ziele, Regel 1: “Kurz und prägnant”

Schlecht:

- Das geplante System soll sowohl von Experten als auch von unerfahrenen Personen (Nutzerinnen und Nutzern) benutzbar sein. Unerfahrene User sollen auch ohne große Vorkenntnisse der Bedienerführung oder des Vorgängersystems die vorgesehene Systemfunktionalität nutzen können. Zur Benutzung des Systems sollen die Anwender also keinerlei Schulungen oder spezielle Hilfestellungen benötigen. Der Umgang mit dem System muss daher leicht verständlich sein und ohne große Erfahrung mit dem Vorgängersystem oder vergleichbaren Systemen erfolgen können.

Besser:

- Ein unerfahrener Benutzer soll das System ohne spezielle Schulung verwenden können.

Ziele, Regel 2: “Aktiv”

Schlecht:

- Die Dauer für die Erfassung und Verarbeitung der Messdaten soll halbiert werden. Dadurch soll die Wartezeit bis zum Vorliegen von Ergebnissen verkürzt werden.

Besser:

- Das System erfasst und verarbeitet die Messdaten im Vergleich zum System xy doppelt so schnell. Dadurch muss der Nutzer kürzer auf das Vorliegen von Ergebnissen warten.

Ziele, Regel 3: “Überprüfbar”

Schlecht:

- Das System soll besser sein als das Vorgängersystem.

Besser:

- Das System soll folgende Verbesserungen gegenüber dem System xy bieten:
 1. Die Reaktionszeiten liegen unter 50ms
 2. Das redundante System übernimmt bei Ausfall des Hauptsystems dessen Aufgaben
 3. ...

Ziele, Regel 4: “Nicht Überprüfbares verfeinern”

Schlecht:

- Die Benutzung des Systems soll selbsterklärend sein.

Besser:

- Das System soll selbsterklärend sein, d.h. ein durchschnittlicher Nutzer soll durchschnittlich nach 2 Min. folgende Funktionen aufrufen können: ...

Ziele, Regel 5: “Mehrwert hervorheben”

Schlecht:

- Das System soll leicht benutzbar sein.

Besser:

- Das System soll ..., so dass sich die Nutzer auf andere Aufgaben konzentrieren können.

Ziele, Regel 6: “Begründbar”

Schlecht:

- Das System soll auch von ungeschulten Benutzern intuitiv benutzbar sein.

Besser:

- ... weil es auch in Mietfahrzeugen einsetzbar sein soll.

Ziele, Regel 7: “Keine Lösungsansätze”

Schlecht:

- Durch komprimierte Datenübertragung im Cache soll das geplante System um 10% kürzere Antwortzeiten aufweisen.

Besser:

- Das System soll um 10% kürzere Antwortzeiten aufweisen als System xy.

Eigenschaften von Zielen

Eigenschaften von Zielen:

- vollständig
- korrekt
- konsistent gegenüber anderen Zielen und in sich konsistent
- testbar
- verständlich für alle Stakeholder
- umsetzbar/realisierbar
- notwendig
- eindeutig und positiv formuliert

→ Ziele sind abstrakte Top-Level-Anforderungen

Dokumentieren von Zielen:

Schablone zur Zielbeschreibung

Ziel	Was soll erreicht werden?
Stakeholder	Welche Stakeholder sind in das Ziel involviert? Ein Ziel ohne Stakeholder macht keinen Sinn.
Auswirkungen auf Stakeholder	Welche Veränderungen werden für die Stakeholder erwartet?
Randbedingungen	Welche unveränderlichen Randbedingungen müssen bei der Zielerreichung beachtet werden?
Abhängigkeiten	Ist die Zielverknüpfung mit anderen Zielen unmittelbar verknüpft? Dies kann einen positiven Effekt haben, indem die Erfüllung von Anforderungen zur Erreichung mehrerer Ziele beiträgt. Es ist aber auch möglich, dass ein Kompromiss gefunden werden muss, da Ziele unterschiedliche Schwerpunkte haben
Sonstiges	Was muss organisatorisch beachtet werden?

Dokumentieren von Zielen:

Beispiel: Projektbeschreibung

- Zu entwickeln ist ein individuell auf die Unternehmenswünsche angepasstes **Werkzeug zur Projektverwaltung**.
- Dabei sind die Arbeitspakete (wer macht wann was) und das Projektcontrolling (wie steht das Projekt bzgl. seiner Termine und des Budgets) zu berücksichtigen.
- Projekte werden zur Zeit ausgehend von Projektstrukturplänen geplant und verwaltet.
- Projekte können in Teilprojekte zerlegt werden.
- Die eigentlichen Arbeiten finden in Arbeitspaketen, auch Aufgaben genannt, statt.
- Projekte werden von zusammenzustellenden Projektteams bearbeitet, die zugehörigen Mitarbeiterdaten sind zu verwalten. Zur Ermittlung des Projektstands tragen Mitarbeiter ihre Arbeitszeit und den erreichten Fertigstellungsgrad in das System ein.

[Grundkurs Software-Engineering mit UML, Kleuker]

Dokumentieren von Zielen:

Beispiel: Ziele für eine Projektmanagementsoftware (1)

Ziel	Der neue Kunde soll von der fachlichen Kompetenz unseres Unternehmens überzeugt werden.
Stakeholder	Management, Vertrieb, Entwickler
Auswirkungen auf Stakeholder	Management: Der Projekterfolg hat große Auswirkungen auf die nächsten beiden Jahresbilanzen. Vertrieb: Der Kunde muss über die Kooperationsmöglichkeiten informiert werden Entwickler: Es werden hohe Anforderungen an die Software-Qualität gestellt.
Randbedingungen	Es muss noch geprüft werden, ob langfristig eine für beide Seiten lukrative Zusammenarbeit überhaupt möglich ist.
Abhängigkeiten	Überschneidung mit dem Ziel 2, da eine Konzentration auf die Wünsche des neuen Kunden eventuell einer Verwendbarkeit für den allgemeinen Markt widersprechen kann.
Sonstiges	Das Verhalten des neuen Kunden bei Änderungswünschen ist unbekannt.

[Grundkurs Software-Engineering mit UML, Kleuker]

Dokumentieren von Zielen:

Beispiel: Ziele für eine Projektmanagementsoftware (2)

Ziel	Das neue Produkt soll für einen größeren Markt einsetzbar sein.
Stakeholder	Management, Vertrieb, Entwickler
Auswirkungen auf Stakeholder	Management: Es soll eine Marktposition auf dem Marktsegment Projektmanagement-Software aufgebaut werden. Vertrieb: In Gesprächen mit Kunden wird das neue Produkt und seine Integrationsmöglichkeit mit anderen Produkten ab Projektstart beworben. Entwickler: Die Software muss modular aufgebaut aus Software-Komponenten mit klaren Schnittstellen bestehen.
Randbedingungen	---
Abhängigkeiten	Zu Ziel 1 (Beschreibung dort)
Sonstiges	Eine Analyse der Konkurrenten auf dem Markt liegt vor. Es sind Möglichkeiten für neue, den Markt interessierende Funktionalitäten aufgezeigt worden.

[Grundkurs Software-Engineering mit UML, Kleuker]

Dokumentieren von Zielen:

Beispiel: Ziele für eine Projektmgmtsoftware (3)



Ziel	Die Software muss die Planung und Analyse aller laufenden Projekte ermöglichen
Stakeholder	
Auswirkungen auf Stakeholder	
Randbedingungen	
Abhängigkeiten	
Sonstiges	

[Grundkurs Software-Engineering mit UML, Kleuker]

Rahmenbedingungen (1)

Rahmenbedingung:

- Legt organisatorische und technische Restriktionen für das Softwaresystem bzw. den Entwicklungsprozess fest
- **Organisatorische Rahmenbedingungen:**
 - Anwendungsbereiche (z.B. Textverarbeitung im Büro)
 - Zielgruppen (z.B. Sekretärinnen, Schreibkräfte)
 - Betriebsbedingungen (z.B. Büroumgebung, mobiler Einsatz)
- **Technische Rahmenbedingungen:**
 - Technische Produktumgebung
 - Anforderungen an die Entwicklungsumgebung

Rahmenbedingungen (2)

Auswirkungen von Rahmenbedingungen auf das Softwareprojekt:

Rahmenbedingungen können:

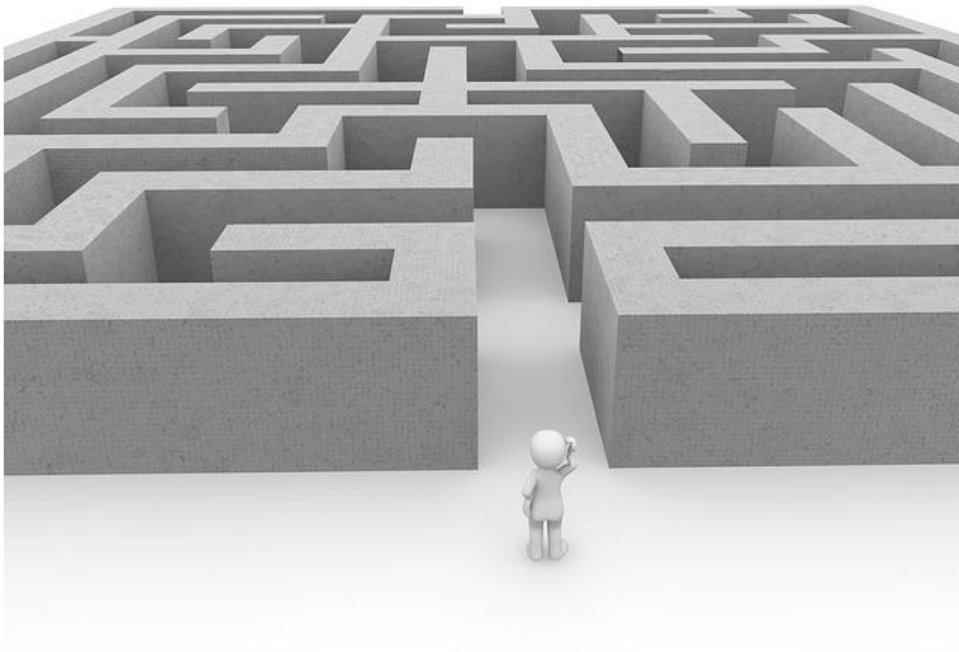
- keine der Anforderungen einschränken
- die mögliche Realisierung von Anforderungen einschränken
- zu Änderung von Anforderungen führen
- zu neuen Anforderungen führen
- zu nicht realisierbaren Anforderungen führen

Denkfalle des Auftraggebers:

Rahmenbedingungen legt der Auftragnehmer fest.

Visionen, Ziele, Rahmenbedingungen

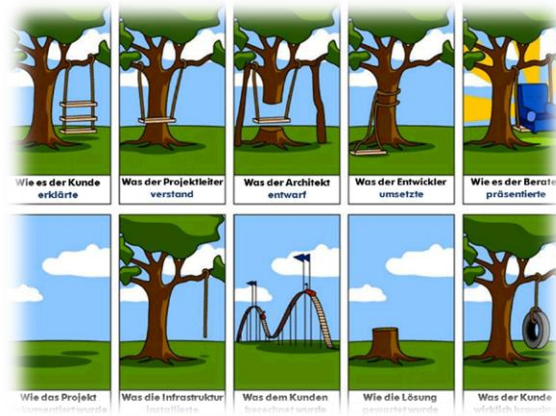
Beispiel



Vision?

Ziel?

Rahmenbedingungen?



Anforderungen

Definition

Definition: Anforderung

Balzert:

“Anforderungen legen fest, was man von einem Softwaresystem als Eigenschaften erwartet”

Häufig verwendete Unterkategorisierung:

– Funktionale Anforderungen:

- Beschreiben, was ein System tun soll
- Dienste, die das System zur Verfügung stellt
- Reaktion auf bestimmte Eingaben
- Verhalten in bestimmten Situationen

– Nicht-funktionale Anforderungen:

- Beschränkungen
- Beziehen sich eher auf das Gesamtsystem als auf einzelne Dienste

Merke: Nicht die Unterscheidung in die beiden Arten ist essentiell sondern die Berücksichtigung beider bei der Softwareentwicklung

Weitere Kategorisierung von Anforderungen

- **Funktionale Anforderungen:**

- Funktionen, Daten, Stimuli, Reaktionen, Verhalten,...
- Bsp.: Das System muss Kunden-, Firmen-, Seminar-, Veranstaltungs- und Dozentendaten permanent speichern.

- **Leistungsanforderungen:**

- Zeit, Geschwindigkeit, Umfang, Durchsatz,...
- Bsp.: Alle Reaktionszeiten auf Benutzeraktionen müssen unter 5 Sek. liegen

- **Qualitätsanforderungen:**

- Zuverlässigkeit, Benutzbarkeit, Sicherheit, Portabilität, Wartbarkeit,...

- **Randbedingungen/Einschränkungen:**

- physikalisch, rechtlich, kulturell, Umgebung, Schnittstellen,...

Beispiele: Funktionale Anforderungen (1)



Funktion: "Vorlesung in Vorlesungsverzeichnis eintragen"

– Eingaben:

- Raum, Zeit und Titel einer Vorlesung.

– Verarbeitungsschritte:

- Prüfe, ob der Vorlesungstitel schon vergeben ist
- Prüfe, ob der Raum zur angegebenen Zeit schon vergeben ist
- Wenn nicht, wird die neue Vorlesung eingetragen und die Daten der Vorlesung werden angezeigt.
- Falls vergeben, wird die Vorlesung nicht eingetragen und eine entsprechende Fehlermeldung wird angezeigt.

– Ausgaben:

- Die Vorlesung wird angezeigt oder ein Fehler wird gemeldet.

Beispiele: Funktionale Anforderungen (2)

Weitere Beispiele:

- Aktionen, die vom System ausgeführt werden sollen:
 - Bsp.: Das System muss Vorlesungen in die DB aufnehmen können.
- Systeminteraktionen, die dem Nutzer ermöglicht werden:
 - Bsp.: Das System muss es dem Administrator beim Eintragen einer Vorlesung ermöglichen, Raum, Zeit und Titel einzugeben.
- Allgemeine funkt. Vereinbarungen und Einschränkungen:
 - Bsp.: Der Client ist für den Kommunikationsaufbau zuständig.

Nicht-funktionale Anforderungen (unvollst.)



Beispiele: Nicht-funktionale Anforderungen

- **Technische Anforderungen:**

- Das System muss mit Java entwickelt werden und in der Version “Oracle Java 12” laufen

- **Ergonomische Anforderungen:**

- Die Benutzerführung erfolgt in deutsch

- **Anforderungen an die Dienstqualität:**

- Das System muss jede Anfrage des Benutzers innerhalb von 2 Sekunden beantworten
- Der Speicherbedarf darf 2 GB nicht übersteigen
- Das System muss 8 Anfragen pro Sekunde beantworten können

- **Anforderungen an die Zuverlässigkeit:**

- Der Dienst muss eine Verfügbarkeit von 999/1000 haben

- **Anforderungen an den Entwicklungsprozess:**

- Entwickler muss mit Kunden monatliche Reviews der zu erstellenden Dokumente machen

- **Rechtlich-vertragliche Anforderungen:**

- Der Kunde leistet für jeden Meilenstein ein Viertel der vertraglich für die Systementwicklung vereinbarten Summe
- Die deutschen Datenschutzrichtlinien müssen erfüllt sein

Probleme bei der Requirements Acquisition (1) (Anforderungsermittlung)

Warum?

- Herausfordernde Aktivität
 - Erfordert Zusammenarbeit von Menschen mit:
 - verschiedenen Erfahrungen/Hintergründen
 - z.B.: Anwender mit Wissen der Anwendungsdomäne
 - z.B.: Entwickler mit Wissen über Lösungsdomäne (Wissen über Design & Implementierung)
- **Problem: Verständnisschwierigkeiten**
- Brückenschlag zwischen Anwender und Entwickler:
 - Szenarios (s. z.B. Sequenzdiagramme):
Beispielanwendung des Systems, ausgedrückt durch eine Serie von Interaktionen zwischen Anwender und System
 - Anwendungsfälle (Use Cases):
Abstraktion, die eine Klasse von Szenarios beschreibt

Probleme bei der Requirements Acquisition (2) (Anforderungsermittlung)

Anforderungen sind generell schwierig zu ermitteln.

Einige weitere Gründe sind:

- Interessensbeteiligte können ihre Anforderungen oft **nicht ausdrücken**
 - Schwierigkeit, Anforderungen mit Worten zu beschreiben
 - Unrealistische Erwartungen
 - Fehlendes Kostenbewusstsein
- **Fachsprache** der Interessensbeteiligten
- **Implizites Wissen** / unbewusstes Wissen der Interessensbeteiligten
- **Unterschiedliche Interessen** von unterschiedlichen Interessensbeteiligten
- **Einfluss politischer Faktoren:**
 - Manager äußern spezifische Anforderungen, um ihren Einfluss zu vergrößern.
 - Rascher Wandel und geringe Stabilität von Anforderungen.

Anforderungen an Anforderungen

Welche Eigenschaften sollen Anforderungen erfüllen?

Sie sollen:

- korrekt
- eindeutig
- vollständig
- konsistent
- klassifizierbar nach Wichtigkeit
- klassifizierbar nach Stabilität
- überprüfbar
- verfolgbar

sein.

Abnahmekriterien

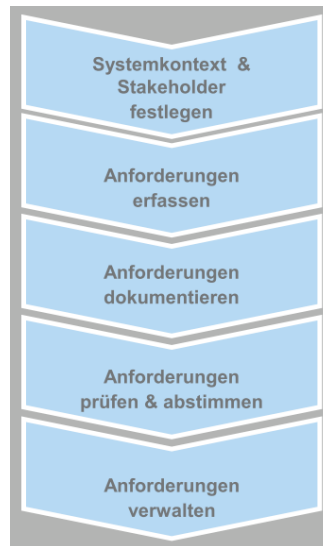
Legen Sie bereits mit den Anforderungen Abnahmekriterien fest, so haben Sie folgende Vorteile:

- Es wird sich bei der **Abnahme** nicht nur auf das realisierte System gestützt.
- Bereits bei Formulierung der Anforderungen wird darauf geachtet, **dass** sie auch **überprüft werden können**, was nach aller Erfahrung zu einer operationalisierten Festlegung und qualitativen Verbesserung der Anforderungen führt.
- Es ist schon zu Beginn klar, **wie überprüft werden kann**, ob Anforderungen korrekt realisiert wurden.
- Die Formulierung von Abnahmekriterien führt zu **bessere**r Veranschaulichung und **Verständlichkeit** der meist abstrakt formulierten Anforderungen.

Beispiel: Webshop (Ziele+Anforderungen)



1. Wie lautet das Grobziel eines Kunden bei der Nutzung eines Webshops?
2. Ermitteln Sie (Teil-)ziele von Kunden eines Webshops
3. Formulieren Sie Anforderungen an das System:



Aktivitäten des Requirement Engineerings

- Systemkontext festlegen
- Stakeholder bestimmen und analysieren
- Ziele bestimmen
- ...

Probleme beim Requirements Engineering

Probleme mit Anforderungen an große Systeme:

- Auftraggeber, Nutzer, Betreiber etc. sind häufig verschiedene Personen, unterschiedliche Personen haben teilweise widersprüchliche Anforderungen
- Die Effekte des angestrebten Systems sind schwer vorhersehbar
- Anforderungen ändern sich im Laufe der Entwicklungszeit
- Großer Umfang der Anforderungen
- Komplexe Interaktion mit anderen Systemen

Initiale Aufgaben im RE:

- 1. Systemkontext festlegen und Stakeholder ermitteln**
- 2. Ermittlung der Ziele** des Systems

Systemkontext festlegen

Früh im Projekt wird der Systemkontext erfasst und dokumentiert

- Textuell und in Form eines Kontextdiagramms
- Form des Kontextdiagramms ist nicht festgelegt (z.B. Paket-, Klassendiagramm oder Bild mit eigenen Notationselementen)

- **Definition “Systemkontext”:**

- Teil der Umgebung eines Systems, der für Definition und Verständnis des betrachteten Systems relevant ist

- **Ziel:**

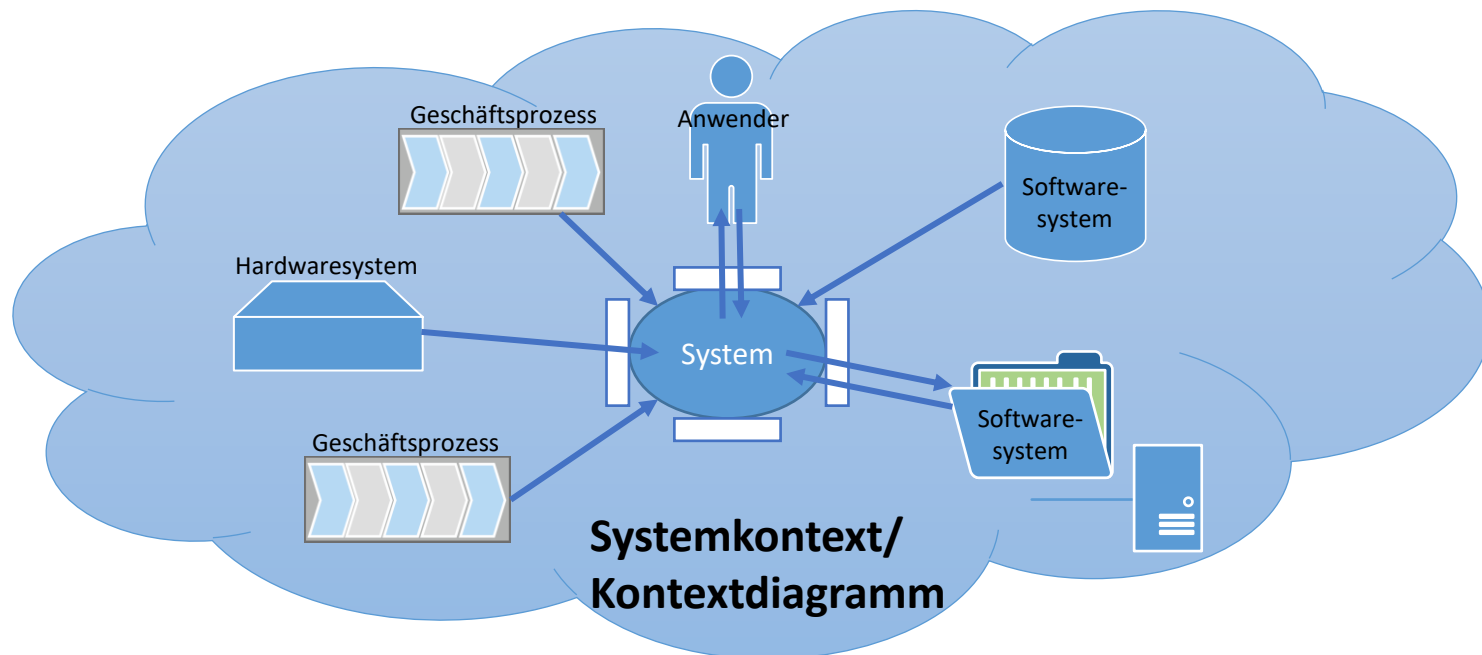
- Abgrenzen des Systems von der Umgebung
 - Identifikation von für Anforderungen relevanter Umgebungsteile

- **Motivation:**

- Der Ursprung aller Anforderungen liegt in deren Umgebung

Systemkontext festlegen (Kontextdiagramm)

- **Beispiel-Typen und -Aspekte im Systemkontext (s. Übung → konkretisieren):**
 - Personen (Stakeholder oder Stakeholdergruppen)
 - System im Betrieb (andere technische Umsysteme)
 - Prozesse (technisch oder physikalisch)
 - Dokumente (z.B. Gesetze, Standards, Systemdokumentation)



Beispiel: Webshop (Kontextdiagramm)



Erstellen Sie ein Kontextdiagramm für einen Webshop:

Stakeholder

Definition “Stakeholder”:

- Jemand der Einfluss auf die Anforderungen hat, da er vom System betroffen ist.
“Systembetroffener”

Stakeholder:

- Sind Quellen für Anforderungen
- Übersehen eines Stakeholders führt zu lückenhaften Anforderungen

Beispiele:

- Kunden
- Mitarbeiter
- Zulieferer
- Eigentümer
- ...

Checkliste zum Finden von Stakeholdern (1)

- **Endanwender**

- Größte und wichtigste Gruppe, liefert Großteil fachlicher Ziele
- Durchdachtes Auswahlverfahren für Anwenderrepräsentanten nötig (Vertrauensbasis der gesamten Anwendergruppe berücksichtigen!)

- **Management des Auftragnehmers (wir)**

- Gewährleisten Konformität mit Unternehmenszielen und Strategien, sowie der Unternehmensphilosophie
- Sind die Sponsoren!

- **Käufer des Systems**

- Wer ist für die Kaufentscheidung verantwortlich?
- Liefer-Vertrags-Zahlungskonditionen?

- **Prüfer, Auditoren**

- Sind für Prüfung, Freigabe und Abnahme notwendig

- **Entwickler**

- Nennen die technologiespezifischen Ziele

Checkliste zum Finden von Stakeholdern (2)

- **Wartungs- und Servicepersonal**

- Wartung und Service muss unkompliziert und zügig durchzuführen sein
- Wichtig bei hohen Stückzahlen

- **Produktbeseitiger**

- Wichtig, wenn ausgeliefertes Produkt nicht nur Software umfasst, Frage der Beseitigung (z.B. Umweltschutz), kann enormen Einfluss auf Zielsetzung von Produktentwicklung haben

- **Schulungs- und Trainingspersonal**

- Liefern konkrete Anforderungen zu Bedienbarkeit, Vermittelbarkeit, Hilfesystem, Dokumentation, Erlernbarkeit

- **Marketing und Vertriebsabteilung**

- Als interne Repräsentanten externer Kundenwünsche und Marktentwicklung

Checkliste zum Finden von Stakeholdern (3)

- **Systemschützer**
 - Stellen Anforderungen zum Schutz vor Fehlverhalten von Stakeholdern
- **Standards und Gesetze**
 - Vorhandene und zukünftige Standards/Gesetze berücksichtigen
- **Projekt- und Produktgegner**
 - Vor allem zu Beginn des Projekts möglichst mit einbeziehen, sonst drohen Konflikte
- **Kulturkreis**
 - Setzt Rahmenbedingungen, z.B. verwendete Symbolik, Begriffe, ...
- **Meinungsführer und die öffentliche Meinung**
 - Beeinflussen oder schreiben Ziele vor, Zielmärkte berücksichtigen

Beispiel: Webshop (Stakeholder)



Finden Sie konkrete Stakeholder für unser Webshop-Produkt:

Stakeholder dokumentieren: Stakeholder-Liste

Stakeholderliste (s. Übung):

- Listet alle für Projekt relevanten Stakeholder auf
- Beschreibt wichtigste Informationen über jeden Stakeholder:

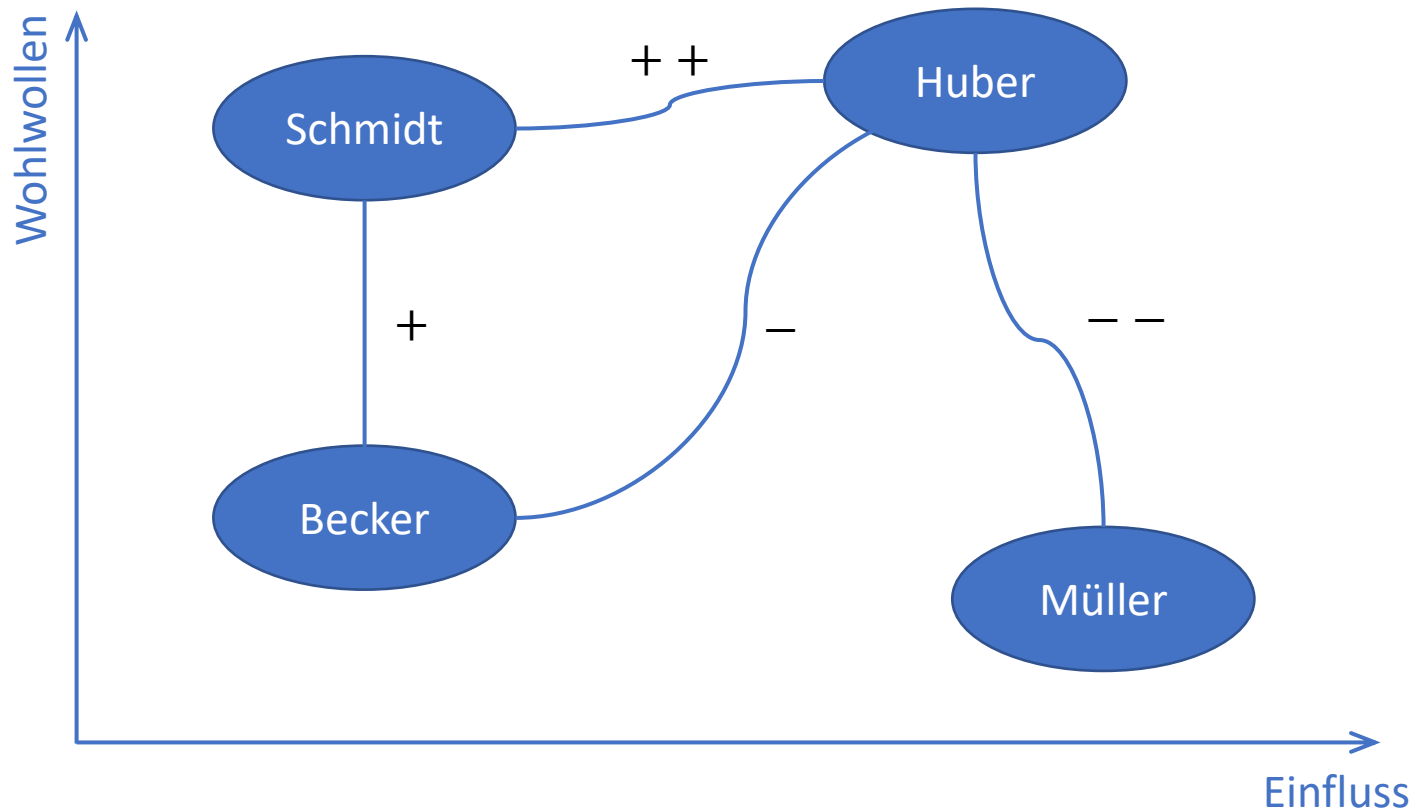
Name/Rolle	Beschreibung	Konkreter Vertreter	Wissensgebiete	Verfügbarkeit	Begründung
Kneipen-Management	Nennt Produkt- und Projektziele	Herr Müller Tel: 4711 Mail: M@oth.de	Kennt Vorgängerprodukte im Detail, da vorher selbst Anwender	5% verfügbar	Entscheidung über Realisierung, Geldgeber
Anwender: Koch	Ist Benutzer des Systems (nicht des Mobilteils)	Herr Becker Tel: 0815 Mail: B@oth.de	Experte der Speisenverbuchung	Urlaub: 1.1.-14.1. 20% verfügbar	Systemanwender muss künftig Speisen verbuchen
...

→ Übersehen von Stakeholdern kann zu unvollständigen Anforderungen führen

Stakeholder dokumentieren: Stakeholder-Portfolio (Beispiel)



Stakeholder-Portfolio:



Anforderungen erfassen

Anforderungen erfassen

Quellen für Anforderungen:

- Stakeholder:
 - involvierte Benutzergruppen
- Dokumentationen:
 - Gesetze, Normen
 - Branchen-/Organisations-spezifische Dokumentation
- Systeme im Betrieb:
 - Legacy- bzw. Vorgängersystem

Anforderungen erfassen

Ermittlungstechniken:

– Befragungstechniken:

- Stakeholder wird direkt zu seinen Anforderungen befragt:
 - Setzt Willen zur Mitarbeit und
 - die Möglichkeit des „sich Ausdrückens“ voraus
- Interview zwischen Requirements-Engineer und Stakeholder
- Fragebogen
- Osborn-Checkliste (siehe z.B. <http://kreativitätstechniken.info/osborn-checkliste/>)

– Dokumentengetrieben:

- Systemarchäologie (Analyse der Dokumentation bestehender Systeme)
- Perspektivenbasiertes Lesen
- Wiederverwendung bereits erstellter Anforderung

Anforderungen erfassen

Ermittlungstechniken:

– Beobachtungstechniken:

- **Feldbeobachtung:** Beobachtung des Stakeholders in dessen gewohnter Umgebung. Dokumentation der Abläufe, Prozesse, Handgriffe, usw.
- **Apprenticing** (= in die Lehre gehen): Requirements-Engineer erlernt die Tätigkeit des Stakeholders. Unklare Schritte werden hinterfragt. Gut geeignet für die Ermittlung „selbstverständlicher Abläufe“

– Kreativitätstechniken:

- **Brainstorming**
- **Perspektivenwechsel** (Sechs-Hüte-Denken)



Anforderungen dokumentieren

Anforderungen dokumentieren

Definition: “Das Anforderungsdokument”:

- Eine Anforderungsspezifikation ist ein Dokument, das spezifizierte Anforderungen enthält, d.h. Anforderungen, die definierten Spezifikationskriterien genügen.

Zentrale Bedeutung von Anforderung und deren Dokumentation:

- Ausgangspunkt für **nachfolgende Phasen** wie Systementwicklung, Test, Abnahme
- Anforderungen sind **rechtlich relevant**
- Anforderungen sind **komplex** (z.B. Umfang und Vernetzung)
- Anforderungen sollen **allen Beteiligten** zur Verfügung stehen

Anforderungen dokumentieren

- **Benutzeranforderungen (Was? und Wofür?):**

- Aussagen in natürlicher Sprache (Vorteile/Nachteile?)
- Einfache Diagramme (Vorteile/Nachteile?)
- Beschreiben Dienste, die System leisten soll
- Beschreiben Randbedingungen unter denen System betrieben wird
- **Systembeschreibung aus Kundensicht (→ Lastenheft)**

- **Systemanforderungen (Wie? und Womit?):**

- Detaillierte Festlegung von Funktionen, Diensten und Beschränkungen
- Beschreibung, was implementiert werden soll
- **Systembeschreibung aus technischer Sicht (→ Pflichtenheft)**

Benutzer- und Systemanforderungen



- Üblicherweise verfeinert man Benutzeranforderungen (Kunde/Anwender) in Systemanforderungen
- **Beispiel Benutzeranforderungen (BA):**
 - *BA: Das Kontoverwaltungssystem erlaubt es einem Bankmitarbeiter, die Kontobewegungen der letzten vier Wochen zu einem gegebenen Kundenkonto einzusehen*
- **Mögliche zugehörige Systemanforderungen (SA):**

[Quelle: Software Requirements, U. Hammerschall, Pearson 2013]

Anforderungen dokumentieren

Lastenheft vs Pflichtenheft

Anforderungsdokumentation üblicherweise in zwei Dokumenten

– Lastenheft:

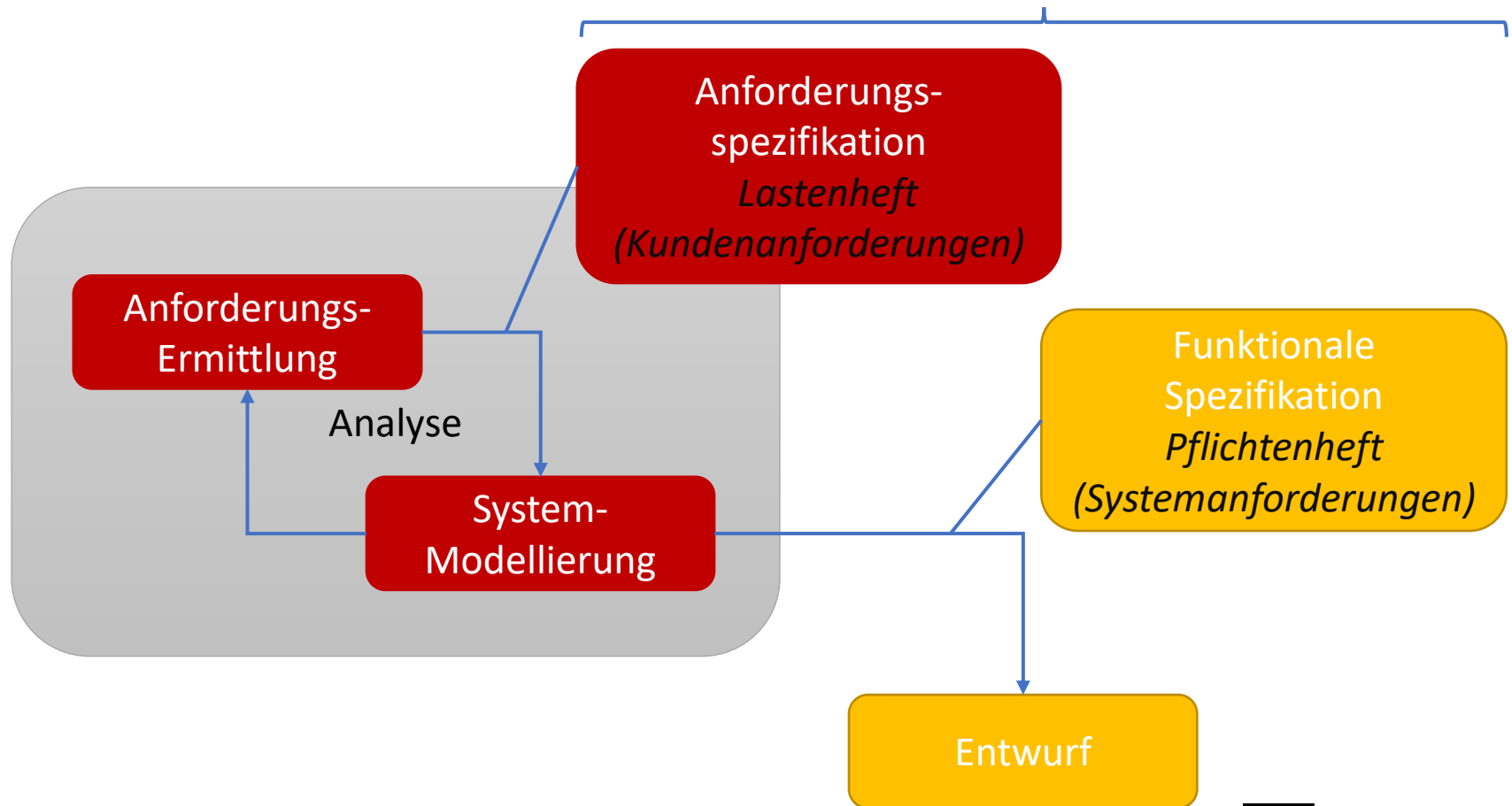
- Wird von Auftraggeber erstellt
- Beschreibung des **“Was”** und **“Wofür”**
- “Grobes” Pflichtenheft
- Details werden bewusst offen gelassen
- Beispiel: s. “Beispiel: Lastenheft” unter “Zusätzliches Unterrichtsmaterial” im ELO-Kurs

– Pflichtenheft:

- Wird vom Auftragnehmer erstellt
- Beschreibung des **“Wie”** und **“Womit”**
- Zu lieferndes System wird detailliert
- Systembeschreibung aus technischer Sicht
- Beispiel: s. “Beispiel: Pflichtenheft” unter “Zusätzliches Unterrichtsmaterial” im ELO-Kurs

Zusammenspiel von Benutzer- und Systemanforderungen

Achtung: wir definieren als Anforderungsspezifikation die Gesamtheit aus Lasten- und Pflichtenheft



Template: Lastenheft

Nach Balzert sollte das **Lastenheft** folgendermaßen aufgebaut sein:

1. Vision und Ziele
2. Rahmenbedingungen
3. Kontext und Überblick
4. Funktionale Anforderungen
5. Qualitätsanforderungen

Template: Pflichtenheft

Nach Balzert sollte das **Pflichtenheft** folgendermaßen aufgebaut sein:

1. Vision und Ziele
2. Rahmenbedingungen
3. Kontext und Überblick
4. Funktionale Anforderungen
5. Qualitätsanforderungen
6. **Abnahmekriterien**
7. **Subsystemstruktur (optional)**
8. **Glossar**

→ das Pflichtenheft ist eine Verfeinerung des Lastenheftes

Anforderungen an die Anforderungsspezifikation

Die Anforderungsspezifikation sollte folgende Eigenschaften besitzen:

- korrekt
- vollständig
- konsistent
- Abhängigkeiten angebar
- modifizierbar
- erweiterbar
- im Umfang angemessen
- nach verschiedenen Sichten auswertbar

(Ähnliche Eigenschaften waren schon bei Zielen und den „Anforderungen an Anforderungen“ gefordert worden)

Requirements Specification Template (Anforderungsspezifikationsschablone):

→ [Volere Template](#) (s. Übung und ELO)

Anforderungen dokumentieren (1)

Für eine Anforderung sollten folgende Attribute erfasst werden:

Attribut	Bedeutung
Identifikation	Kurze, eindeutige Identifikation einer Anforderung
Name	Eindeutiger, charakterisierender Name
Beschreibung	Beschreibt in maximal 3 Sätzen die Anforderung
Verweis	Verweis auf andere Dokumente
Version	Aktueller Versionsstand
Autor	Benennt den Autor
Quelle	Benennt die Quelle
Begründung	Warum ist diese Anforderung wichtig
Abnahmekriterium	Eine messbare Testanweisung welche die Überprüfung der Anforderung erlaubt
Stabilität	Wie stabil in Bezug auf Änderungen ist die Anforderung. (fest, gefestigt, volatile)
Kritikalität	Abschätzung von Schadenshöhe + Eintrittswahrscheinlichkeit bei Nichterreichen
Priorität	„notwendig, gewünscht, optional“

Anforderungen dokumentieren (2)

Anforderungen in natürlicher Sprache dokumentieren: (Prosatext)

Vorteil:

- Für Muttersprachler leicht zu verstehen

Nachteil:

- Natürliche Sprache kann mehrdeutig sein
- Redundanzen und Widersprüche schwer zu erkennen

Beispiele?

- „*Alle Bereiche haben einen Beenden-Knopf*“
- „*Patientenzustand wird automatisch begutachtet und aufgenommen*“

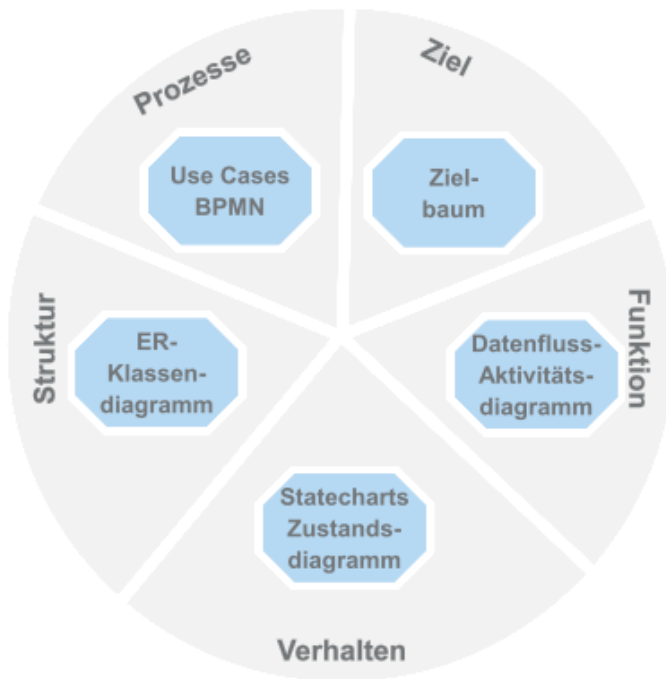
Probleme natürlichsprachiger Anforderungen

**Lesen Sie die folgenden Anforderungen gründlich durch.
Welche Probleme fallen Ihnen auf?**

1. „Einmal täglich muss kontrolliert werden, welche ausgeliehenen Bücher überfällig sind“
2. „Das System soll Mahnungen verschicken“
3. „Bei jeder Ausleihe sollen die Daten auf der Anzeige erscheinen“
4. „Durch das Aufstellen von Selbstbedienungsstationen soll das Ausleihen und Zurückgeben von Büchern beschleunigt werden“

Anforderungen dokumentieren (3)

Anforderungen modellbasiert dokumentieren:



Bei uns liegt Fokus auf OO-Methoden:

- Use Cases
- Aktivitätsdiagramme
- Zustandsdiagramme
- Klassendiagramme
- ...

→ **Anforderungsdokumente bestehen i.d.R. aus Prosa und Modellen**

Anforderungen prüfen und abstimmen

Anforderungen prüfen und abstimmen (1)

Jedes Artefakt einer Softwareentwicklung ist einer Qualitätsprüfung zu unterziehen!

Für die Anforderungsspezifikation beinhaltet das zwei Schritte:

– **Analyse der Anforderungsspezifikation:**

- Bei natürlichsprachigen Anforderungen: manuell
- Bei formal beschriebenen Anforderungen: manuell und automatisiert

– **Validierung der Anforderungsspezifikation:**

(Prüfen, ob das gewünschte Produkt richtig beschrieben ist)

- Prüfen jeder Anforderung gegen die Visionen und Ziele
- Stakeholder-Review

Anforderungen prüfen und abstimmen (2)

Analyse der Anforderungsspezifikation

Prüfung der Anforderungen im Hinblick auf die Qualitätskriterien

- Prüfmethoden:
 - Inspektionen
 - Reviews
 - Walkthroughs, ...
- Wichtiges Hilfsmittel:
 - Checklisten, um nichts zu vergessen

Anforderungen prüfen und abstimmen (3)

Validierung der Anforderungsspezifikation

Prüfen, ob das gewünschte Produkt richtig beschrieben ist

- Prüfmethoden:
 - Prüfen jeder Anforderung gegen die Visionen und Ziele
(Trägt jede Anforderung dazu bei, die Visionen und Ziele zu verwirklichen?
Falls „nein“, Anforderung entfernen)
 - Stakeholder Review
(Stakeholder erhalten Anforderungsspezifikation zur Prüfung)
- Formale Abnahme der Anforderungsspezifikation ist empfohlen

Priorisierung und Prioritäten von Anforderungen

- **Priorisierung:**

- Grundlage von technischen und Management-Entscheidungen
- Kompromisse zwischen in Konflikt stehenden Anforderungen finden
- Releases der Software planen (zuerst die “wichtigen” Anforderungen)

- **Prioritäten:**

- Essentiell:
 - Anforderung muss implementiert werden, ansonsten ist System nutzlos
- Notwendig:
 - System ist ohne Implementierung der Anforderung weniger einsetzbar/effizient/effektiv
- Wünschenswert:
 - System wäre mit Implementierung der Anforderung attraktiver

Beispiel: Chat (z.B. Whatsapp)
Finden Sie Anforderungen
unterschiedlicher Prioritäten

Beispiel für Klassifikation von Anforderungen (1)

Das Kano-Modell (5 Merkmale)

1. Basis-Merkmale:

- So grundlegend und selbstverständlich, dass sie Kunden erst bei Nichterfüllung bewusst werden (implizite Erwartungen)
- Werden die Grundforderungen nicht erfüllt, entsteht Unzufriedenheit
- Werden sie erfüllt, entsteht aber keine Zufriedenheit
- Nutzensteigerung im Vergleich zur Differenzierung am Wettbewerber ist sehr gering
- **Am Beispiel Auto:** Sicherheit, Rostschutz

2. Leistungs-Merkmale:

- Sind dem Kunden bewusst,
- Sie beseitigen Unzufriedenheit oder schaffen Kundenzufriedenheit abhängig vom Ausmaß der Erfüllung
- **Am Beispiel Auto:** Fahreigenschaften, Beschleunigung, Lebensdauer, Verbrauch

Beispiel für Klassifikation von Anforderungen (2)

Das Kano-Modell (5 Merkmale)

3. Begeisterungs-Merkmale:

- Sind Nutzen stiftende Merkmale, mit denen Kunde nicht unbedingt rechnet
- Sie zeichnen das Produkt gegenüber der Konkurrenz aus und rufen Begeisterung hervor
- Kleine Leistungssteigerung kann zu einem überproportionalen Nutzen führen
- Differenzierungen gegenüber der Konkurrenz können gering sein, der Nutzen enorm
- **Am Beispiel Auto:** Sonderausstattung, besonderes Design, Hybridtechnologie

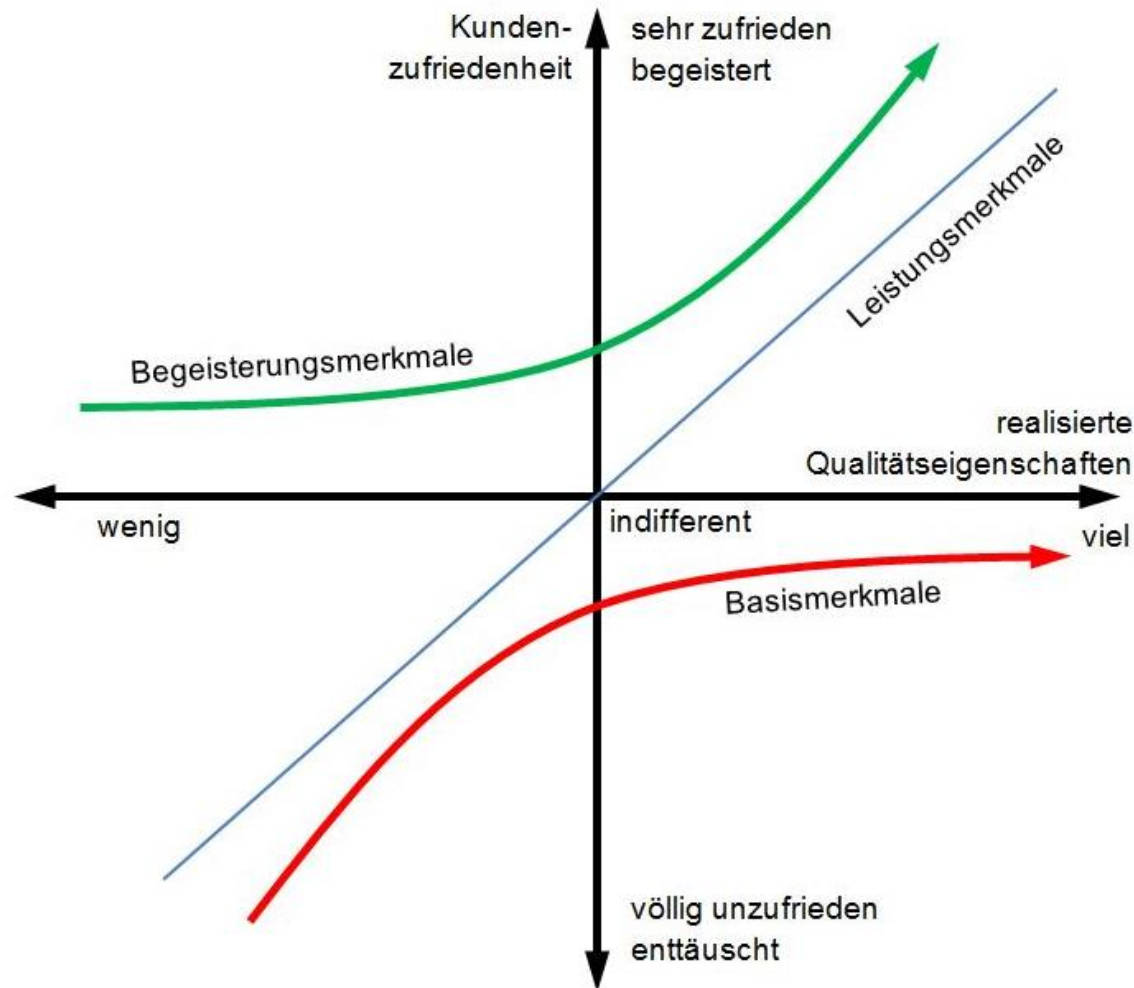
4. Unerhebliche Merkmale:

- Sind sowohl bei Vorhandensein wie auch bei Fehlen ohne Belang für Kunden
- Sie können daher keine Zufriedenheit stiften, führen aber auch nicht zu Unzufriedenheit
- **Am Beispiel Auto:** je nach Kundengruppe z.B. Automatikgetriebe, Schiebedach

5. Rückweisungs-Merkmale:

- Führen bei Vorhandensein zu Unzufriedenheit
- Bei Fehlen jedoch nicht zu Zufriedenheit
- **Am Beispiel Auto:** Rostflecken, abgelaufener TÜV

Veranschaulichung des Kano-Modells









Beispiel: Kano-Modell für intelligentes Krankenbett (1 von 2; s. Übung)

Wörtliche Kundenaussage	Aussage für Hersteller	Anforderung	Kategorie
Nie ist jemand erreichbar	Erreichbarkeit der Hotline nicht sichergestellt	Hotline im 24/7-Betrieb	Basis
Der Patient muss die gewünschte Position schnell und einfach einstellen können			
Das Bett muss beweglich sein und Türen passieren			
Ich habe für die Reinigung täglich 2 Minuten Zeit			
Eine App, um den Zustand zu prüfen, wäre toll			
Ein Bett, das selbständig rund um die Uhr Vitalfunktionen überwacht			
Ein selbstreinigendes Bett wäre eine echte Innovation			

[Quelle: produktmanager-blog.de]

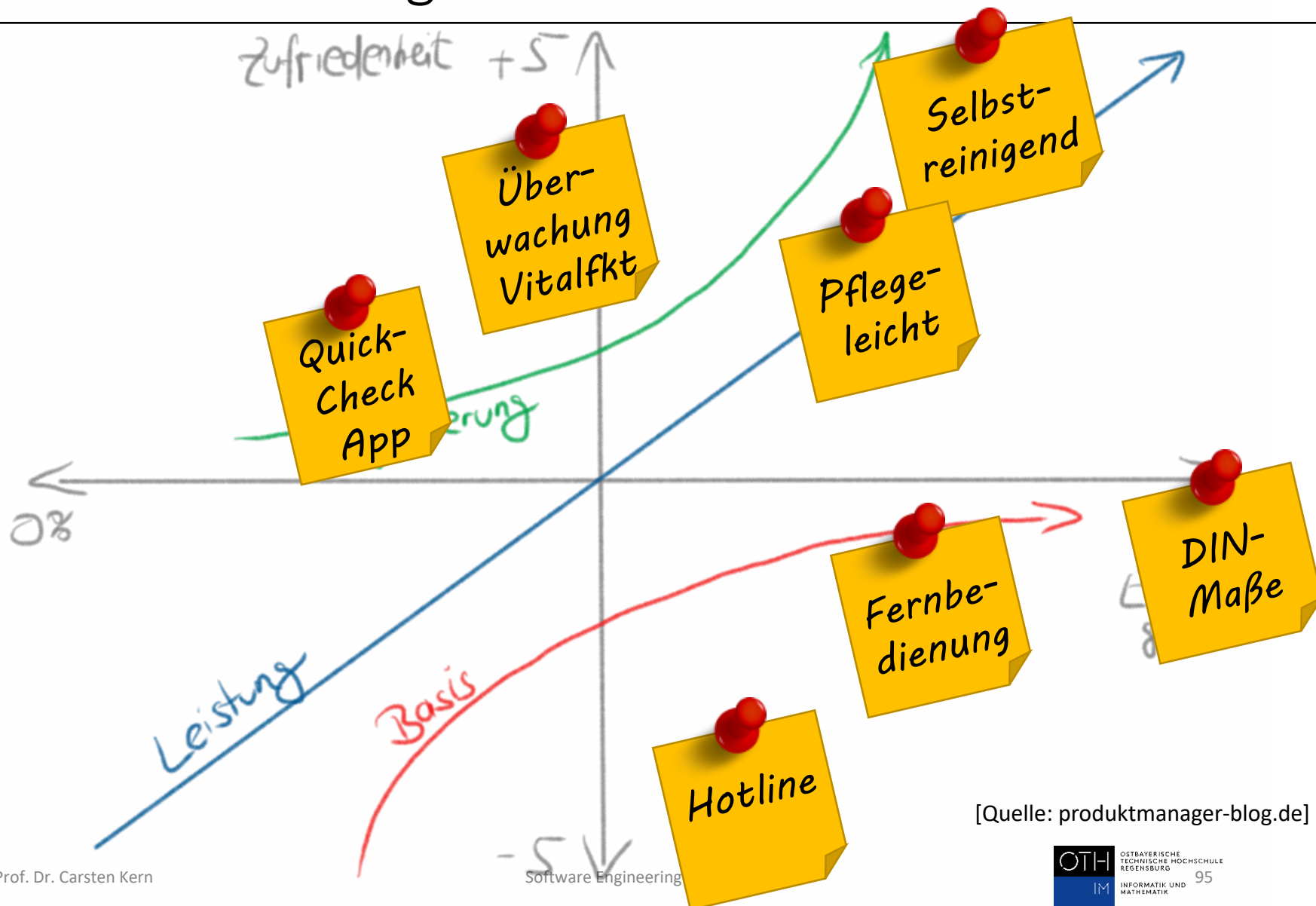
Beispiel: Kano-Modell für intelligentes Krankenbett (2 von 2; s. Übung)

Wörtliche Kundenaussage	Kategorie	Zufriedenheit	Erfüllungsgrad	Aufwand/Nutzen	Priorität
Nie ist jemand erreichbar	Basis	-3	66%		A
Der Patient muss die gewünschte Position schnell und einfach einstellen können	Basis	-1	75%		B
Das Bett muss beweglich sein und Türen passieren	Basis	0	100%		Erfüllt
Ich habe für die Reinigung täglich 2 Minuten Zeit	Leistung	3	70%		A
Eine App, um den Zustand zu prüfen, wäre toll	Begeisterung	2	35%		A
Ein Bett, das selbständig rund um die Uhr Vitalfunktionen überwacht	Begeisterung	4	45%		B
Ein selbstreinigendes Bett wäre eine echte Innovation	Begeisterung	5	80%		C

[Quelle: produktmanager-blog.de]

Beispiel: Kano-Modell für intelligentes Krankenbett

Veranschaulichung



Anforderungen verwalten

Anforderungen verwalten und managen

- Üblicherweise Tool-gestützt
- Wichtige Elemente:
 - Änderungen
 - Nachverfolgbarkeit
 - Versionierung

Risiken des Requirement Engineerings

Risiken des Requirement Engineerings

1. Kunden im Projekt **unzureichend repräsentiert**
2. Kritische Anforderungen werden **übersehen**
3. **Nur funktionale** Anforderungen berücksichtigt
4. Anforderungen werden **unkontrolliert geändert**
5. Anforderungen **beschreiben den Entwurf**
6. Anforderungen werden **nicht auf Qualität geprüft**
7. Anforderungen werden **perfektioniert**

Hauptprobleme beim Requirements Engineering?

- Hauptproblemquellen bei Software-Projekten:
 - Anforderungen unvollständig 13.1%
 - Kunden nicht ausreichend einbezogen 12.4%
 - Mittel nicht ausreichend 10.6%
 - Unrealistische Erwartungen 9.9%
 - Mangelnde Unterstützung durch Mgmt. 9.3%
 - Änderungen in Anforderungen 8.7%
 - mangelnde Planung 8.1%
 - ...

[Quelle: [CHAOS-Report](#), Standish Group 1995]

Weitere Probleme beim Requirements Engineering

- Kunde weiß nicht, was er will
- Unterschiedliche Terminologie (Kunde vs. Projektleiter vs. Programmierer)
- Anforderungen ändern sich während Analyse und Entwicklung
- Verschiedene Stakeholder → Widersprüchliche Anforderungen
- Neue Stakeholder kommen in das Projekt
- Beeinflussung durch politische/organisatorische Faktoren

Ein toller Comedy-Sketch, der diese Probleme beschreibt:

<https://www.youtube.com/watch?v=BKorP55Aqvg>

Best practices des Requirements Engineerings

1. Kunden und Benutzer einbinden
2. Identifizieren und Konsultieren aller möglichen RE-Quellen
3. Erfahrene Projektmanager und Teammitglieder einsetzen
4. 15 – 30 % der Ressourcen für RE
5. Anforderungsschablonen und Beispiele zur Verfügung stellen
6. Gute Beziehungen zu allen Beteiligten und Betroffenen herstellen
7. Anforderungen priorisieren
8. Ergänzende Modelle gemeinsam mit Prototypen entwickeln
9. Eine Nachverfolgungsmatrix pflegen
10. Peer Reviews durch Benutzer, Szenarien und Walk-Throughs zum Validieren und Verifizieren der Anforderungen

Welche Auswirkungen?

Zeitliche Änderung von Anforderungen

Zeitliche Änderung von Anforderungen

Wer hat Beispiele?

1910



1960



2000



20??



1990



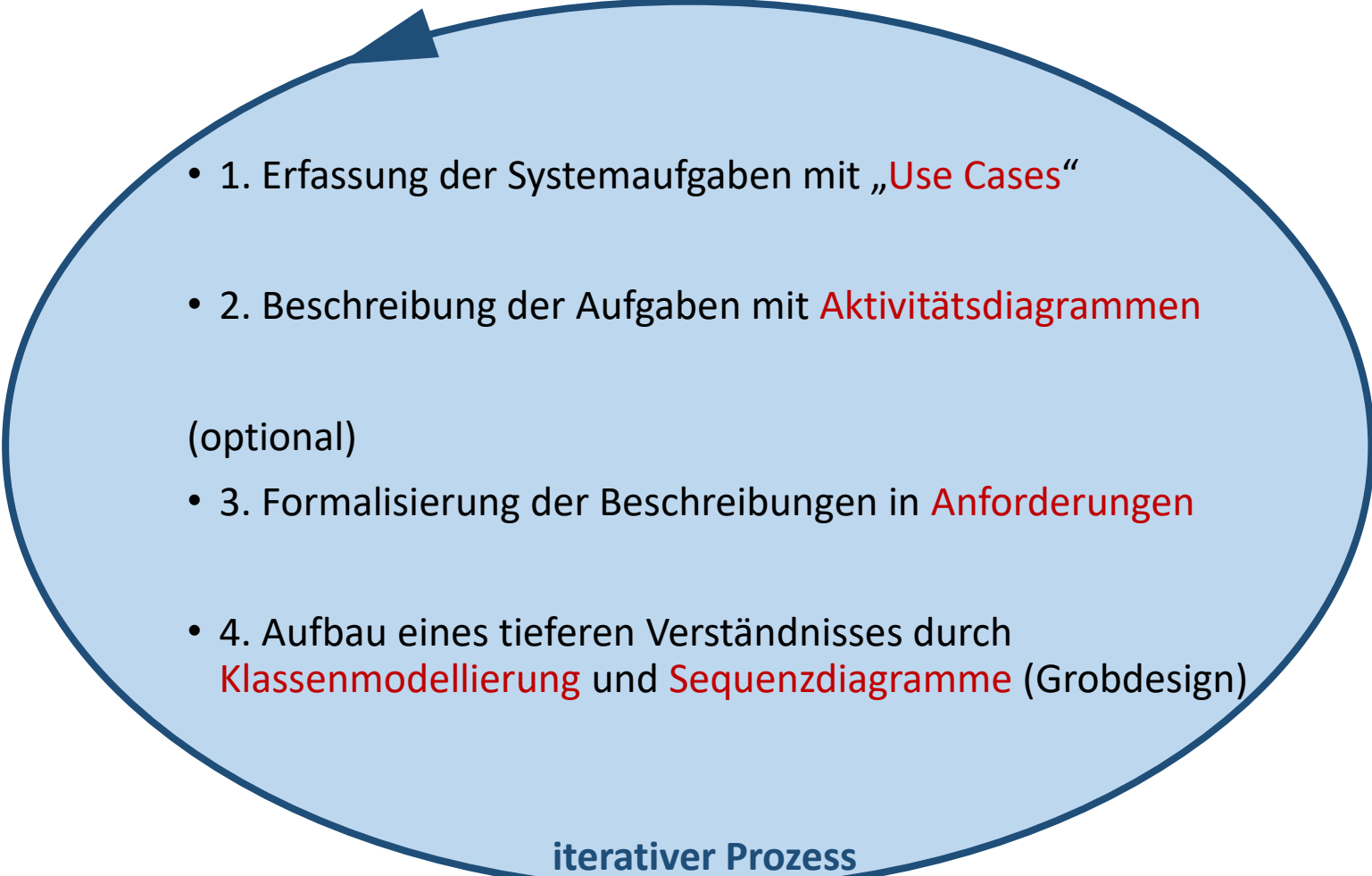
2012



- schneller, sicherer, preisgünstiger, weiter reisen
- schnurlos Telefonieren
- Mobil kommunizieren (SMS)
- Handlichkeit
- Benutzerfreundlichkeit
- Leistungsfähigkeit
- Computer-Ersatz

Erfassung von Anforderungen mithilfe von Use Cases

Der Analyseprozess: Requirements Analysis

- 
- 1. Erfassung der Systemaufgaben mit „**Use Cases**“
 - 2. Beschreibung der Aufgaben mit **Aktivitätsdiagrammen**
 - (optional)
 - 3. Formalisierung der Beschreibungen in **Anforderungen**
 - 4. Aufbau eines tieferen Verständnisses durch **Klassenmodellierung** und **Sequenzdiagramme** (Grobdesign)

iterativer Prozess

Erfragung des “WAS”?

Zentrale Fragen:

- **Was** sind die Hauptaufgaben des Systems?
 - **Wer** ist an den Aufgaben beteiligt?
 - **Welche Schritte** gehören zur Aufgabenerfüllung?
- Aufgaben werden als **Use Cases** (Anwendungsfälle) beschrieben
- Beteiligte werden als **Akteure** festgehalten
(können meist aus der Menge der Stakeholder entnommen werden)

Definition: Use Case (Anwendungsfall)

Ein Use Case:

- In Sprache der Stakeholder (**natürliche Sprache**)
- Beschreibt konsistente und zielgerichtete **Interaktion des Benutzers mit System**
- Anfang der Interaktion: **fachlicher Auslöser**
- Ende der Interaktion: **definiertes Ergebnis** von fachlichem Wert
- Beschreibt gewünschtes externes Systemverhalten aus Anwendersicht und somit Anforderungen, die das System erfüllen soll
- Beschreibt was das System leisten muss, aber nicht wie es dies leisten soll
- Unterscheidung in Geschäftsanwendungsfall (**business use case**) formuliert aus Geschäftssicht (z.B. Vertriebsprozess vom Anfang) und Systemanwendungsfall (**system use case**) formuliert aus Sicht der durch die neue SW zu lösenden Aufgabe

Use Cases (Anwendungsfälle)

- Idee:

- Gliederung der Funktionalität des Systems in logisch zusammenhängende und handliche, funktionale Einheiten
- Beschreibung in standardisierter Form

- Definition: Use Case

Abgeschlossene, zusammenhängende Einheit
Teil der Funktionalität des Systems

Struktur der Use-Case-Beschreibung

Dozent trägt Vorlesung in Vorlesungsverzeichnis ein



- Titel:
- Kurzbeschreibung:
- Akteure:
- Vorbedingungen:
- Beschreibung des Ablaufs:
- Auswirkungen:
- Anmerkungen:

Aktoren

- **Definition: Akteur**

Objekt der Systemumgebung, das mit dem System interagiert (und einen oder mehrere Anwendungsfälle auslösen kann)

- **Merkmale von Aktoren:**

- Können Personen, externe Geräte oder verbundene Nachbarsysteme sein
- Tauschen mit dem System Nachrichten aus
- Können Sender und/oder Empfänger von Nachrichten sein
- Sind i. Allg. nicht Bestandteil des Systems
- Häufig müssen aber Daten von Ihnen verwaltet werden (Login, Berechtigungen,...)

Use Cases: Anmerkungen (1)

- Verwende für den Use Case eine **sinnvolle Bezeichnung**, die mindestens aus einem **echten Substantiv** und einem **aktiven Verb** ("Antrag erfassen") oder dem zugehörigen Gerundium ("Antragserfassung") besteht!
- Definiere zuerst **fachlichen Auslöser** und **fachliches Ergebnis**, um Anfang und Ende des Use Cases festzulegen!
- Formuliere Use Case **so abstrakt wie möglich** und **so konkret wie nötig**!
- Betreibe *zunächst* keine Zerlegung in abgeleitete, sekundäre Use Cases!
- Standardisiere die Sprache in den Use Cases!

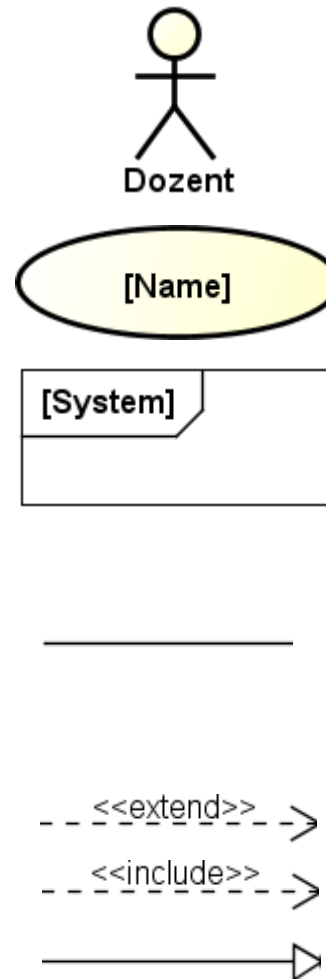
Use Cases: Anmerkungen (2)

- Use Cases eignen sich nicht zur funktionalen Zerlegung
 - D.h. Use Case beschreibt **keine einzelnen Schritte**, Operationen oder Transaktionen, **sondern relativ große Abläufe**
 - Beispiel: anstatt "Vertrag drucken", "Kunden-Nr. erzeugen" grobe Darstellung nutzen wie "Neuen Kunden aufnehmen"
- Es wird **keine Ablaufreihenfolge** definiert. Hierzu gibt es andere Ausdrucksmittel, z.B. Aktivitätsdiagramme
- Use Cases belassen das Sprachmonopol beim Stakeholder, wodurch die Use Cases angreifbarer und besser kritisierbar werden

Use-Case-Diagramme (Standardelemente)

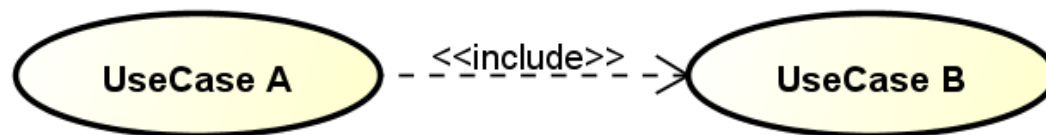
Elemente:

- Akteur
- Use Case mit Bezeichner [Name]
- Systemgrenze mit Systembezeichner [System]
- Kommunikationsbeziehung zwischen Aktoren und Use Cases
- Beziehung zwischen Anwendungsfällen



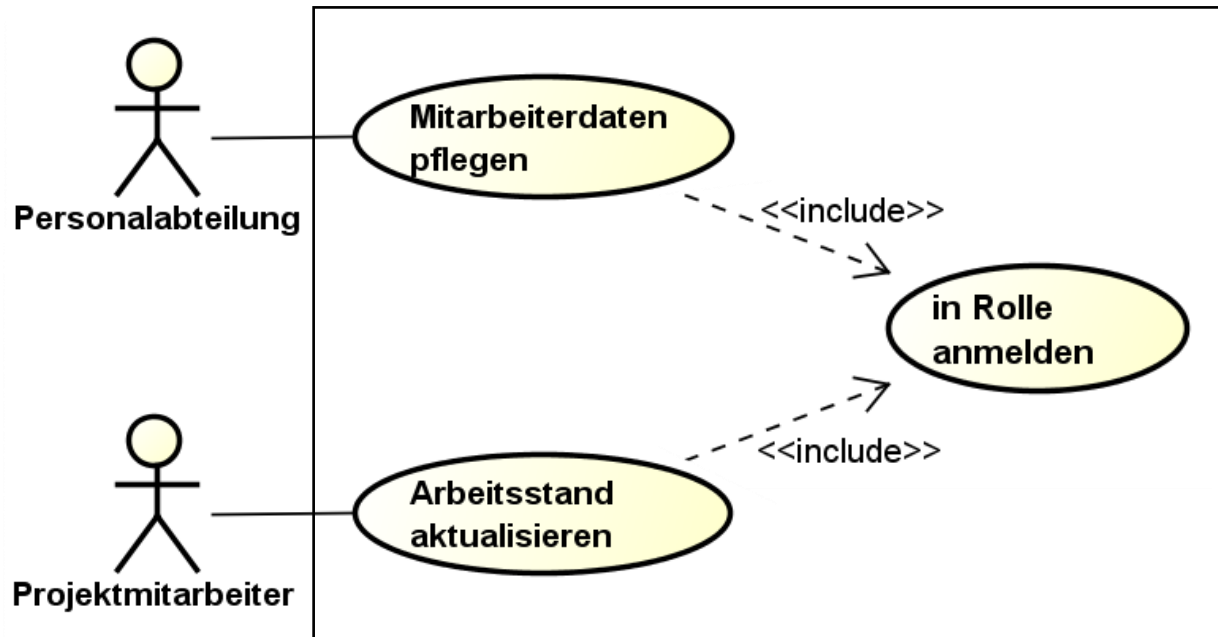
Analyse von Use Case Dokumentationen

- Bei der Dokumentation von Use Cases kann es passieren, dass identische Abläufe **mehrfach beschrieben** werden
- Diese (nicht trivialen) Abläufe können **als eigene Use Cases ausgegliedert** werden; man sagt dann „ein Use Case nutzt einen anderen Use Case“
- UML-Darstellung:



- In spitzen <<Klammern>> stehen so genannte **Stereotypen**, mit denen man UML-Elementen zusätzliche Eigenschaften zuordnen kann

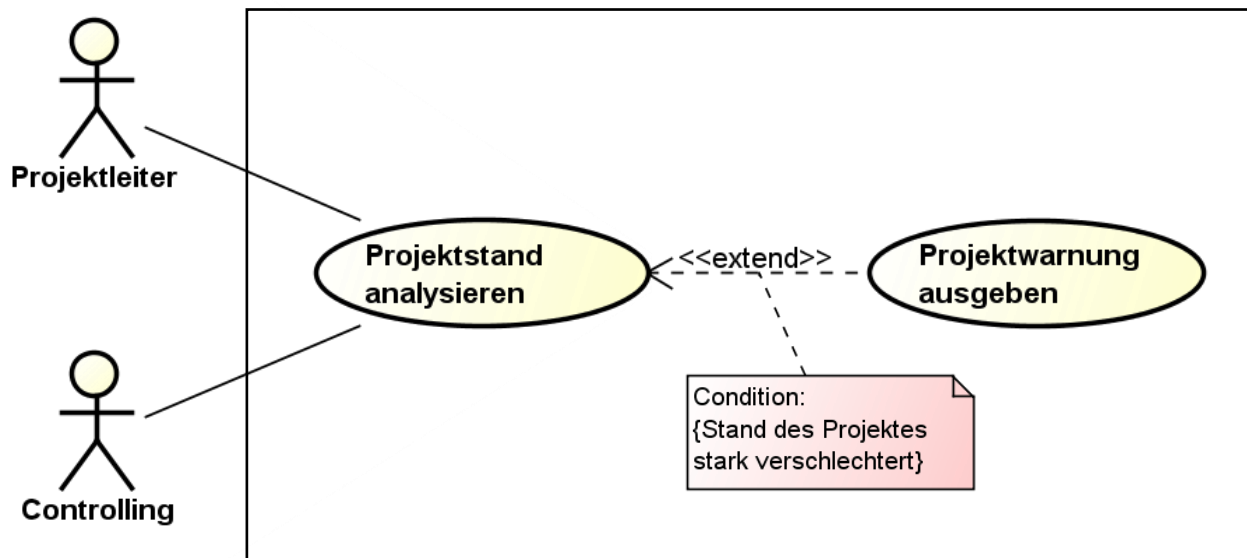
Beispiel zum <<include>>-Stereotyp



Bedeutung?

Das <<extend>>-Stereotyp

- (Seltenere) **Variation des erweiterten Use Cases**
- Wird **nur unter bestimmter Bedingung ausgeführt**, z.B. Sonderfall oder Fehlerbehandlung
- eigentlicher Use Case **nicht durch Spezialfälle überfrachtet**



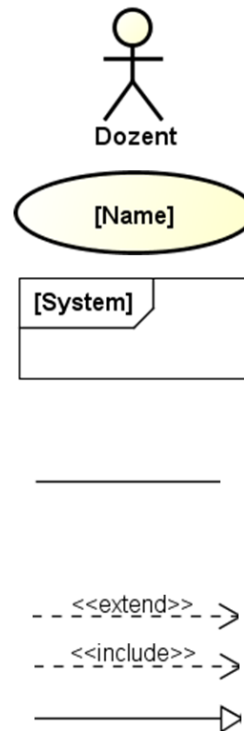
Jetzt sind Sie dran ...



Modellieren Sie den Besuch eines Studierenden in einem Club (Disko) in einem Use Case Diagramm

Zur Erinnerung:

- Elemente:
 - Akteur
- Use Case mit Bezeichner [Name]
- Systemgrenze mit Systembezeichner [System]
- Kommunikationsbeziehung zwischen Akteuren und Use Cases
- Beziehung zwischen Anwendungsfällen



Jetzt sind Sie dran ...

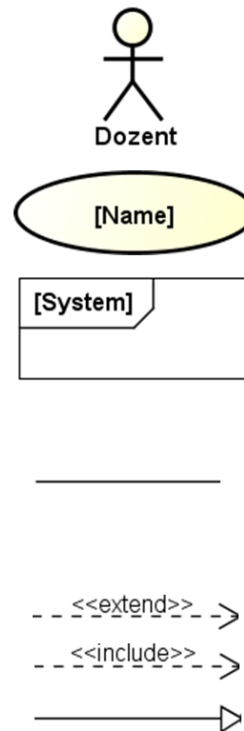


Modellieren Sie einen Online-Shop in einem Use-Case-Diagramm

- Das Use-Case-Diagramm soll 6 Use Cases (5 wesentliche Funktionen) und dabei mindestens eine include- und eine extend-Beziehung aufweisen

Zur Erinnerung:

- Elemente:
 - Aktor
 - Use Case mit Bezeichner [Name]
 - Systemgrenze mit Systembezeichner [System]
 - Kommunikationsbeziehung zwischen Aktoren und Use Cases
 - Beziehung zwischen Anwendungsfällen



Zusammenfassung:



Zusammenfassung: Überblick

