Universität Zürich[UZH]

Institut für Informatik

Martin Glinz Thomas Fritz

Software Engineering

Kapitel 4

Spezifikation von Anforderungen

4.1 Grundlagen

4.2 Problemüberblick

4.3 Dokumentation von Anforderungen

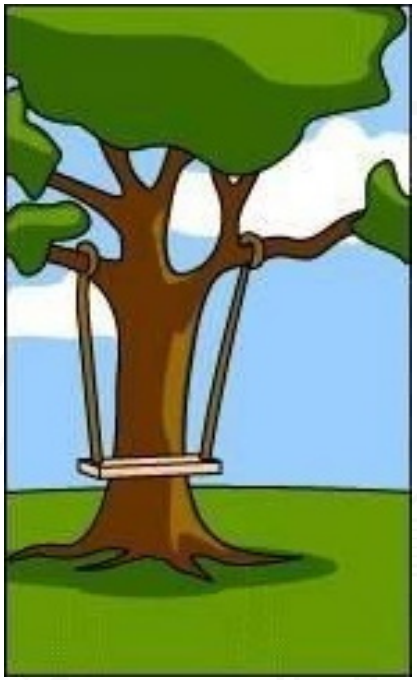
4.4 Prozesse und Praktiken

4.5 Spezifikationsmethoden und -sprachen

4.6 Verwalten von Anforderungen

Bedürfnis

Was der Kunde wollte

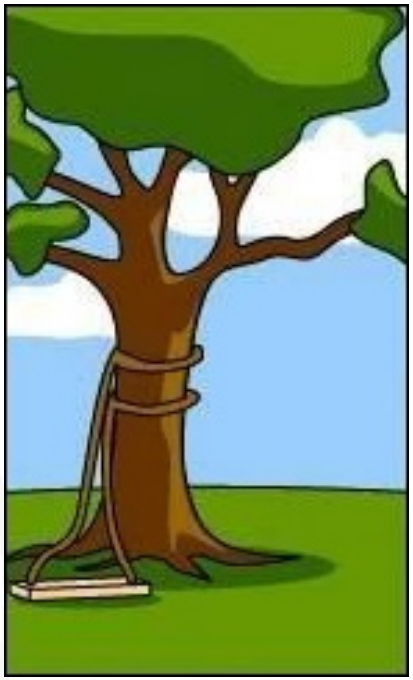


Analyse

Was der Analytiker verstand

Entwurf

Was der Architekt konzipierte



Installiertes System

Was die Programmierer implementierten

Wir müssen die Anforderungen kennen.

DEFINITION. Anforderung –

1. Ein Bedürfnis eines Interesseneigners (Stakeholders).
2. Eine von einem System (oder einer Komponente) verlangte Fähigkeit oder Eigenschaft.
3. Eine dokumentierte Darstellung eines Bedürfnisses, einer Fähigkeit oder Eigenschaft.

DEFINITION. Anforderungsspezifikation – Eine systematisch dargestellte und gegebenen Kriterien genügende Sammlung von Anforderungen, typischerweise an ein System oder eine Komponente.

[Glinz 2013]

Aufgaben

Anforderungen kennen heißt:

- Ermitteln
- Analysieren
- Dokumentieren
- Prüfen
- Verwalten

von Anforderungen

➔ Requirements Engineering (Anforderungstechnik)

Requirements Engineering

DEFINITION. Requirements Engineering (Anforderungstechnik) – Das systematische, disziplinierte Spezifizieren und Verwalten von Anforderungen mit dem Ziel

(1) Die Wünsche und Bedürfnisse der Interesseneigner zu verstehen,

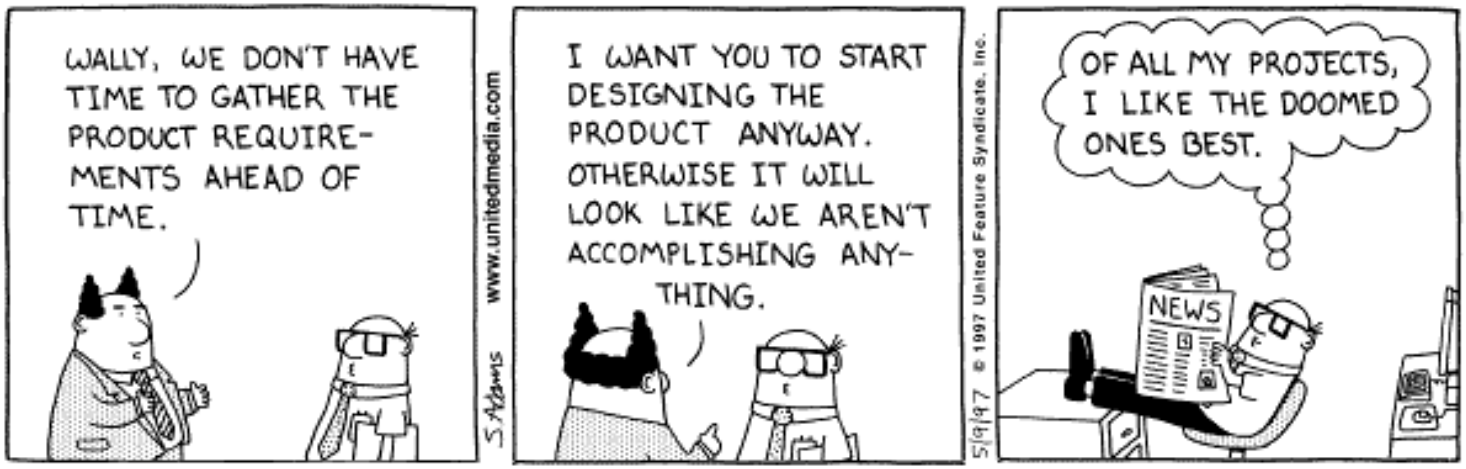
(2) Einen Konsens über die relevanten Anforderungen zu erzielen und diese Anforderungen systematisch zu dokumentieren und zu verwalten,

(3) Das Risiko zu minimieren, dass das zu entwickelnde System die Wünsche und Bedürfnisse der Interesseneigner nicht erfüllt.

[Glinz 2013]

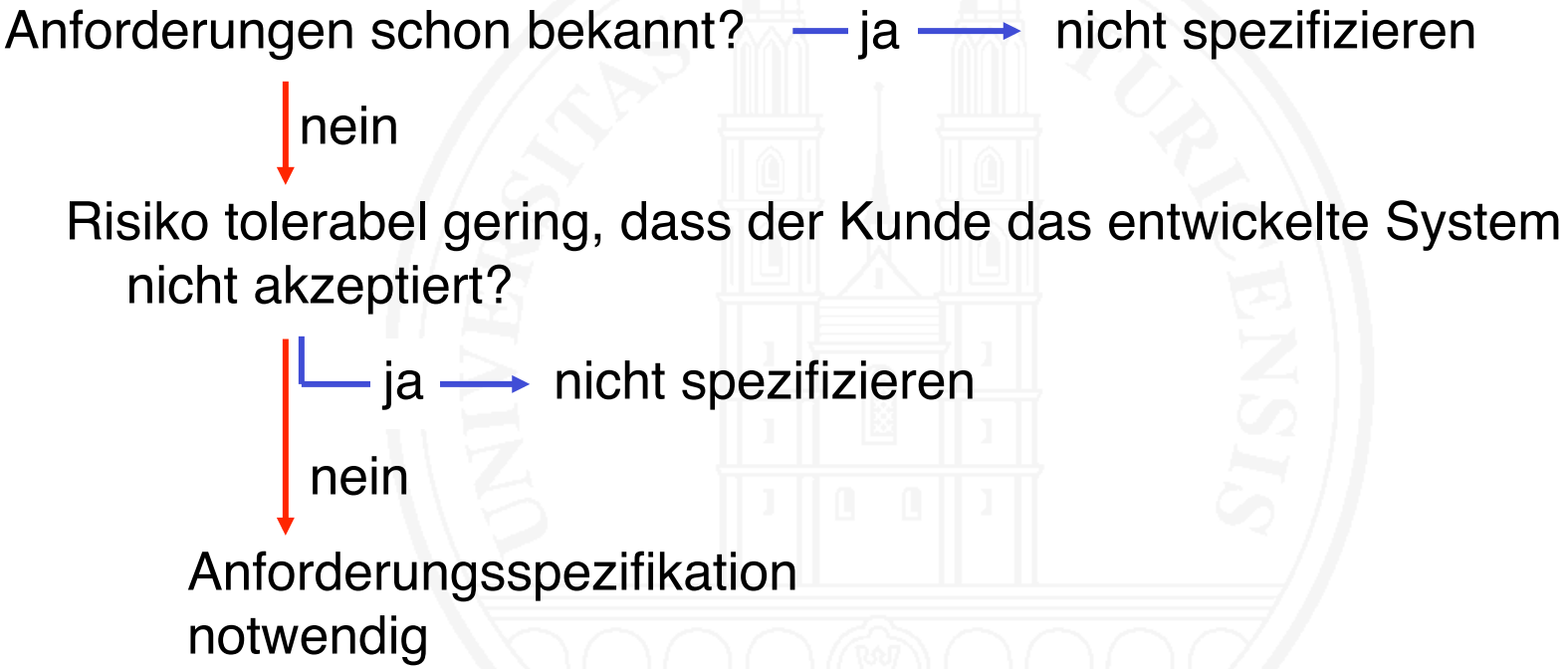
Requirements Engineering – 2

- Kundenzufriedenheit schaffen
- Die Probleme der Interesseneigner verstehen
- Aufwand und Wert in Balance
- Aber: warum nicht gleich programmieren?



Weil das Risiko meist zu hoch ist.

Anforderungsspezifikation und Risiko



Anforderungsspezifikation und Risiko – 2

„Wir haben keine Zeit für eine vollständige Spezifikation."

„Ist uns zu teuer!"

„Bei agilem Vorgehen genügen grobe Stories vollständig."

➪ falscher Ansatz

Richtige Frage: „Wie viel müssen wir tun, damit das Risiko eine Größe annimmt, die wir bereit sind zu akzeptieren?"

Merkregel:

Der *Aufwand* für das Requirements Engineering soll *umgekehrt proportional* zum *Risiko* sein, das man bereit ist, einzugehen.

Pflichtenheft

Nur im Deutschen gebräuchlich und leider mehrdeutig

Pflichtenheft –

1. Vom Lieferanten erarbeitete Lösungsvorgaben für ein System, in der Regel auf der Basis eines *Lastenhefts*.
2. Synonym für Anforderungsspezifikation (in der Regel für ein softwarebasiertes System, erstellt vom Lieferanten)
3. Anforderungsspezifikation unter Einschluss der für den Kunden relevanten Teile des Projektplans.

Lastenheft – Durch einen Kunden angefertigte grobe Beschreibung der benötigten Fähigkeiten eines Systems aus Kundensicht.

Nutzen einer Anforderungsspezifikation

- Kosten senken
  - Geringere Herstellungskosten (Senken der Fehlerkosten!)
  - Geringere Pflegekosten
- Risiken verkleinern
  - Kundenerwartungen besser erfüllen
  - Zuverlässigere Prognosen für Termine und Kosten

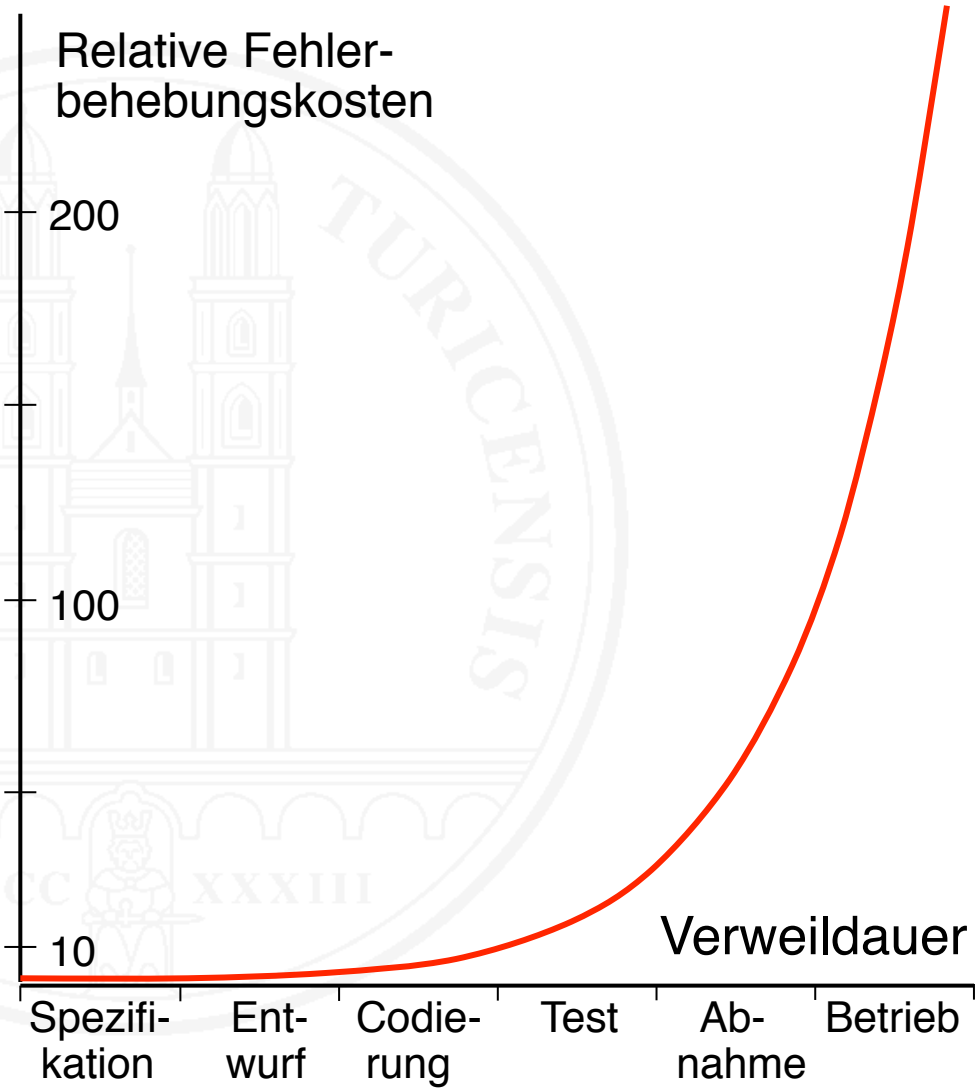
Mehr verdienen!

Zufriedenere Kunden!

☞ Die wirtschaftliche Wirkung von Requirements Engineering ist indirekt; das RE selbst kostet nur!

Anforderungsspezifikation – Warum (2)

Kosten für die Behebung von Fehlern abhängig von ihrer Verweildauer in der Software



4.1 Grundlagen

4.2 Problemüberblick

4.3 Dokumentation von Anforderungen

4.4 Prozesse und Praktiken

4.5 Spezifikationsmethoden und -sprachen

4.6 Verwalten von Anforderungen

Ein Beispielproblem

In einem Skigebiet gibt es eine Reihe von Sesselliften. Die Kunden kaufen mit RFID ausgerüstete Tageskarten. Die Zugangskontrolle erfolgt über Drehkreuze, die mit RFID-Lesern ausgerüstet sind. Wenn ein Leser eine gültige Tageskarte erkennt, wird das Drehkreuz für eine Drehung entriegelt, so dass der Kunde passieren kann.

Interesseneigner (Stakeholder)

Wer ist «der Kunde»?

In unserem Beispielproblem: Nur die Skifahrer / Boarder?

In Wirklichkeit sind viele Personen in vielen Rollen beteiligt

DEFINITION. Interesseneigner (Stakeholder) – Eine Person order Organisation, welche die Anforderungen an ein System direkt oder indirekt beeinflusst.

Synonyme: Interessenvertreter, Beteiligter

[Glinz und Wieringa 2007]
[Macaulay 1993]

Sichten

Die gleiche Sache.
Verschiedene Sichten.

Verschiedene Sichten verschiedener Interesseneigner müssen berücksichtigt werden

[Nuseibeh, Kramer und Finkelstein 2003]

Konsens und Variabilität

Die Sichten / Bedürfnisse verschiedener Interesseneigner können in Konflikt zueinander stehen

Requirements Engineering bedeutet

- Erkennen von Konflikten und Inkonsistenzen
- Verhandeln
- Moderieren
- Konsens finden

Aber auch: feststellen, wo Variabilität erforderlich ist

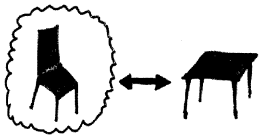
Qualität

(vgl. Kapitel 2)

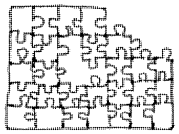
DEFINITION. Qualität – Der Grad, in dem ein Satz inhärenter Merkmale [eines Produkts, Prozesses, Gegenstands, etc.] Anforderungen erfüllt.

Undefinierte Anforderungen bedeuten undefinierte Qualität.

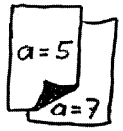
Qualitätsmerkmale einer guten Anforderungsspezifikation



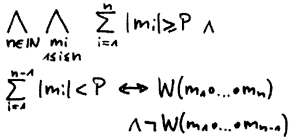
Adäquat – beschreibt das, was der Kunde will bzw. braucht



Vollständig – beschreibt alles, was der Kunde will bzw. braucht



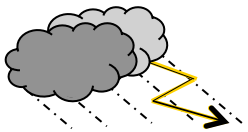
Widerspruchsfrei – sonst ist die Spezifikation nicht realisierbar



Verständlich – für alle Beteiligten, Kunden wie Informatiker

Eindeutig – vermeidet Fehlinterpretationen

Prüfbar – feststellen können, ob das realisierte System die Anforderungen erfüllt



Risikogerecht – Umfang umgekehrt proportional zum Risiko, das man eingehen will

Welches System?

Ein paar Anforderungen für unser Beispielproblem:

Das System soll für jedes Drehkreuz die Zahl der Passagiere zählen, die dieses Drehkreuz passieren.

Drehkreuz-Steuersoftware

Das System soll eine effektive Zugangskontrolle zu den Liften des Skigebiets sicherstellen.

Anlagen, Geräte, Karten, Software, ...

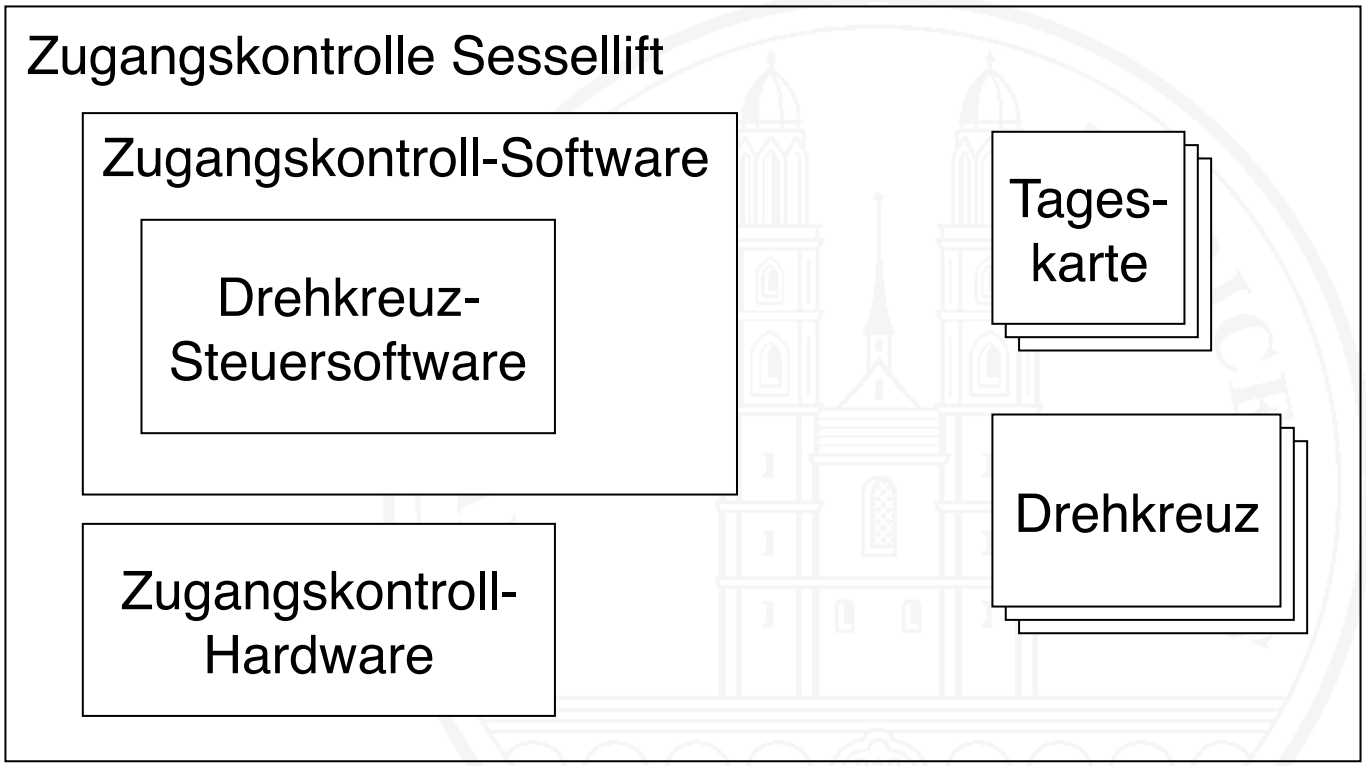
Das System soll im Temperaturbereich von -30° C to +30° C funktionieren.

Rechnerhardware und Geräte

Der Operateur soll in der Lage sein, das System in drei Betriebsarten zu fahren: normal (Drehkreuz wird bei gültiger Karte für eine Drehung entriegelt), geschlossen (alle Drehkreuze verriegelt) und offen (alle Drehkreuze entriegelt).

Zugangskontroll-Software für einen Sessellift

Systeme von Systemen

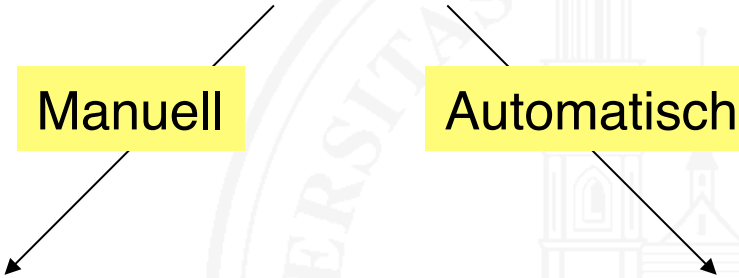


- Anforderungen haben einen Kontext
- Umgang mit mehrstufigen Anforderungen ist unvermeidlich

Anforderungen vs. Entwurfsentscheidungen

Das System soll eine effektive Zugangskontrolle zu den Liften des Skigebiets sicherstellen.

Eine Anforderung



Mögliche Entwurfs-entscheidungen

Anforderungen betr. Auswahl und Schulung von Leuten

Anforderungen betr. Drehkreuze, RFID-Karten und Steuersoftware

Anforderung auf tieferer Ebene

- Entwurfsentscheidungen beeinflussen die Anforderungen auf tieferen Ebenen
- Anforderungen und Entwurf sind miteinander verzahnt

WAS und WIE im Requirements Engineering

Volksweisheit: WAS = Spezifikation, WIE = Entwurf

Aber: ist folgender Satz eine Anforderung oder eine Entwurfsentscheidung?

> „Das System druckt eine wahlweise nach Namen oder Land alphabetisch sortierte Liste von Teilnehmern mit Nummer, Name, Vorname, Affiliation und Land. Auf jeder Seite sind unten links das Erstellungsdatum und unten rechts die Seitenzahl aufgedruckt.“

➜ WAS vs. WIE ist kontextabhängig und liefert keine brauchbare Abgrenzung zwischen Anforderungen und Entwurfsentscheidungen. Die gleiche Sache kann je nach Kontext beides sein.

Abgrenzung Anforderungen – Entwurf

- WAS vs. WIE funktioniert nicht, da abhängig von Betrachtungsebene
- Im Prozess sind Anforderungen und Entwurf nicht vollständig trennbar
- In der Dokumentation ist eine Trennung angezeigt

- Möglichkeit: operationale Abgrenzung:
  - Änderungen der Anforderungen brauchen die Zustimmung des Auftraggebers/Kunden
  - Änderungen im Entwurf kann der Auftragnehmer/Lieferant autonom vornehmen

Problemwelt und Systemwelt

① Eine Anforderung in der Problemwelt:

Das System soll für jedes Drehkreuz die Zahl der Passagiere zählen, die dieses Drehkreuz passieren.

② Abgebildet auf eine Software-Anforderung:

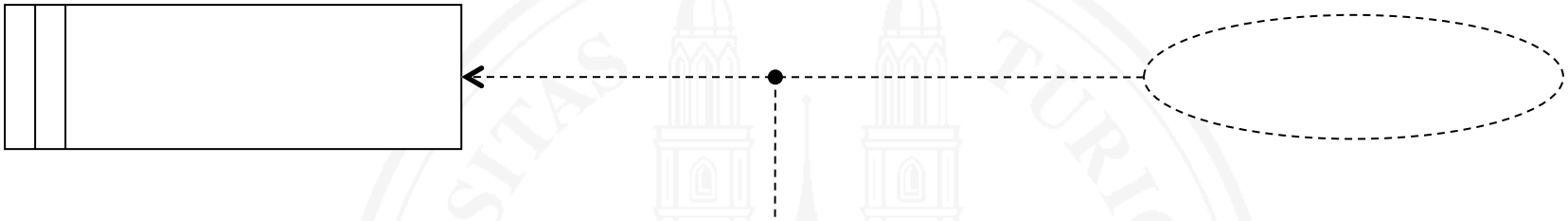
Die Drehkreuz-Steuersoftware soll die Zahl der ans Drehkreuz ausgegebenen «Entriegeln für eine Drehung»-Befehle zählen.

② erfüllt ① nur, wenn folgende Umgebungsannahmen gelten:

- Ein Entriegelungsbefehl entriegelt das Drehkreuz tatsächlich
- Wird ein Drehkreuz entriegelt, so geht eine einzelne Person hindurch
- Niemand passiert ein verriegeltes Drehkreuz

Das Anforderungsproblem (nach Jackson)

[Zave und Jackson 1997]
[Jackson 2005]



Eine Maschine mit Fähigkeiten gemäß der Spezifikation *S*

Umgebungs-eigenschaften *D*

Erwartetes Ver-halten *R* in der Realwelt

Das Anforderungsproblem (nach Jackson):

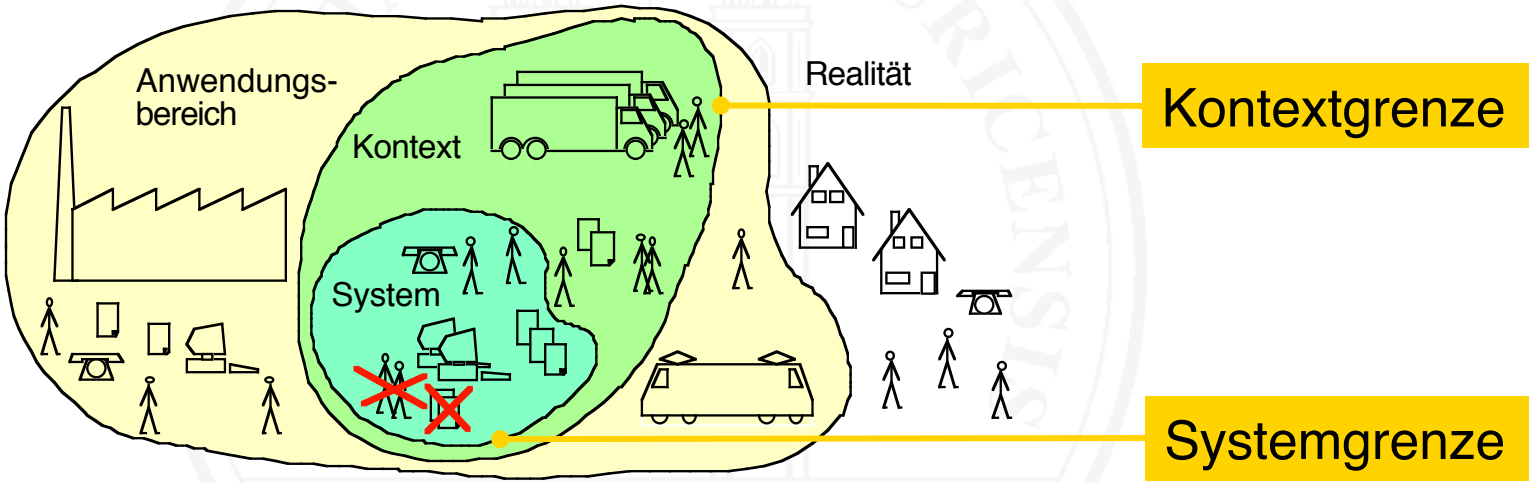
Erfüllt eine Maschine die Spezifikation *S*

und gelten die Umgebungseigenschaften *D*,

dann ist die Anforderung *R* in der Realwelt erfüllt: $S \wedge D \vdash R$

System und Kontext

Kontext – Derjenige Teil der Umgebung eines Systems, der für das Verständnis des Systems und seiner Anforderungen relevant ist



- Ein System ist eingebettet in seinen Kontext
- Requirements Engineering legt die Systemgrenze fest
- Alles außerhalb der Kontextgrenze ist nicht systemrelevant

Klassifikation von Anforderungen

Die Drehkreuz-Steuersoftware soll die Zahl der ans Drehkreuz ausgegebenen «Entriegeln für eine Drehung»-Befehle zählen.

Eine Funktion

Der Operateur soll in der Lage sein, das System in drei Betriebsarten zu fahren: normal (Drehkreuz wird bei gültiger Karte für eine Drehung entriegelt), geschlossen (alle Drehkreuze verriegelt) und offen (alle Drehkreuze entriegelt).

Ein Verhalten

Das System soll spätestens fünf Monate nach Vertragsunterzeichnung in Betrieb gehen.

Eine Projektanforderung

Das System muss das Datenschutzgesetz des Landes, in dem es installiert wird, einhalten.

Eine (gesetzliche) Randbedingung

Die Reaktionszeit von der Erkennung einer gültigen Karte bis zur Ausgabe eines «Entriegeln für eine Drehung»-Befehls muss kürzer als 0,5 s sein.

Eine Leistungsanforderung

Das System soll hochverfügbar sein.

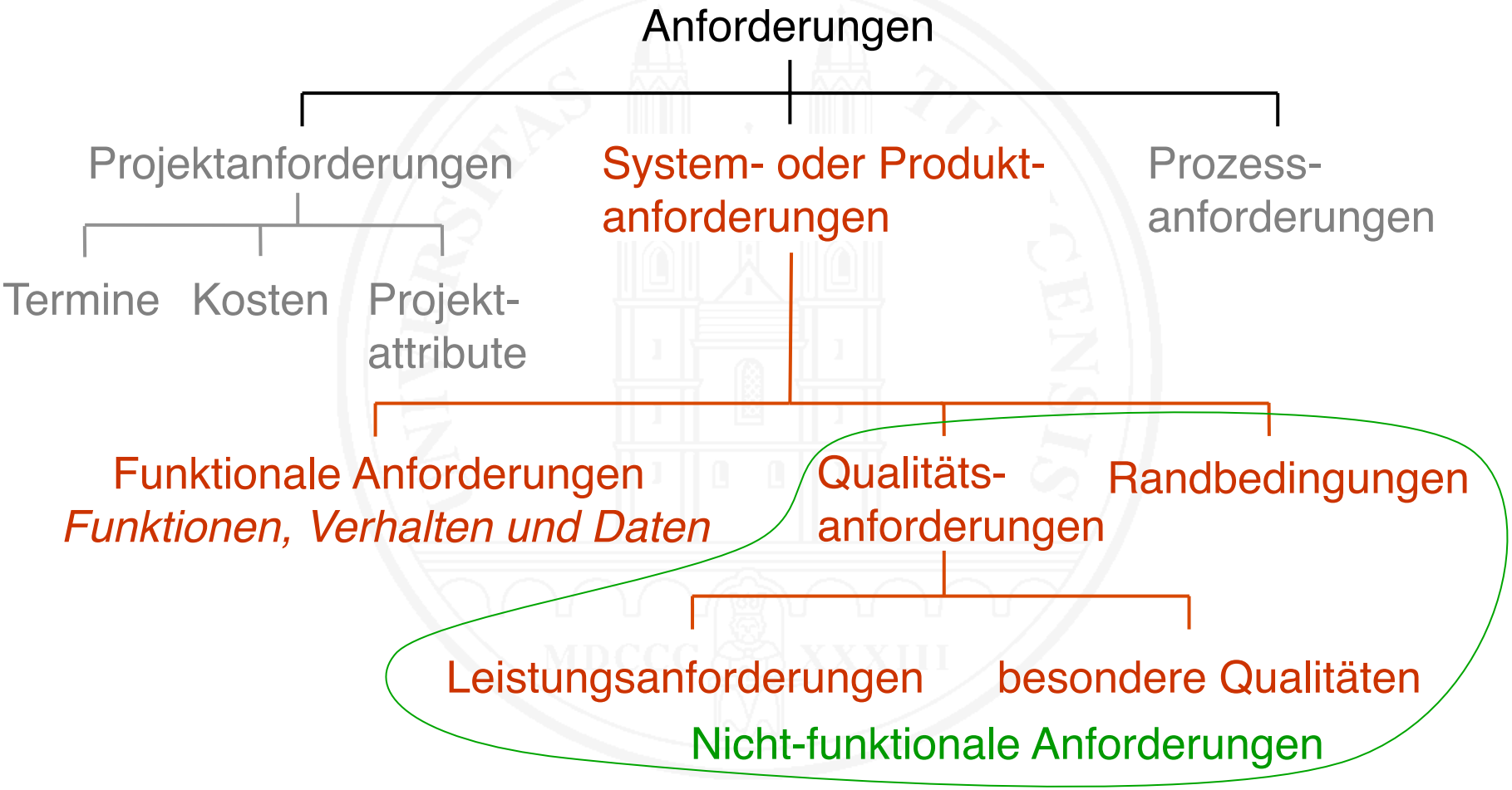
Eine spezielle Qualitätsanforderung

Es gibt verschiedene Anforderungsarten

Wird diese Anforderung gestellt, weil...	
... Systemverhalten, Daten, Eingaben oder Reaktionen auf Eingaben zu spezifizieren sind – unabhängig davon, wie dies geschehen soll?	funktionale Anforderung
... Restriktionen bezüglich Verarbeitungs-/ Reaktionszeiten, Datenmengen oder Datenraten zu spezifizieren sind?	Leistungs-anforderung
... eine spezielle Qualität, die das System aufweisen soll, zu spezifizieren ist?	besondere Qualität
... irgend eine andere Restriktion zu spezifizieren ist?	Randbedingung

Klassifikation von Anforderungen nach ihrer Art

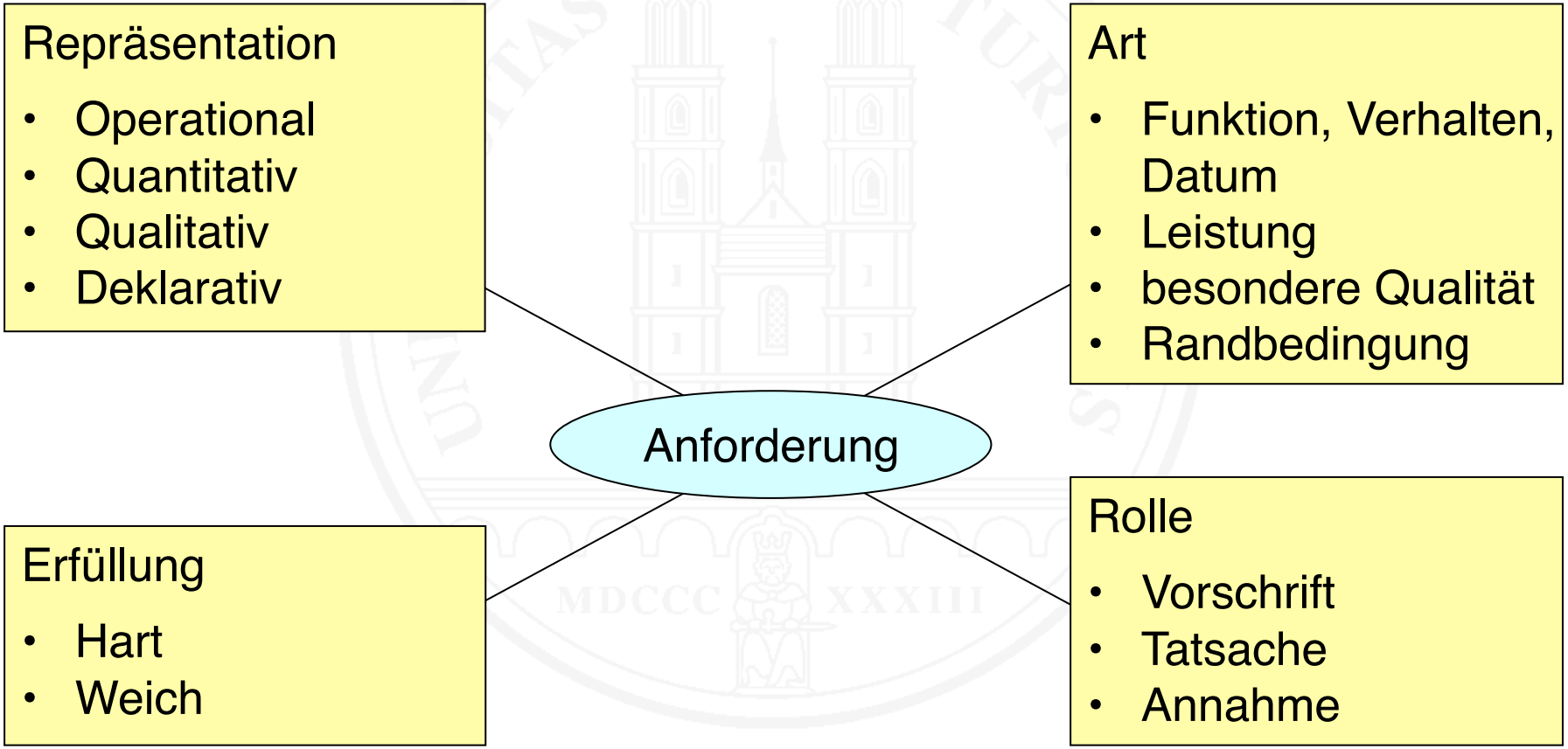
[Glinz 2007]



Facettierte Klassifikation von Anforderungen

[Glinz 2005]

Anforderungen unterscheiden sich nicht nur nach ihrer Art:



Erläuterungen zu den Facetten

Repräsentation

- Operational: „Der Kontostand wird angezeigt“
- Quantitativ: „Antwortzeit < 0,5 s“
- Qualitativ: „Das System muss hochverfügbar“
- Deklarativ: „Das System soll auf einer Linux-Plattform laufen“

Erfüllung

- Hart – Anforderung ist ganz oder gar nicht erfüllt (binäres Verhalten): „Das System soll abgelaufene Karten sperren“
- Weich – Anforderung kann graduell erfüllt sein: „Das System soll für Gelegenheitsbenutzer einfach zu bedienen sein“

Erläuterungen zu den Facetten – 2

Art

- siehe oben

Rolle

- Vorschrift – Forderung an das zu erstellende System: „Der Füllstandsensor soll alle 100 ms einmal abgetastet werden“
- Tatsache – Fakten in der Systemumgebung : „Alleinstehende werden nach Tarif A besteuert“
- Annahme – Annahmen über das Verhalten von Akteuren in der Systemumgebung: „Jeder Alarm wird vom Operateur quittiert“

Mini-Übung 4.1: Wie sind diese drei Anforderungen repräsentiert?

Anforderungsdokumentation als Brücke

Interessen-
eigner
(Stakeholder)

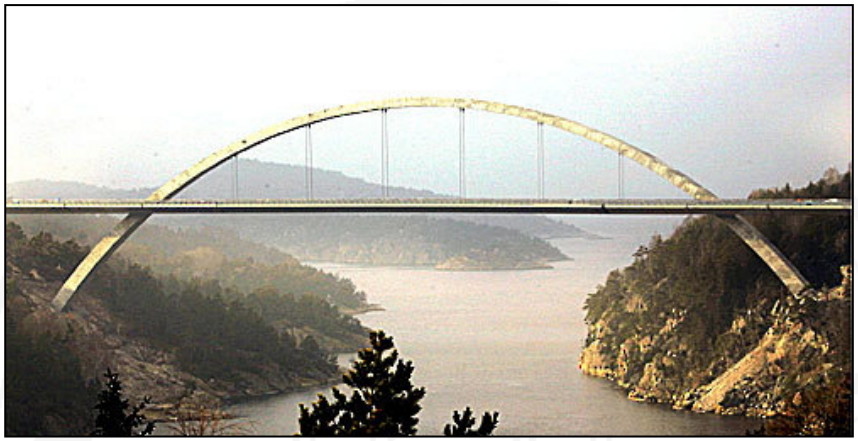


Photo © Lise Aserud / DPA

Architekten und
Programmierer

Das Bedürfnis:

- Anforderungen kommunizieren
- Eine Basis für Verträge und Abnahmeentscheidungen haben

Das Mittel: Eine Anforderungsspezifikation (als Dokument)

Formen von Anforderungsspezifikationen

- System- (oder Software-)Anforderungsspezifikation
  - Das klassische Dokument
  - Anforderungen an ein System in seinem Kontext
  - Typisch von Anforderungsingenieuren des Systemlieferanten geschrieben
  - Im Deutschen oft als Pflichtenheft bezeichnet
- Kunden-Anforderungsspezifikation
  - Anforderungen aus Kundensicht
  - Unabhängig von konkreten zu entwickelnden Systemen
  - Typisch von Anwendungsexperten der Auftraggeberseite geschrieben, ggf. unter Einbezug externer Berater
  - Im Deutschen: Kernbestandteil von Lastenheften*

*Lastenhefte enthalten die Gesamtheit der Forderungen betreffend Leistungen und Lieferungen eines Auftraggebers an einen Auftragnehmer; mit der Kunden-Anforderungsspezifikation als Kernbestandteil

Formen von Anforderungsspezifikationen – 2

- Geschäfts-Anforderungsspezifikation
  - Kompaktes Dokument, welches Geschäftsziele und/oder Geschäftsanforderungen dokumentiert
  - Dient als Entscheidungsgrundlage für das Aufsetzen von Entwicklungsprojekten
- Systemvision
  - Geschäftsziele und Skizze einer möglichen Lösung
  - Typisches Startdokument bei agiler Software-Entwicklung
- Benutzergeschichten (user stories)
  - Sammlung von Einzelanforderungen aus Benutzersicht bei agiler Software-Entwicklung
  - Erfordert Präzisierung durch Abnahmetestfälle
  - Wird fortlaufend erweitert und angepasst

(zu agiler Software-Entwicklung siehe Kap. 13)

Inhalt einer Anforderungsspezifikation – 1

Darzustellende Aspekte (unabhängig von den verwendeten Gliederungs- und Darstellungsmethoden):

- Funktionaler Aspekt
  - Daten: Struktur, Verwendung, Erzeugung, Speicherung, Übertragung, Veränderung
  - Funktionen: Ausgabe, Verarbeitung, Eingabe von Daten
  - Verhalten: Sichtbares dynamisches Systemverhalten, Zusammenspiel der Funktionen (untereinander und mit den Daten)
  - Fehler: Normalfall und Fehlerfälle

Inhalt einer Anforderungsspezifikation – 2

- Leistungsaspekt
  - Zeiten „...Reaktionszeit < 0,5 s...“
  - Mengen „...verwaltet bis zu 10 000 Kunden...“
  - Raten „...verarbeitet maximal 100 Transaktionen/s...“
  - Ressourcen „...benötigt 2 GByte Hauptspeicher...“
  - Genauigkeit „...berechnet auf vier Nachkommastellen genau...“

  Wo immer möglich:
  - ➤ messbare Angaben machen
  - ➤ Durchschnitts- und Extremwerte angeben

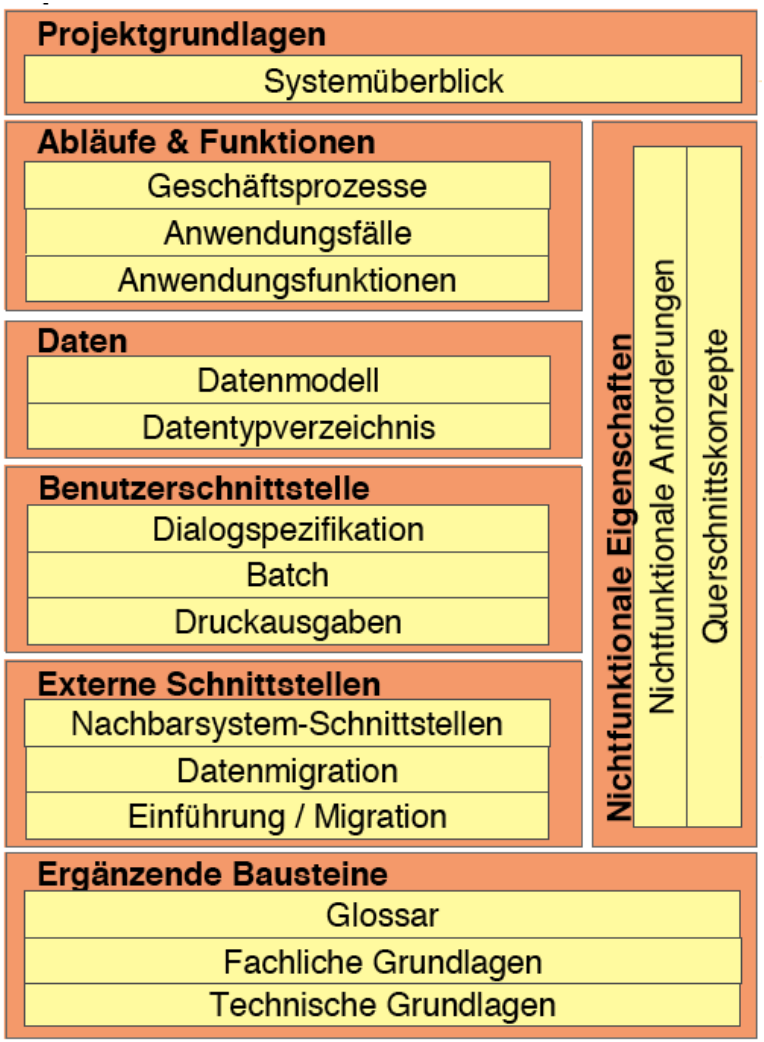
- Qualitätsaspekt
  - Geforderte besondere Qualitäten
    (zum Beispiel Zuverlässigkeit, Benutzbarkeit, Änderbarkeit, Portabilität,...)

Inhalt einer Anforderungsspezifikation – 3

- Randbedingungsaspekt
  - Einzuhaltende/zu verwendende Schnittstellen, Plattformen, Nachbarsysteme,...
  - Normen und Gesetze
  - Datenschutz, Datensicherung
  - Kulturelles: zum Beispiel internationale Verständlichkeit von Symbolen
  - Explizite Vorgaben des Auftraggebers
  - ...

Aufbau einer Anforderungsspezifikation

- Gliederung und Art der Darstellung abhängig von verwendeten Methoden und Sprachen
- Gliederung teilweise durch Normen und Richtlinien bestimmt
  - Normen, z.B. IEEE 830-1993
  - Firmeninterne Richtlinien
- Wahl der Gliederung hat großen Einfluss auf Verständlichkeit



Beispiel: Gliederungsrichtlinien der Firma sd&m[1)]

[1)] aus: Andreas Birk (2004). Anforderungsspezifikation in großen IT-Projekten. *Jahrestagung der GI-Fachgruppe Requirements Engineering*, Kaiserslautern.

Einfache Struktur für eine Anforderungsspezifikation

Teil I: Überblick

1 Anlass und Ziele
   Warum soll dieses System entwickelt werden?
2 Interesseneigner
   Benutzer, Auftraggeber und weitere Interesseneigner
3 Umfang und Kontext
   Systemumfang, relevantes Umfeld, Systemgrenze; ggf. Kontextdiagramm
4 Kernanforderungen
   Die wichtigsten Anforderungen im Überblick, ggf. gegliedert in Teilprobleme bzw. Subsysteme
5 Annahmen
   Annahmen, die erfüllt sein müssen, damit das System funktioniert

Einfache Struktur – 2

Teil II: Einzelanforderungen

Auflistung aller Anforderungen in geeigneter Gliederung

Anhänge

- Glossar

  Verzeichnis aller verwendeten Fachbegriffe mit Definitionen, Abkürzungen, Synonymen, etc.

- Verzeichnis referenzierter Dokumente

  Nachweis aller verwendeten, mitgeltenden Dokumente

Optionen für die Darstellung der Einzelanforderungen

- Falls System in Teilprobleme/Subsysteme gegliedert ist (vgl. Teil I): Einzelanforderungen analog strukturieren, z.B. mit hierarchischem Objektmodell oder hierarchischer Dezimalklassifikation

- Pro Teilproblem/Subsystem verschiedene Strukturen möglich, z.B.
  - Auflistung natürlichsprachlicher Einzelanforderungen
  - Auflistung von Benutzergeschichten
  - Modellbasierte Struktur, typisch
    - Anwendungsfälle
    - Daten/Klassen/Objektmodell
    - Verhaltensmodell
    - Schnittstellen (zu Benutzern und Nachbarsystemen)
    - Qualitätsanforderungen
    - Randbedingungen

Darstellungsformen

- Die möglichen Darstellungsformen unterscheiden sich konzeptionell vor allem in zwei Aspekten:
  - Art der Darstellung: konstruktiv oder deskriptiv
  - Formalitätsgrad der Darstellung

- Ein weiterer Freiheitsgrad besteht im Detaillierungsgrad der Darstellung

Deskriptive Darstellung – 1

Beispiele

- Darstellung mit Text in natürlicher Sprache:

  «Die Funktion Kontostand liefert den aktuellen Stand des Kontos für die eingegebene Kontonummer.»

- Darstellung in einer formalen Notation:
  Sqrt: Real → Real;
  Pre: $x \geq 0$;
  Post: $|Sqrt^2(x) - x| < \varepsilon \wedge \varepsilon \leq 10^{-16} \wedge \varepsilon \leq 10^{-6}x$.

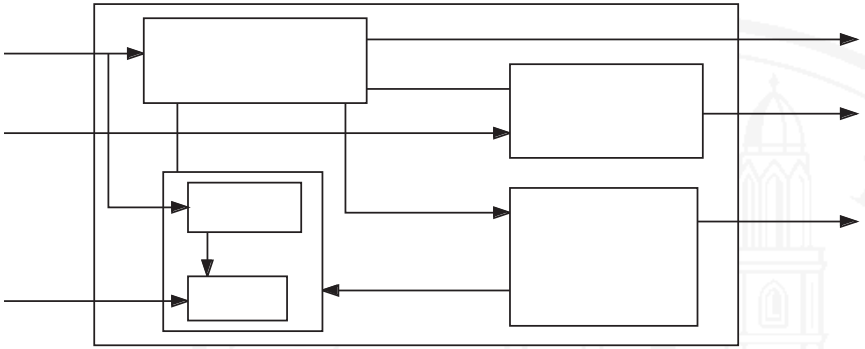
Deskriptive Darstellung – 2

- Vorteile:
  - \+ Nur äußeres Verhalten spezifiziert
  - \+ Lösungsneutral

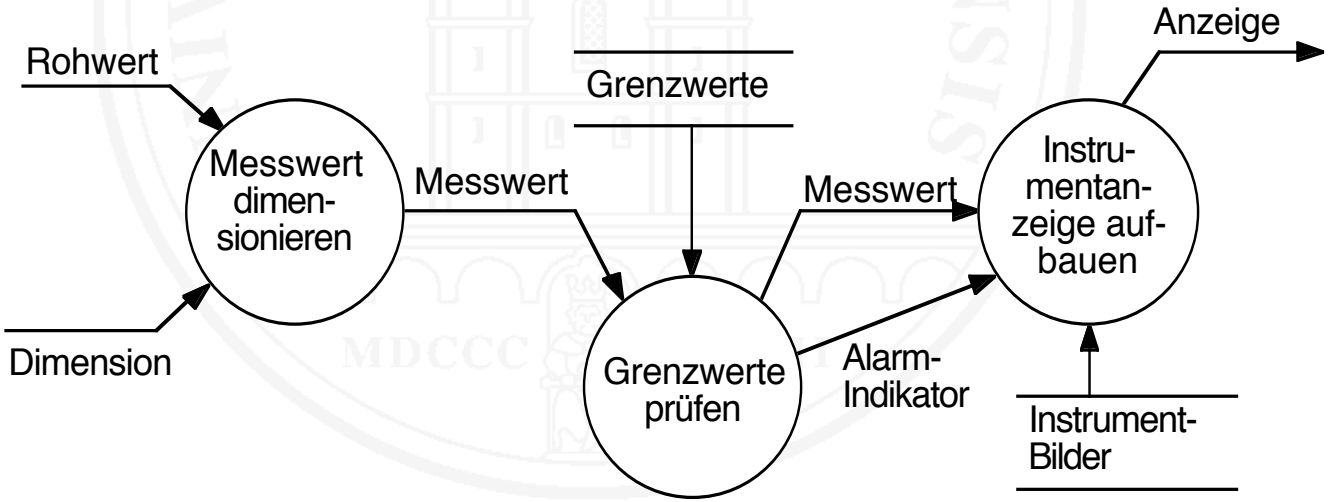
- Nachteile:
  - – Bei Verwendung von Text: umfangreich und wenig strukturiert; Zusammenhänge nicht erkennbar; fehlerträchtig und schwierig zu prüfen
  - – Bei Verwendung formaler Mitteln: sehr schwierig zu erstellen; Prüfung auf Adäquatheit oft fast unmöglich

⇒ Nur für die Darstellung der Anforderungen kleiner, überschaubarer Teilprobleme geeignet

Konstruktive Darstellung – 1



Beispiel



Konstruktive Darstellung – 2

Vorteile:

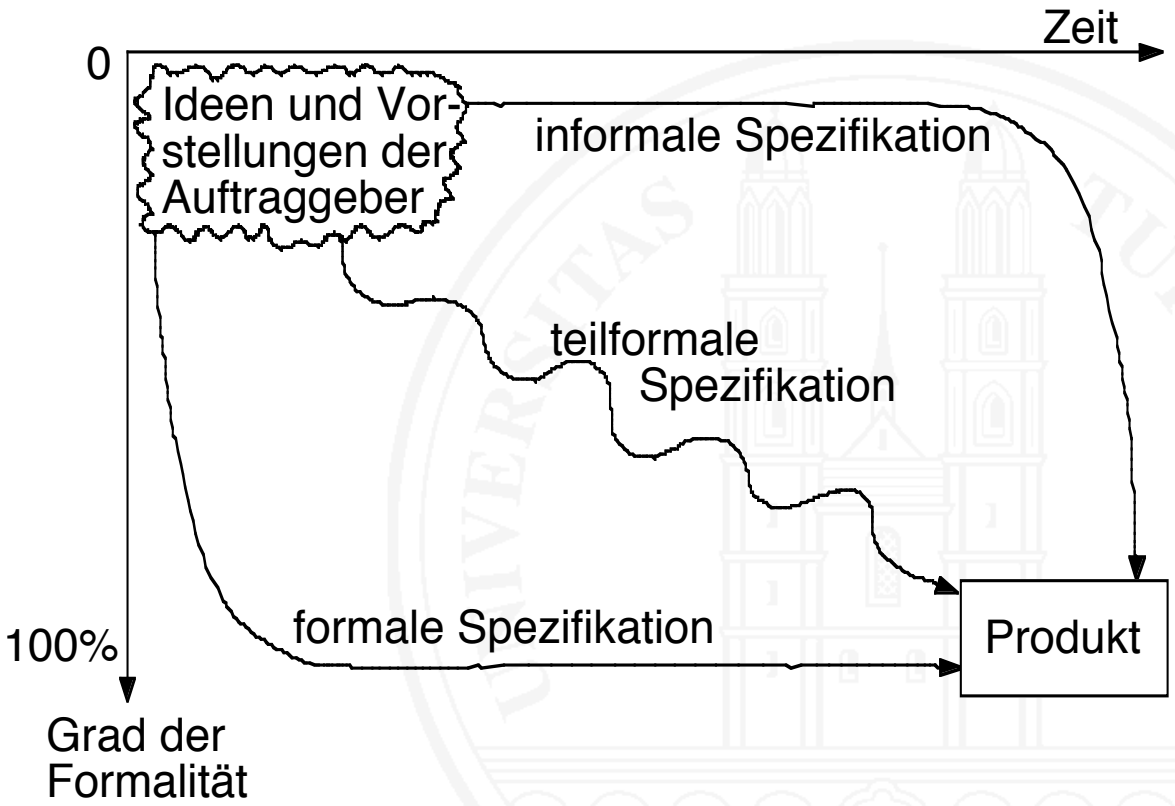
+ Anschauliches Modell der Problemstellung; leicht verstehbar und nachvollziehbar
+ Ermöglicht die Zerlegung der Gesamtaufgabe in kleinere, besser überschaubare Teilaufgaben
+ Kombination unterschiedlich stark formalisierter Teile möglich
+ Das Modell ist idealisierte Lösung: Tatsächliche Lösung oft analog strukturierbar

Nachteile:

– Modell ist idealisierte Lösung. Gefahr von
  - implementierungsorientierter Spezifikation
  - Implementierung suboptimaler Lösungen

⇨ Vor allem für die Modellierung von Anforderungen im Großen geeignet

Formalitätsgrad der Darstellung



- informal, in der Regel deskriptiv mit natürlicher Sprache
- formal, deskriptive und konstruktive Verfahren möglich
- teilformal mit konstruktiven, anschaulichen Modellen

Mini-Übung 4.2: Informale Spezifikation

«Der Bediener drückt eine Wahltaste und bezahlt den geforderten Betrag. Sobald die Summe der eingeworfenen Münzen den geforderten Betrag übersteigt, wird das Getränk zubereitet und ausgegeben. Ferner wird das Wechselgeld berechnet und ausgegeben. Der Bedienvorgang endet, wenn das Getränk entnommen wird und wenn die Bedienung für länger als 45s unterbrochen wird. Mit einer Annulliertaste kann der Bedienvorgang jederzeit abgebrochen werden. Bereits eingeworfenes Geld wird beim Drücken der Annulliertaste zurückgegeben. Nach dem Drücken einer Wahltaste kann entweder bezahlt oder eine andere Wahltaste gedrückt werden. Die zuletzt getätigte Auswahl gilt.»

a. Lesen Sie den neben-stehenden Text zügig durch. Fallen Ihnen irgendwelche Probleme auf?

b. Lesen Sie den Text nochmals langsam und sorgfältig und markieren Sie alle Problemstellen

Risikogerechte Detaillierung

Welche Variante ist besser:

A. «Die Teilnehmer-Eingabemaske enthält Felder für Name, Vorname, Geschlecht und Adresse des Teilnehmers. »

B. «Die Teilnehmer-Eingabemaske enthält Felder für Name, Vorname, Geschlecht und Adresse des Teilnehmers. Namen- und Vornamenfelder sind je maximal 32 Zeichen lang und obligatorisch. Das System verwendet Unicode als Zeichensatz. Für die Eingabe des Geschlechts enthält die Maske zwei Ankreuzfelder, beschriftet mit männlich und weiblich. Die Voreinstellung ist männlich, Ankreuzungen schließen sich gegenseitig aus, eine Ankreuzung ist erforderlich. ...»

➪ Der notwendige Detaillierungsgrad wird bestimmt durch Abwägung

- der Kosten
- des Risikos, unbrauchbare Systeme zu erhalten
- des Entscheidungsspielraums für die Entwickler

Präzision

- Sich präzise ausdrücken
  - Ausdrucksvielfalt bewusst reduzieren
  - Glossar verwenden
  - Definierte Subjekte: kein „man“, kein „es wird“

[Glinz 2013]

- Anforderungen prüfbar formulieren
  - Testbar
  - Quantifiziert
  - Vergleichbar
  - Formal

Tiefe

- Je präziser, desto umfangreicher (meistens)

➜ Lesbarkeit erhalten durch Strukturierung in die Tiefe:

«...

4.3 Teilnehmerverwaltung

4.3.1 Erfassen neuer Teilnehmer

4.3.1.1 Eingabemaske

4.3.1.2 Eingabebestätigung

4.3.2 Mutieren erfasster Teilnehmer

...»

Regeln für die Darstellung von Anforderungen

Anforderungsspezifikation fortlaufend inkrementell aufbauen

Anforderungen in kleinen Einheiten dokumentieren

Anforderungen strukturieren (zum Beispiel durch Kapitelgliederung)

Vom Ende her denken: Resultat → Funktion → Eingabe

Verhalten in unerwarteten Fällen spezifizieren

Je kritischer eine Anforderung, desto genauer dokumentieren

Wichtige Annahmen explizit dokumentieren

Redundanz vermeiden

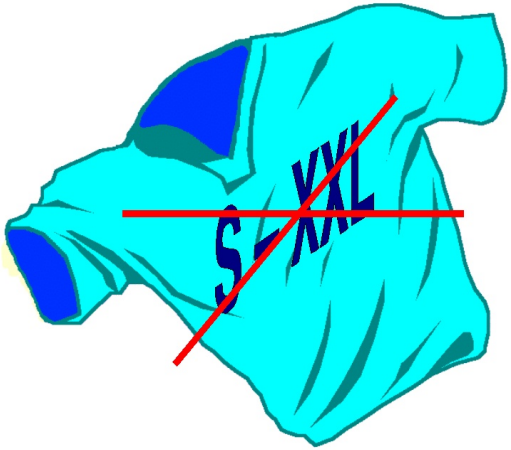
Sich an die Terminologie im Glossar halten

Zwei Hauptaufgaben

- Anforderungen spezifizieren (requirements specification)
  - Ermittlung (elicitation)
  - Analyse (analysis)
  - Dokumentation (documentation)
  - Prüfung (validation)
- Anforderungen verwalten (requirements management)
  - Freigabe (baselining, release management)
  - Änderung (modification, change management)
  - Verfolgung (traceability)

Der Spezifikationsprozess

Es gibt keinen idealen RE-Prozess.

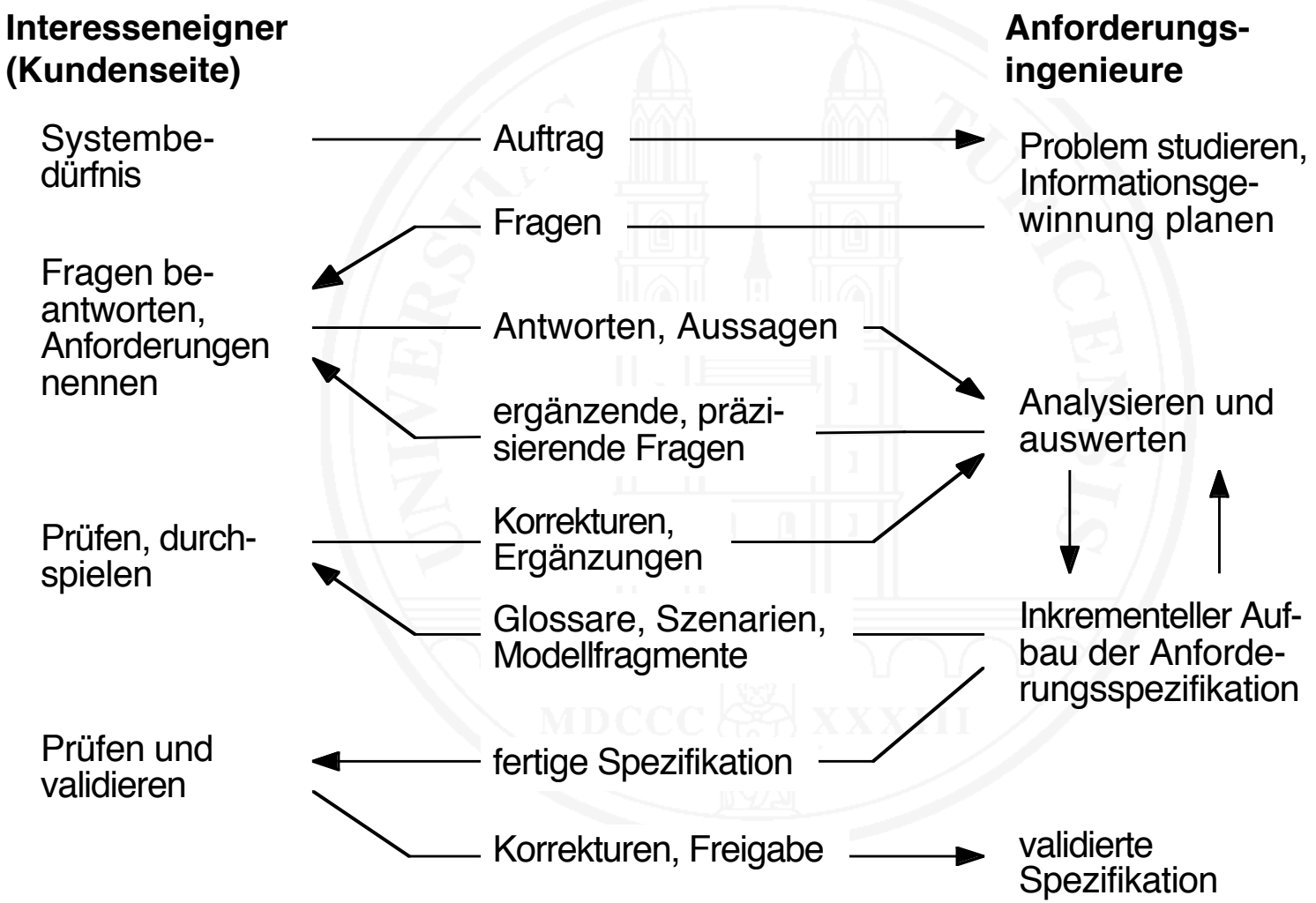


- Einflussfaktoren
  - Linear oder inkrementell?
  - Vertrag oder Zusammenarbeit?
  - Interesseneigner (Stakeholder) namentlich bekannt?
  - Kundenauftrag oder Entwicklung für den Markt?
  - Soll Standardsoftware zum Einsatz kommen?

➪ Den Prozess projektspezifisch zuschneiden

➪ Unter Verwendung der Grundprinzipien des RE

Typischer interaktiver Spezifikationsprozess



Analyse der Interesseneigner (Stakeholder)

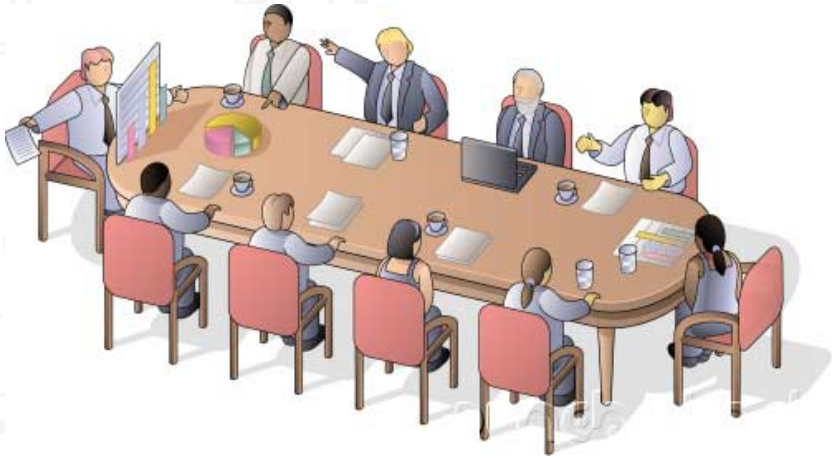
- Interesseneigner identifizieren
- In komplexen Fällen: ein Modell von Zielen, Abhängigkeiten und Intentionen erstellen

[Yu 1997]
[van Lamsweerde 2001]

- Interesseneigner klassifizieren
  - Kritische
  - Wichtige
  - Unwichtige

[Glinz und Wieringa 2007]

- In jeder Rolle konkrete Ansprechpersonen festlegen



Mini-Übung 4.3: Analyse der Interesseneigner

Die Leitung eines Universitätsinstituts will den Betrieb der Institutsbibliothek rationalisieren. Folgende Informationen sind bekannt:

Gewünscht wird ein softwarebasiertes System mit folgenden Fähigkeiten:

- Ausleihe, Rückgabe, Verlängern und Vormerken vollständig in Selbstbedienung durch die Bibliotheksbenutzer,
- Benutzerverwaltung und Katalogpflege durch die Bibliothekarin,
- Teilautomatisierter Mahnprozess (Schreiben der Mahnbriefe und Führen des Mahnstatus automatisiert; Versand und Inkasso manuell),
- Katalogrecherche, Verlängern und Vormerken lokal und über WWW.

Ferner ist ein Diebstahlsicherungssystem zu installieren, das verhindert, dass jemand die Bibliothek mit nicht ausgeliehenen Büchern verlässt.

Identifizieren Sie Interesseneigner-Rollen für dieses Projekt.

Systeme im Kontext situieren

Den Kontext abgrenzen

Randbedingungen identifizieren

- Physisch, gesetzlich
- Kulturell, Umwelt
- Entwurf & Implementierung
- Einbettung, Schnittstellen

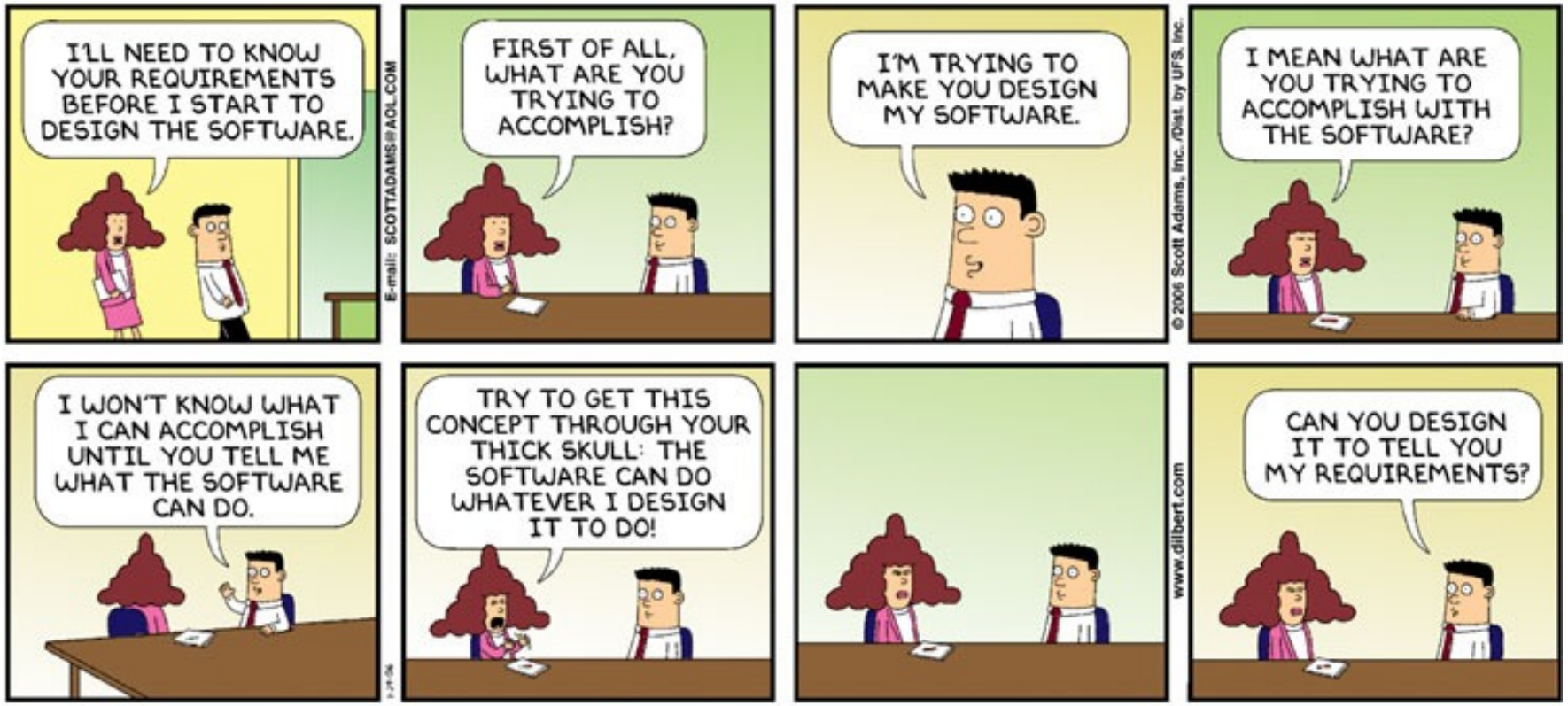
Annahmen über die Systemumgebung identifizieren und explizit machen

Phänomene der Realwelt angemessen abbilden

- auf die verlangten Systemfähigkeiten
- ... und umgekehrt

Photo © Universitätsklinikum Halle (Saale)

Ermittlung von Anforderungen



Ermittlung von Anforderungen

Anforderungsermittlung – Suchen, Erfassen und Konsolidieren von Anforderungen aus den verfügbaren Anforderungsquellen.

- Wünsche und Bedürfnisse der Interesseneigner erkennen
- Den Interesseneignern Möglichkeiten aufzeigen, wenn diese sie selbst nicht erkennen
- Wenn nötig, zunächst den IST-Zustand erheben
- Bei Produktentwicklungen Marktpotential klären
- Kann auch Rekonstruktion und Erschaffung von Anforderungen umfassen

Anforderungsquellen

- Interesseneigner (Stakeholder)
  - Wer hat wie mit dem zu erstellenden System zu tun?
  - Häufig nicht Individuen, sondern Rollen
  - Typische Rollen: Endbenutzer, Auftraggeber, Betreiber, Entwickler, Projektleitung, ...
  - Diverse Gesprächs- und Befragungstechniken
- Prozessabläufe
  - Wie wird heute gearbeitet?
  - Was ist gut und was soll anders werden?
  - Wer verwendet wo welche Daten?
- Unterlagen
  - Ausschreibungstexte, Visionsdokumente, ...

Ermittlungstechniken

- Vielzahl unterschiedlicher Techniken, insbesondere
  - Interesseneigner interviewen
  - Unterlagen auswerten
  - Anforderungsworkshops veranstalten
  - Interesseneigner beobachten
  - Prototypen bauen und erproben
- Situationsgerecht auswählen
- Gegebenenfalls Techniken kombinieren

[Zowghi und Coulin 2005]
[Gottesdiener 2002]
[Goguen und Linde 1993]
[Hickey und Davis 2003]

Ermittlungstechniken: Übersicht

Form	**Eignung für**			
	Wünsche ausdrücken	Möglichkeiten aufzeigen	IST-Zustand erheben	Marktpotenzial klären
Interviews	+	–	+	o
Beobachtung der Benutzer	o	–	+	o
Rollenspiele	+	o	o	–
Beispiele analysieren	o	–	+	–
Staffagen und Prototypen	o	+	–	o
Umfragen/Fragebogen	o	–	+	+
Gemeinsame Arbeitstagungen	+	o	o	–
Marktstudien	–	–	o	+
Problemmeldungsauswertung	+	–	–	o
Vergleich mit anderen	o	+	–	+

Typische Probleme

Diskrepanzen zwischen Interesseneignern

- Bedürfnisse und Erwartungen
- Terminologie

Interesseneigner

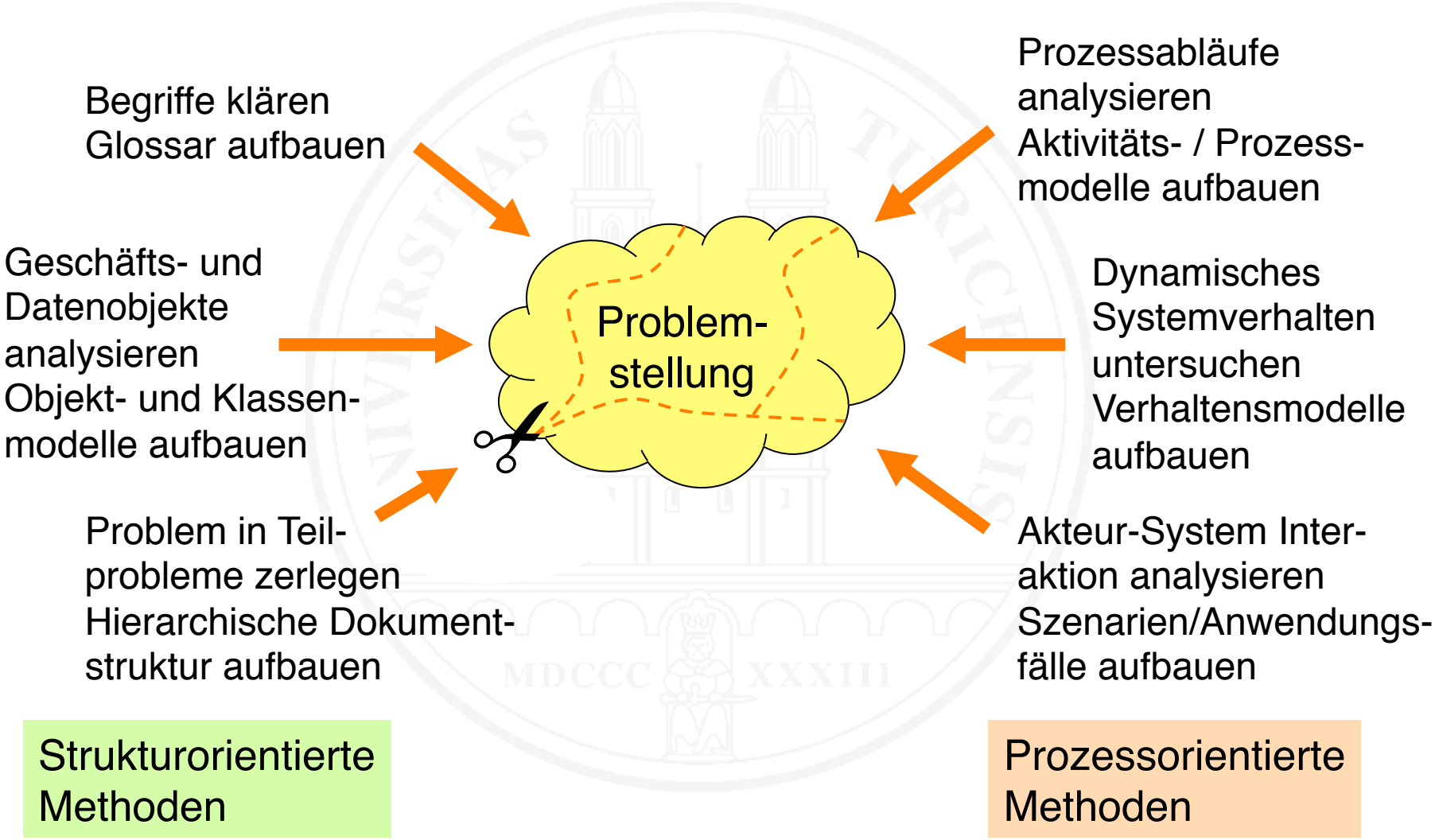
- wissen was sie wollen, aber können es nicht ausdrücken
- wissen nicht, was sie wollen
- haben verdeckte Ziele
- denken in Lösungen statt in Problemen

Attribute und Randbedingungen

- werden häufig nicht genannt („das ist doch selbstverständlich“)

➥ explizit ermitteln

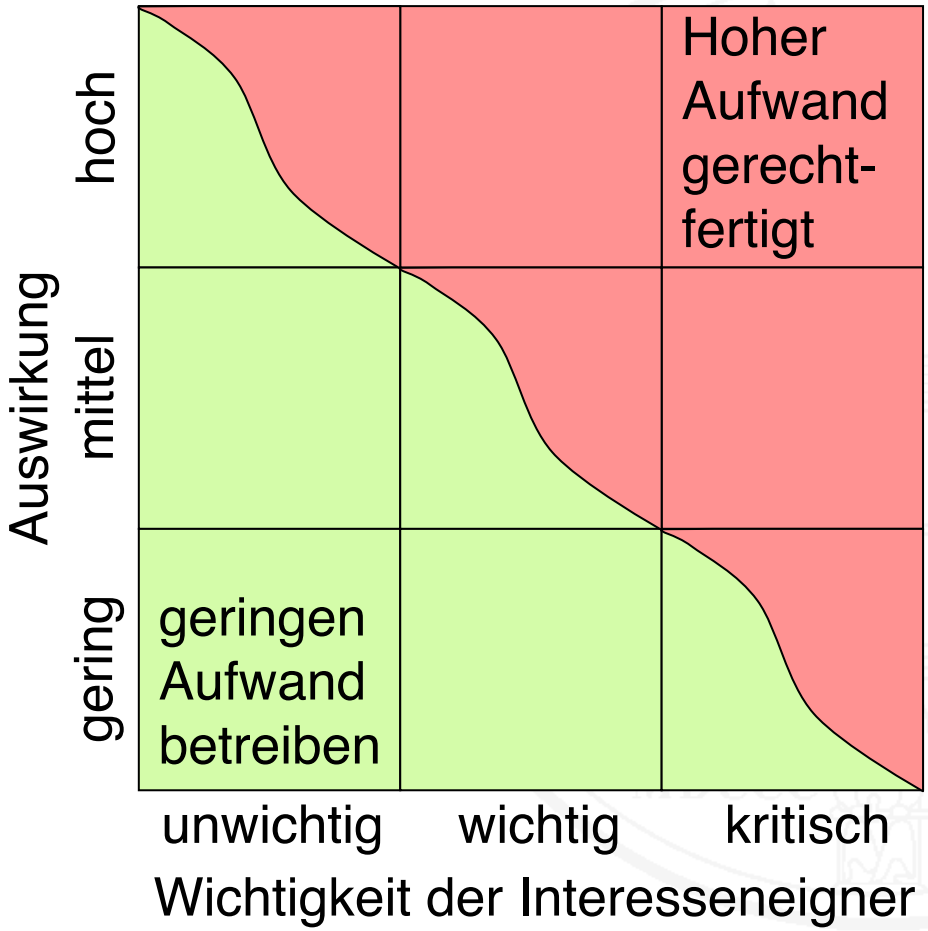
Anforderungsanalyse: Methodische Ansätze



Regeln

- Analysemethoden situationsgerecht auswählen
- Anforderungsspezifikation fortlaufend inkrementell aufbauen
  - Keine großen Materialsammlungen
  - Ermittlung, Analyse und Darstellung miteinander verzahnen
  - Rückkopplung ist wichtig
  - Von festem Grund ausgehen: vom Bekannten und Gesicherten zum Unbekannten und Offenen
- Anforderungen betreffen einen SOLL-Zustand
  - ➥ Den IST-Zustand nur analysieren, wenn dies notwendig ist

Risikoorientiert vorgehen



Risiken identifizieren

Aufwand abhängig von

- Auswirkung (wenn die Anforderung verfehlt wird)
- Wichtigkeit der Interessen-eigner (Stakeholder)

[Glinz 2008]
[Glinz und Wieringa 2007]

Weitere Risikofaktoren

- Gemeinsames (implizites) Verständnis
- Existenz von Referenzsystemen
- Länge des Rückkopplungszyklus
- Art der Auftraggeber-Auftragnehmer-Beziehung
- Notwendigkeit von Zertifizierung

Nicht vergessen:

Der *Aufwand* für das Requirements Engineering soll *umgekehrt proportional* zum *Risiko* sein, das man bereit ist, einzugehen.

Aufwand und Wert ausbalancieren

Was macht den Wert aus?

Kunden zahlen nicht für die Anforderungen

Wert entsteht indirekt:

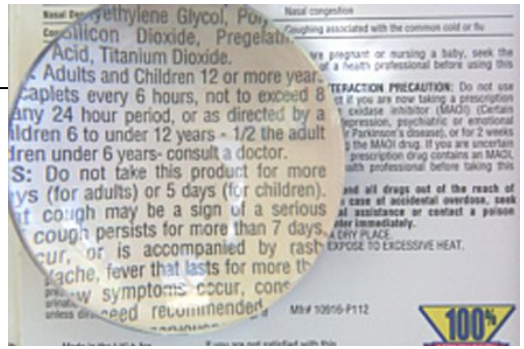
- Geringere Kosten für Nacharbeit
- Risikobeherrschung
- Höhere Kundenzufriedenheit

☞ Kein Prozess um des Prozesses willen

☞ Erst nach dem Wert fragen, dann handeln

☞ Aber: Keine Rechtfertigung für schlampige Anforderungen!

Anforderungen validieren



Entsprechen die Anforderungen den Wünschen und Bedürfnissen der Interesseneigner?

- Fehler, Lücken, Unklarheiten, Mehrdeutigkeiten, etc. finden und beheben
- Abweichungen von der geforderten Qualität der Spezifikation feststellen und beheben
- Konflikte zwischen den Wünschen / Forderungen verschiedener Interesseneigner erkennen und lösen

Kurze Rückkopplungszyklen etablieren

Mit Abnahmetestfällen verdeutlichen / eindeutig machen

Techniken für die Validierung

Review

- 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

Prüf- und Analysemittel in Werkzeugen

- Einsatz bei werkzeuggestützter Erstellung der Spezifikation
- Auffinden von Lücken und Widersprüchen

Techniken für die Validierung – 2

Simulation/Animation

- Untersuchung des dynamischen Systemverhaltens
- Spezifikation wird durch Simulator ausgeführt und/oder durch Animator visualisiert

Prototyp

- Beurteilung der Adäquatheit / praktischen Brauchbarkeit in der geplanten Einsatzumgebung
- Modell für das zu schaffende Produkt
- Mächtigstes (aber auch aufwendigstes) Mittel zur Validierung

Formale Verifikation / Model Checking

- Formaler Beweis kritischer Eigenschaften

Innovation

Bild: © Apple

„Dem Kunden genau das liefern, was er wünscht.“

Falsch

„Wir wissen schon, was gut für den Kunden ist.“

Auch falsch

Wie entsteht Innovation?

- Den Interesseneignern innovative Lösungen vorschlagen
- Kreativität der Interesseneigner anregen
  - Zukunftsszenarien entwerfen und durchspielen
  - Alle Beschränkungen fallen lassen
  - Metaphern suchen und explorieren

➜ Anforderungen lassen sich nicht einfach „erheben“

[Maiden, Gitzikis und Robertson 2004]
[Maiden und Robertson 2005]

Ausgewählte Spezifikationsmethoden und -sprachen

Informal

- Spezifikation mit natürlicher Sprache

Formal

- Algebraische Spezifikation

Teilformal

- Objektorientierte Spezifikation
- Spezifikation mit Anwendungsfällen
- Verhaltensspezifikation mit Automaten

Details dazu: siehe Vorlesung Informatik IIa: Modellierung

Anforderungsspezifikation mit natürlicher Sprache

Das System soll ...

Die älteste

...und am meisten verwendete Spezifikationssprache

- Keine Extra-Ausbildung erforderlich
- Sehr ausdrucksmächtig
- Manches nicht anders ausdrückbar

Aber nicht die beste Spezifikationssprache

- Mehrdeutig
- Unscharf
- Fehlerträchtig
- Prüfung nur durch sorgfältiges Lesen

Weniger geeignet als alleiniges Ausdrucksmittel

Moses hält die zehn Gebote in der Hand: geschrieben in natürlicher Sprache. Statue von Michelangelo (San Pietro in Vincoli, Rom)

Regeln für natürlichsprachliche Anforderungen

[Rupp et al. 2009]

- Sätze mit vollständiger Satzstruktur zum jeweiligen Verb bilden
- Anforderung im Aktiv formulieren mit definiertem Subjekt
- Nur Begriffe verwenden, die im Glossar definiert sind
- Nomen mit unspezifischer Bedeutung („die Daten“, „der Kunde“, „die Anzeige“,...) hinterfragen und durch spezifische Nomen ersetzen oder mit präzisierenden Zusätzen ergänzen
- All-Quantifizierungen („Jeder“, „Immer“, „Nie“,...) kritisch hinterfragen
- Nominalisierungen hinterfragen; sie können unvollständig spezifizierte Prozesse verbergen
- Anforderungen in Hauptsätzen formulieren. Nebensätze nur zur Vervollständigung (wann, mit wem, unter welchen Bedingungen, ...)
- Pro Einzelanforderung ein Satz

Satzschablonen

[Rupp et al. 2009
ISO/IEC/IEEE 29148:2011
Cohn 2004]

- Satzschablonen helfen beim Formulieren wohlgeformter Einzelanforderungen.
- Typische Schablone für natürlichsprachliche Anforderungen

  [<Bedingung>] <Subjekt> <Aktionsverb> <Objekte> [<Restriktionen>]

Wenn die erkannte Karte gültig ist, soll das System das Kommando „Entriegeln für eine Drehung“ an das Drehkreuz senden, und zwar innerhalb von 100 ms.

- Typische Schablone für Benutzergeschichten

  Als <Rolle> will ich <meine Anforderung> [so dass < Nutzen für>]

Als Skifahrer will ich die Zugangskontrolle passieren, so dass ich den Sessellift benutzen kann, ohne ein Ticket zeigen oder scannen zu müssen.

Mini-Übung 4.4: Natürlichsprachliche Anforderungen

Beurteilen Sie die sprachliche Qualität der folgenden mit natürlicher Sprache formulierten Anforderungen an ein Bibliotheksverwaltungssystem:

«Einmal täglich muss kontrolliert werden, welche ausgeliehenen Bücher überfällig sind.»

«Das System soll Mahnungen verschicken.»

«Nach erfolgreicher Authentifizierung des Ausleihers soll die Ausleihe von einem oder mehreren Büchern möglich sein.»

«Bei jeder Ausleihe sollen die Daten auf der Anzeige erscheinen.»

«Damit die Ausleihe notfalls auch manuell erfolgen kann, soll die Benutzerkarte den Benutzernamen als Text und in maschinenlesbarer Form enthalten.»

Algebraische Spezifikation

- Deskriptive, formale Methode
- Primär für die formale Spezifikation komplexer Datentypen
- Syntax durch Signaturen (Definitions- und Wertebereiche) der Operationen
- Semantik durch Axiome (Ausdrücke, die immer wahr sein müssen)

+ Immer eindeutig (da Semantik formal definiert)
+ Widerspruchsfreiheit formal prüfbar
+ Erfüllung wichtiger Eigenschaften beweisbar
+ Formale Verifikation von Programmen möglich

– Erstellung sehr aufwendig
– Prüfung/Nachweis der Vollständigkeit wird nicht einfacher
– Schwer lesbar → Prüfung auf Adäquatheit schwierig

Algebraische Spezifikation – 2: Beispiel

Spezifikation eines Kellers (Stack)

Sei bool der Datentyp mit dem Wertebereich {false, true} und der Booleschen Algebra als Operationen. Sei ferner elem der Datentyp für die Datenelemente, die im spezifizierten Keller zu speichern sind.

```
TYPE Stack
FUNCTIONS
new:    ()              →  Stack;   -- neuen (leeren) Keller anlegen
push:   (Stack, elem)   →  Stack;   -- Element hinzufügen
pop:    Stack           →  Stack;   -- zuletzt hinzugefügtes Element entfernen
top:    Stack           →  elem;    -- liefert zuletzt hinzugefügtes Element
empty:  Stack           →  bool;    -- wahr, wenn Keller kein Element enthält
full:   Stack           →  bool;    -- wahr, wenn Keller voll ist
```

Algebraische Spezifikation – 3: Beispiel (Fortsetzung)

AXIOMS

∀ s ∈ Stack, e ∈ elem

(1)	¬ full(s) → pop(push(s,e)) = s	--	Pop hebt den Effekt von Push auf
(2)	¬ full(s) → top(push(s,e)) = e	--	Top liefert das zuletzt gespeicherte Element
(3)	empty(new) = true	--	ein neuer Keller ist leer
(4)	¬ full(s) → empty(push(s,e)) = false	--	nach Push ist ein Keller nicht mehr leer
(5)	full(new) = false	--	ein neuer Keller ist nicht voll
(6)	¬ emtpy(s) → full(pop(s)) = false	--	nach Pop ist ein Keller niemals voll

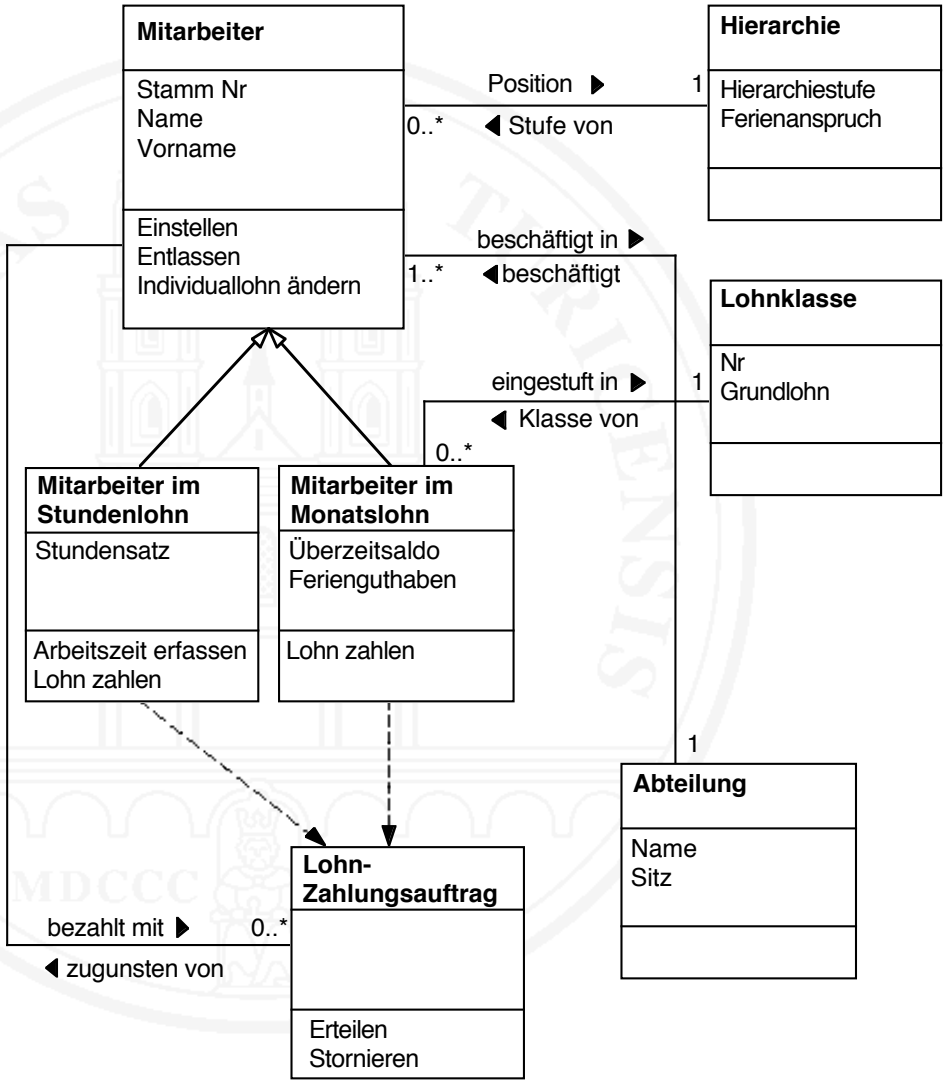
Objektorientierte Spezifikation

- Grundidee:
  - Systembeschreibung durch Objekte
  - Jedes Objekt beschreibt in sich geschlossenen Teil der Daten, der Funktionalität und des Verhaltens eines Systems
  - Abbildung eines Ausschnitts der Realität auf Objekte/Klassen
  - Objekte kapseln logisch zusammengehörige Daten, Operationen und Verhaltensweisen
  - Gleichartige Objekte werden als Klassen modelliert.
- Konstruktives, teilformales Verfahren
- Anfänglich eine Vielzahl verschiedener Ansätze, z. B. Booch, Coad und Yourdon, Jacobson, Rumbaugh, Wirfs-Brock
- Industriestandard heute: UML (Unified Modeling Language)

Objektorientierte Spezifikation – 2: Klassenmodell

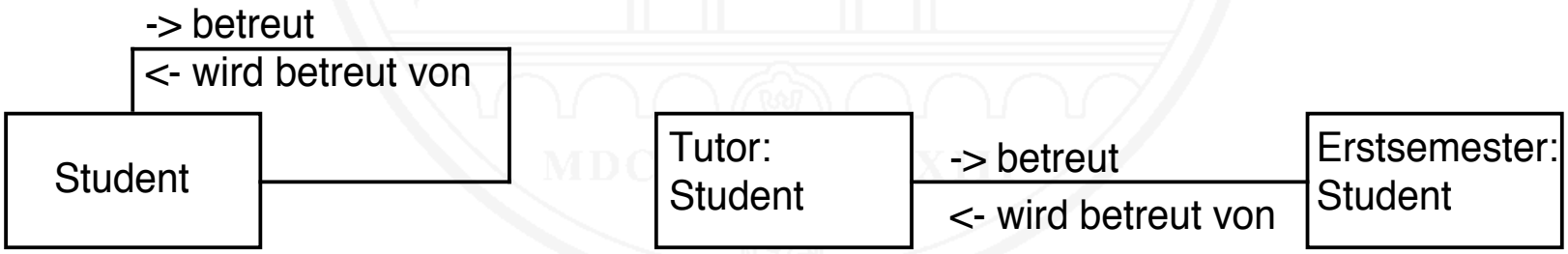
Beispiel eines Klassenmodells...

...in der Notation der Unified Modeling Language (UML)



Objektorientierte Spezifikation – 3: Objektmodelle

- Alternative oder Ergänzung zu Klassenmodellen
- Modellierung abstrakter Objekte
- Stehen als Muster bzw. Repräsentanten für konkrete Objekte
- In UML nur als Ergänzung zu Klassenmodellen verwendet
- Objektmodelle anstelle von Klassenmodellen: große Vorteile, wenn
  - verschiedene Objekte der gleichen Klasse zu modellieren sind
  - ein Modell hierarchisch in Komponenten zerlegt werden soll



Klassenmodell Objektmodell

Objektorientierte Spezifikation – 4: Vor- und Nachteile

+ Gut geeignet zur Beschreibung der Systemstruktur
+ Unterstützt Lokalität von Daten und Einkapselung von Eigenschaften
+ Erlaubt strukturähnliche Implementierungen
+ Systemdekomposition möglich

– Funktionalität aus Benutzersicht wird nicht modelliert
– Verhaltensmodelle nur schwach integriert
– Dekomposition häufig nicht unterstützt

Spezifikation mit Anwendungsfällen – 1

- Klassenmodelle modellieren den Systemkontext und die Interaktionen zwischen System und Umgebung nicht
- Diese sind aber wichtig. Darum
- ➪ Ergänzung durch Anwendungsfallmodelle

Anwendungsfall (use case) – Eine durch genau einen Akteur angestoßene Folge von Systemereignissen, welche für den Akteur ein Ergebnis produziert und an welchem weitere Akteure teilnehmen können.

- Anwendungsfälle modellieren die Interaktion zwischen systemexternen Akteuren und dem System
- Pro Interaktionssequenz ein Anwendungsfall
- Teilformales, konstruktives Verfahren

Spezifikation mit Anwendungsfällen – 2

- Modellierung der einzelnen Anwendungsfälle
  - informal durch Text, z.T. formatiert durch Schlüsselworte

> Buch ausleihen
> 1. Ausweiskarte der Benutzerin lesen und Angaben überprüfen
> 2. Signatur eines Buchs lesen und zugehörigen Katalogeintrag ermitteln
> 3. Ausleihe registrieren und Diebstahlsicherungsetikett deaktivieren
> 4. ...

  - teilformal, z.B. mit Zustandsautomaten

\+ Modellierung aus Benutzersicht: leicht verstehbar und überprüfbar

\+ Hilft bei der Abgrenzung zwischen System und Kontext

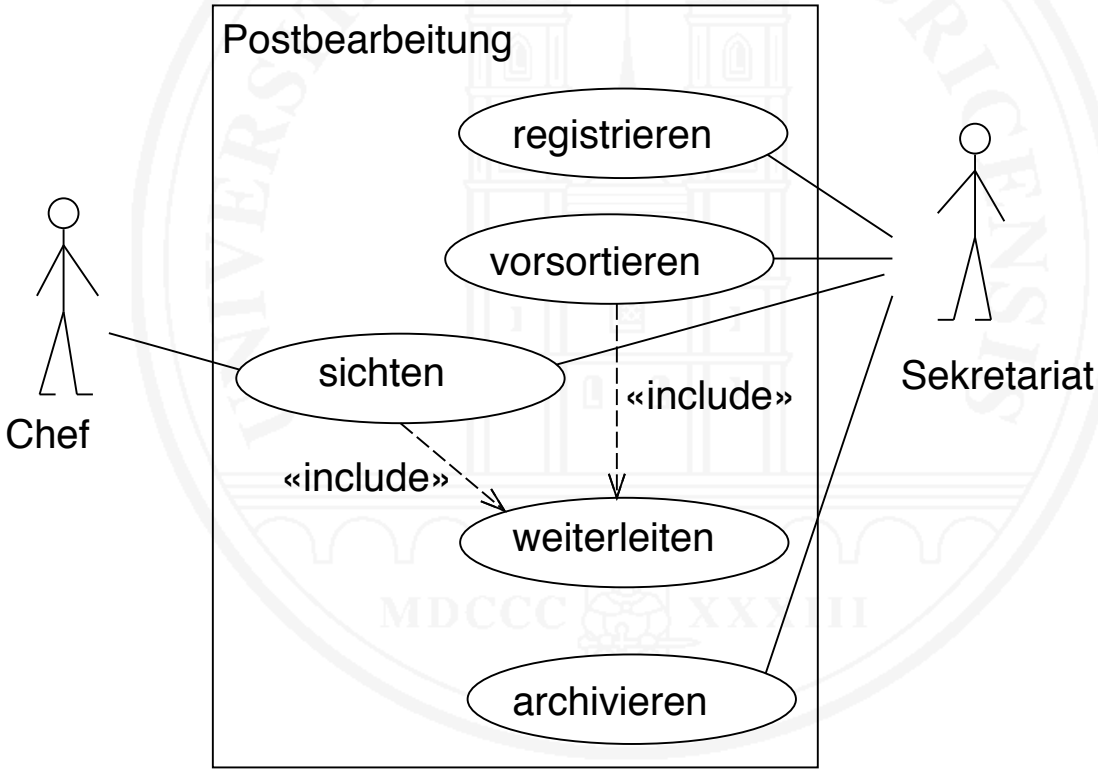
\+ Dekomposition möglich

– Zusammenhänge / Abhängigkeiten zwischen Szenarien nicht modelliert

– Statische Struktur nicht modelliert

Spezifikation mit Anwendungsfällen – 3

- Modellierung des Systemkontextes und der Menge aller Anwendungsfälle: Anwendungsfalldiagramm

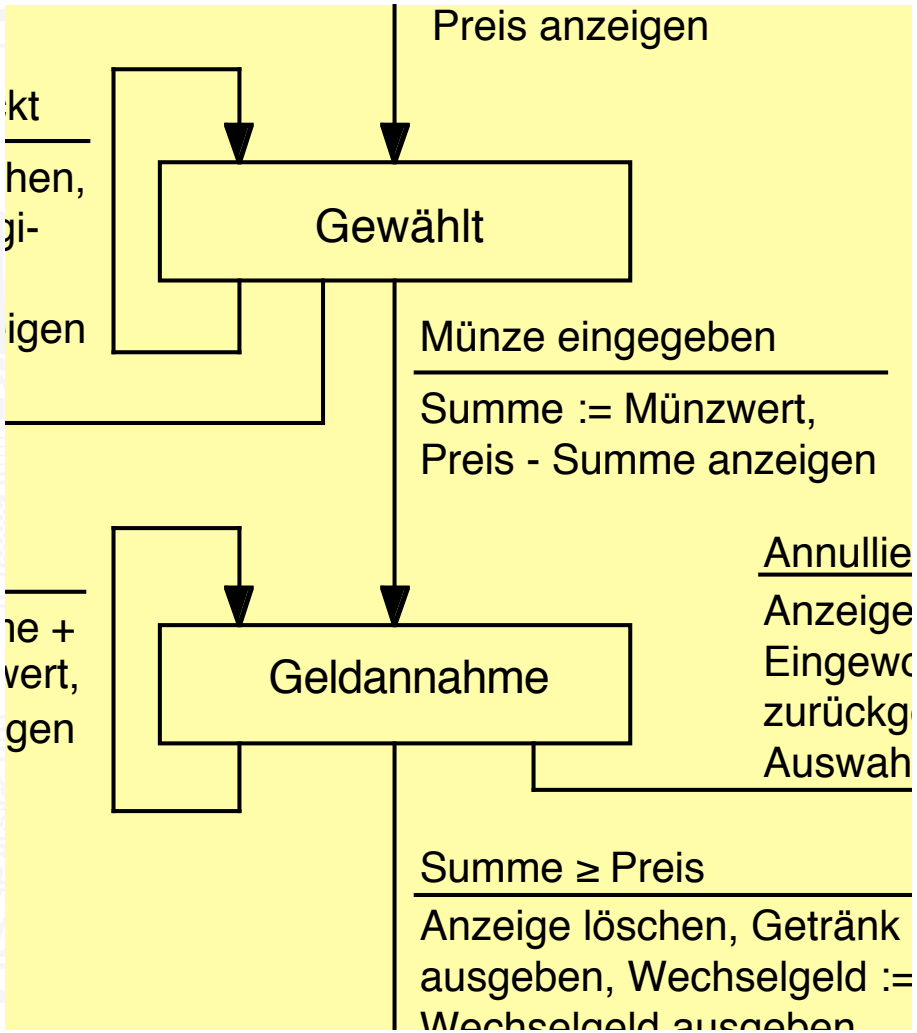


Verhaltensspezifikation mit Automaten

- Modellierung des zeitlich-dynamische Systemverhaltens
- Basis: Zustandsautomaten
- Teilformales, konstruktives Verfahren

+ Leicht nachvollziehbar und simulierbar
+ Dekomposition möglich

– Aktionen meist nur genant, aber nicht modelliert
– Statische Struktur nicht modelliert



Spezifikation von Attributen und Randbedingungen

- Zu den Inhalten siehe oben („Inhalt einer Anforderungsspezifikation“)
- In der Regel mit natürlicher Sprache ausgedrückt
- Attribute charakterisieren
  - das ganze System
    ➥ als globale Attribute dokumentieren
  - eine einzelne funktionale Anforderung
    ➥ zusammen mit dieser dokumentieren
  - querschneidend eine Menge von funktionalen Anforderungen
    ➥ separat dokumentieren mit Querverweisen
- Randbedingungen werden in der Anforderungsspezifikation separat als eigene Anforderungsart dokumentiert

Anforderungsmanagement

- Anforderungen geordnet ablegen und wiederfinden
  - Anforderungen einzeln identifizierbar
  - Anforderungen attributiert
  - In der Regel mit Werkzeughilfe
- Anforderungen priorisieren
- Anforderungen geordnet ändern
- Anforderungen verfolgen

Anforderungen priorisieren

Nicht alle Anforderungen sind gleich wichtig.

Typisch drei Kategorien

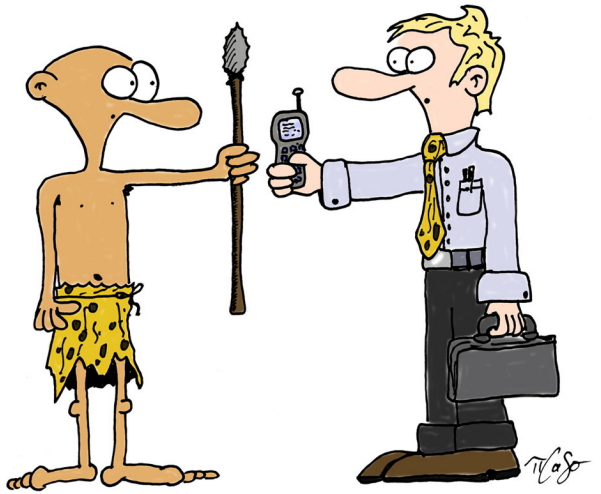
- Kritisch
- Wichtig
- Nebensächlich

Insbesondere wichtig bei

- Gegebenen Preisobergrenzen
- Verwendung von Standardsoftware
- Aufwand-Wert-Abwägungen
- Inkrementeller Entwicklung
- Releaseplanung (in bestehenden Systemen)

Priorisierung erfolgt durch Interesseneigner!

Die Evolution von Anforderungen im Griff haben



Die Welt verändert sich.

Und mit ihr die Anforderungen.

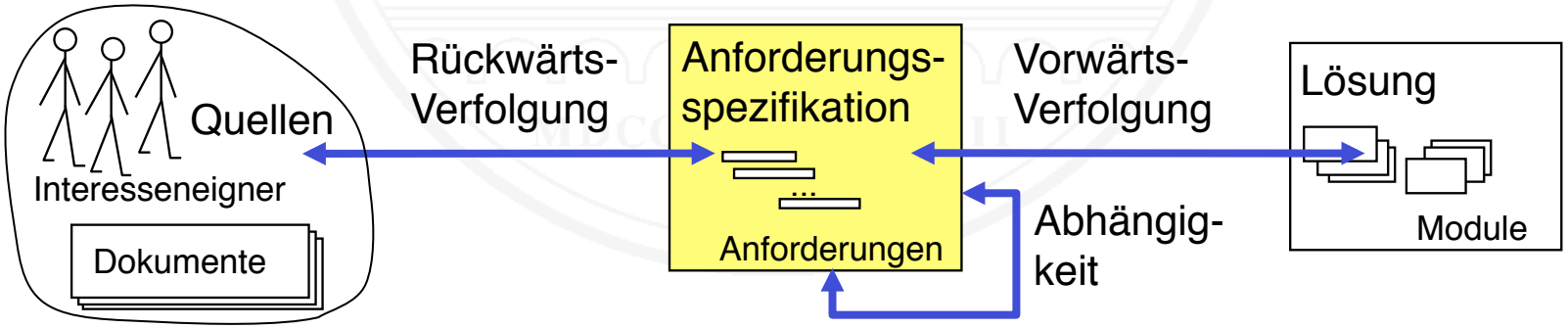
Das Problem:
Anforderungen stabil halten ...
... und gleichzeitig ihre Änderung zulassen

Mögliche Ansätze

- Sehr kurze Entwicklungszyklen (1-6 Wochen)
- Explizites Anforderungsmanagement

Explizites Requirements Management

- Konfigurationsmanagement für Anforderungen
  - Anforderungen einzeln identifizierbar
  - Geordneter Änderungsprozess
  - Klare Zuständigkeiten und Verantwortlichkeiten
- Verfolgbarkeit (traceability) [Gotel und Finkelstein 1994, Dick 2005]
  - Rückwärts: Wo kommt welche Anforderung her?
  - Vorwärts: Wo ist welche Anforderung entworfen bzw. implementiert?
  - Wie hängen Anforderungen voneinander ab?



Literatur

M. Cohn (2004). *User Stories Applied: For Agile Software Development*. Boston: Addison-Wesley.

J. Dick (2005). Design Traceability. *IEEE Software* **22**, 6 (Nov./Dec. 2005). 14-16.

M. Glinz (2005). Rethinking the Notion of Non-Functional Requirements. *Proceedings of the Third World Congress for Software Quality (3WCSQ 2005)*, München, Vol. II, 55-64.

M. Glinz (2007). On Non-Functional Requirements. *Proceedings of the 15th IEEE International Requirements Engineering Conference*, Delhi, India. 21-26.

M. Glinz, R. Wieringa (2007). Stakeholders in Requirements Engineering. *IEEE Software* **24**, 2. 18-20.

M. Glinz (2008). A Risk-Based, Value-Oriented Approach to Quality Requirements. *IEEE Software* **25**,2. 34-41.

M. Glinz (2013). *A Glossary of Requirements Engineering Terminology*, Version 1.5. International Requirements Engineering Board (IREB).
Verfügbar auf http://www.ireb.org (CPRE Glossary klicken)

E. Gottesdiener (2002). *Requirements by Collaboration: Workshops for Defining Needs*. Boston: Addison-Wesley.

J. Goguen and C. Linde (1993). Techniques for Requirements Elicitation. *Proceedings 1st IEEE International Symposium on Requirements Engineering (RE'93)*, San Diego, USA. 152-164.

O. Gotel, A. Finkelstein (1994) An Analysis of the Requirements Traceability Problem, *Proceedings 1st International Conference on Requirements Engineering,* Colorado Springs. 94-101.

A.M. Hickey and A.M. Davis (2003). Elicitation Technique Selection: How Do Experts Do It? *Proceedings 11th IEEE International Requirements Engineering Conference (RE'03)*, Monterey Bay, USA. 169-178.

Literatur – 2

ISO/IEC (2011). *Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models*. ISO/IEC Standard 25010:2011.

M. Jackson (2005). Problem Frames and Software Engineering. *Information and Software Technology* **47**, 14. 903-912.

A. van Lamsweerde (2001). Goal-Oriented Requirements Engineering: A Guided Tour. *Proceedings 5th IEEE International Symposium on Requirements Engineering (RE'01)*, Toronto, Canada. 249-261.

L. Macaulay (1993.) Requirements Capture as a Cooperative Activity. *Proceedings 1st IEEE International Symposium on Requirements Engineering*, San Diego. 174–181.

N. Maiden, A. Gizikis, S. Robertson (2004). Provoking Creativity: Imagine What Your Requirements Could Be Like. *IEEE Software* **21**, 5 (Sept./Oct. 2005). 68-75.

N. Maiden and S. Robertson (2005). Integrating Creativity into Requirements Processes: Experiences with an Air Traffic Management System. Proceedings *13th IEEE International Requirements Engineering Conference (RE'05)*, Paris, France. 105-114.

B. Nuseibeh, J. Kramer, A. Finkelstein (2003). ViewPoints: Meaningful Relationships are Difficult! *Proceedings 25th International Conference on Software Engineering (ICSE'03)*, Portland, Oregon. 676-681.

C. Rupp, die SOPHISTen (2009). *Requirements-Engineering und -Management*. 5. Auflage. München: Hanser.

E. Yu (1997). Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. *Proceedings 3rd IEEE International Symposium on Requirements Engineering (RE'97)*. 226-235.

Literatur – 3

P. Zave, M. Jackson (1997). Four Dark Corners of RequirementsEngineering. *ACM Transactions on Software Methodology* **6**, 1.1-30.

D. Zowghi, C. Coulin (2005). Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In Aurum, A., C. Wohlin. *Engineering and Managing Software Requirements*. Springer. 19-46.

Ferner siehe auch Literaturverweise im Kapitel 7 des Skripts.