

Martin Glinz Harald Gall

Software Engineering

Kapitel 4

Spezifikation von Anforderungen



Universität Zürich
Institut für Informatik

4.1 Grundlagen

4.2 Der Spezifikationsprozess

4.3 Anforderungsgewinnung und -analyse

4.4 Dokumentation von Anforderungen

4.5 Spezifikationsmethoden und -sprachen

4.6 Prüfen von Anforderungen

4.7 Verwalten von Anforderungen

Definitionen und grundlegende Begriffe – 1

Anforderung (requirement) – 1. Eine **Bedingung oder Fähigkeit**, die von einer **Person** zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird.

2. Eine **Bedingung oder Fähigkeit**, die eine **Software** erfüllen oder besitzen muss, um einen Vertrag, eine Norm oder ein anderes, formell bestimmtes Dokument zu erfüllen. (IEEE 610.12-1990)

Anforderungsspezifikation (requirements specification) – Die **Zusammenstellung aller Anforderungen** an eine Software.

Synonyme: Anforderungsdokument, Software Requirements Specification.

Definitionen und grundlegende Begriffe – 2

„Die Spezifikation“: Im Alltag nicht immer eindeutig:

- Dokument oder Prozess
- Anforderungs-, Lösungs- oder Produktspezifikation

Pflichtenheft → verschiedene Begriffe:

- Synonym zu Anforderungsspezifikation
- Spezifikation + Überblick über Lösung
- Spezifikation + Elemente der Projektabwicklung
- „Pflichtenheft“ mit Vorsicht verwenden

Requirements Engineering

Requirements Engineering (Anforderungstechnik) –

1. Das **systematische**, **disziplinierte** und **quantitativ erfassbare** Vorgehen beim **Spezifizieren**, d.h. Erfassen, Beschreiben und Prüfen von Anforderungen an Software.
2. **Verstehen** und **Beschreiben**, was die Kunden **wünschen** oder **brauchen**.
3. Spezifikation und Verwaltung von Anforderungen mit dem Ziel, das **Risiko zu minimieren**, dass Software entwickelt wird, welche den Kunden nicht nützt oder gefällt.

Anforderungsspezifikation als Risikominimierung

„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.

Anforderungsspezifikation – Warum

○ Kosten senken

- Geringere Herstellungskosten (Senken der Fehlerkosten!)
- Geringere Pflegekosten

Mehr verdienen!

○ Risiken verkleinern

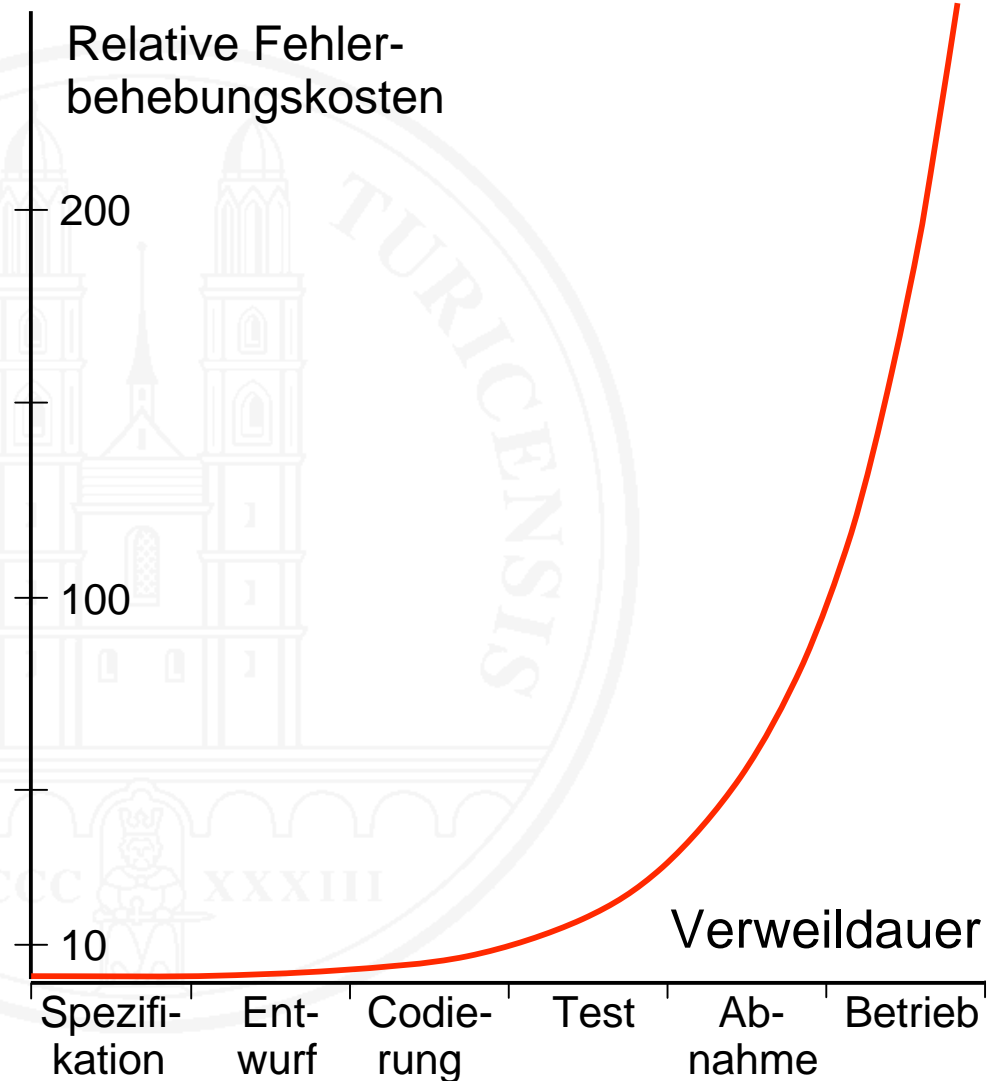
- Kundenerwartungen besser erfüllen
- Zuverlässigere Prognosen für Termine und Kosten

Zufriedenere Kunden!

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

Anforderungsspezifikation – Warum (2)

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



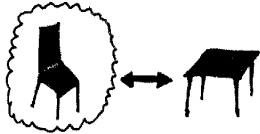
Wirtschaftlichkeit der Anforderungsspezifikation



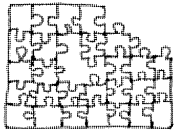
⇒ **Wenig** Anforderungsfehler machen

⇒ Möglichst viele der dennoch gemachten Fehler **früh** finden

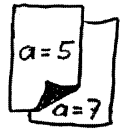
Merkmale einer guten Anforderungsspezifikation



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

$$\bigwedge_{n \in \mathbb{N}} \bigwedge_{m_i \text{ ASien}} \sum_{i=1}^n |m_i| \geq P \wedge$$

$$\sum_{i=1}^{n-1} |m_i| < P \leftrightarrow W(m_1, \dots, m_n)$$

$$\wedge \neg W(m_1, \dots, m_{n-1})$$

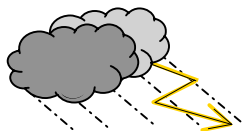
Verständlich – für alle Beteiligten, Kunden wie Informa-
tiker



Eindeutig – vermeidet Fehler durch Fehlinterpretationen

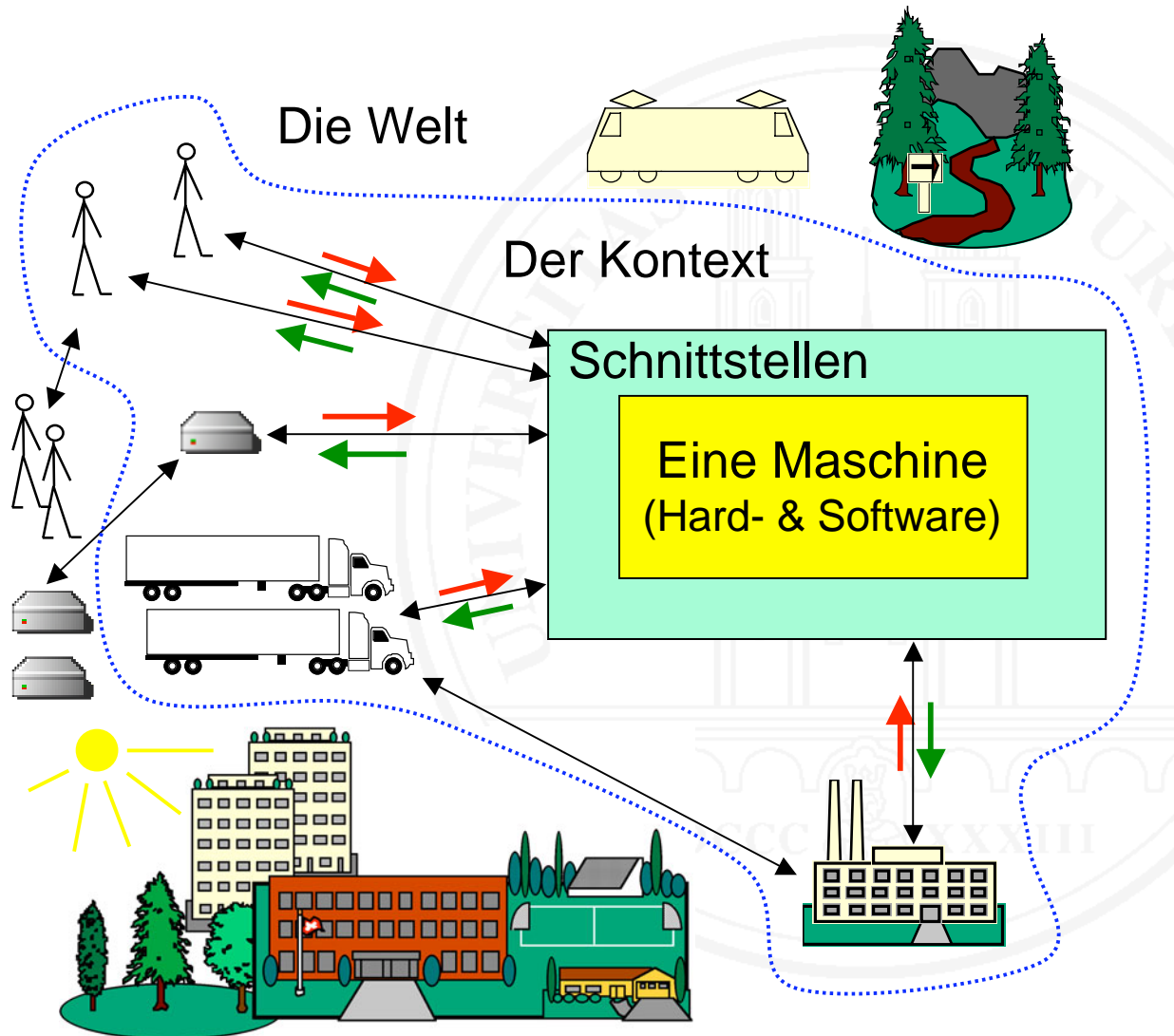


Prüfbar – feststellen können, ob das realisierte System die Anforderungen erfüllt



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

Kontextuelle Anforderungsspezifikation – 1

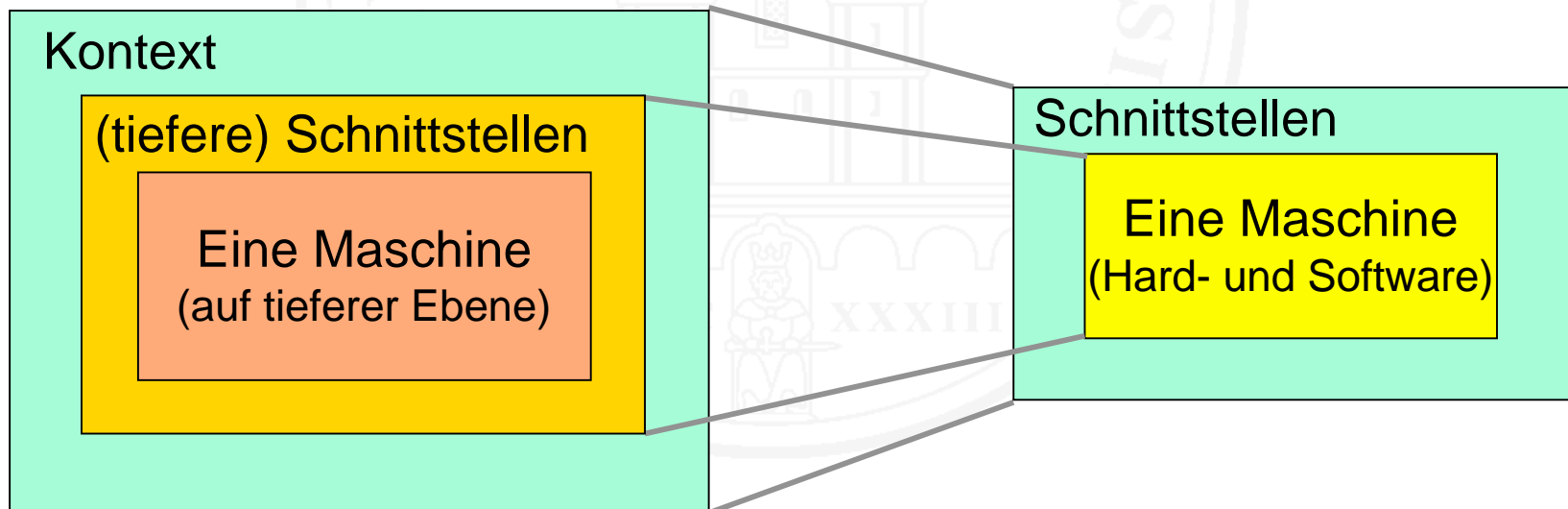


Anforderungsspezifikation bedeutet:

- **Kontext** identifizieren
- **Anstöße** spezifizieren...
- und **Antworten** darauf
- sowie **Restriktionen** (Leistung, Qualität, Randbedingungen)

Kontextuelle Anforderungsspezifikation – 2

- Mehrere Betrachtungsebenen mit unterschiedlichem Kontext, typisch
 - Geschäftsebene
 - Systemebene
 - Softwareebene
 - Oft weitere Ebenen, z. B. Komponentenebene
- ⇒ Verzahnung von Anforderungen und Entwurf



Anforderungen vs. Ziele

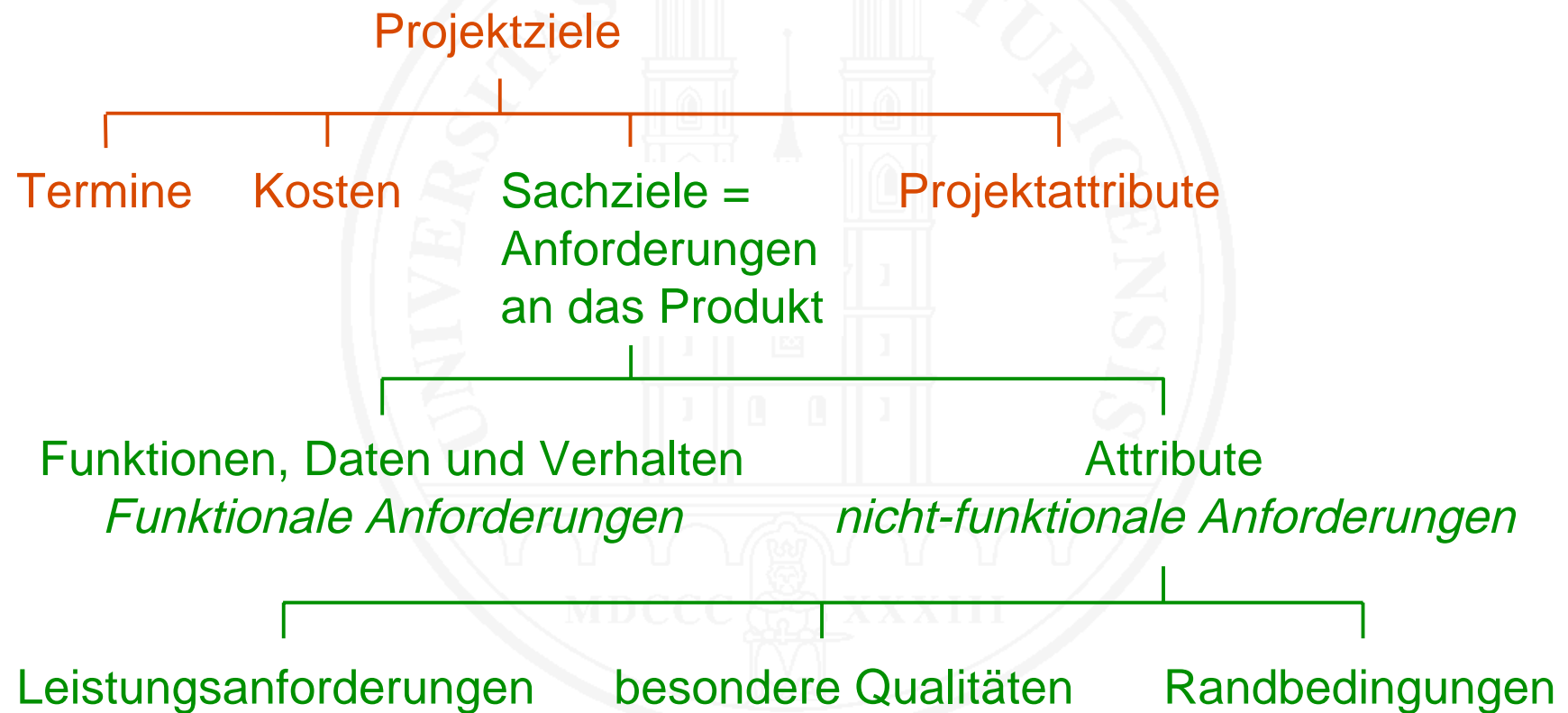
- **Gemeinsamkeiten:** Beide beschreiben etwas zu Erreichendes
- **Unterschiede:**
 - **Ziel:** ein zu erreichender Zustand, meist in umfassendem Sinn gemeint
 - **Anforderung:** eine zu erreichende Eigenschaft, meist auf Detailebene
- **Abgrenzung** im Einzelfall oft schwierig
- Häufig geben **Ziele auf einer Ebene** (z.B. Geschäftsebene) den Rahmen für **Anforderungen auf einer tiefer liegenden Ebene** (z.B. System- oder Softwareebene) vor
- Im Software Engineering sind die **Sachziele eines Projekts** typisch die **Anforderungen an** das zu entwickelnde (Software-)Produkt und die davon betroffenen **Prozesse** (vgl. Kapitel 2)

Anforderungen und Qualität

- **Qualität** – der Grad, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt. (ISO 9000:2000; vgl. Kapitel 2 und 19)
 - Qualität definiert: **Spezifikation** der Anforderungen
 - Qualität erreicht: Die Software erfüllt alle gestellten **Anforderungen**
- Qualitätsmodelle (vgl. Kapitel 2) helfen beim **Strukturieren** von **Anforderungen an besondere Qualitäten**, zum Beispiel
 - Zuverlässigkeit
 - Benutzbarkeit
 - ...
- Qualität ist Zielerfüllung
 - ➔ Anforderungen müssen so gestellt werden, dass ihre **Erfüllung** durch **Prüfung** oder **Messung festgestellt** werden kann

Klassifikation von Anforderungen

Klassische Art der Klassifikation:



Hinweise

- ☆ „Nicht-funktionale Anforderungen“ ist ein sehr gebräuchlicher, aber leider auch sehr unscharfer Begriff
- ☆ Insbesondere ist die **Abgrenzung** zu funktionalen Anforderungen **oft schwierig**.

Beispiel:

„Das System muss den unautorisierten Zugriff auf die Kundenstammdaten verhindern, soweit dies technisch möglich ist“

ist eine Anforderung, welche als **besondere Qualität** zu klassifizieren wäre.

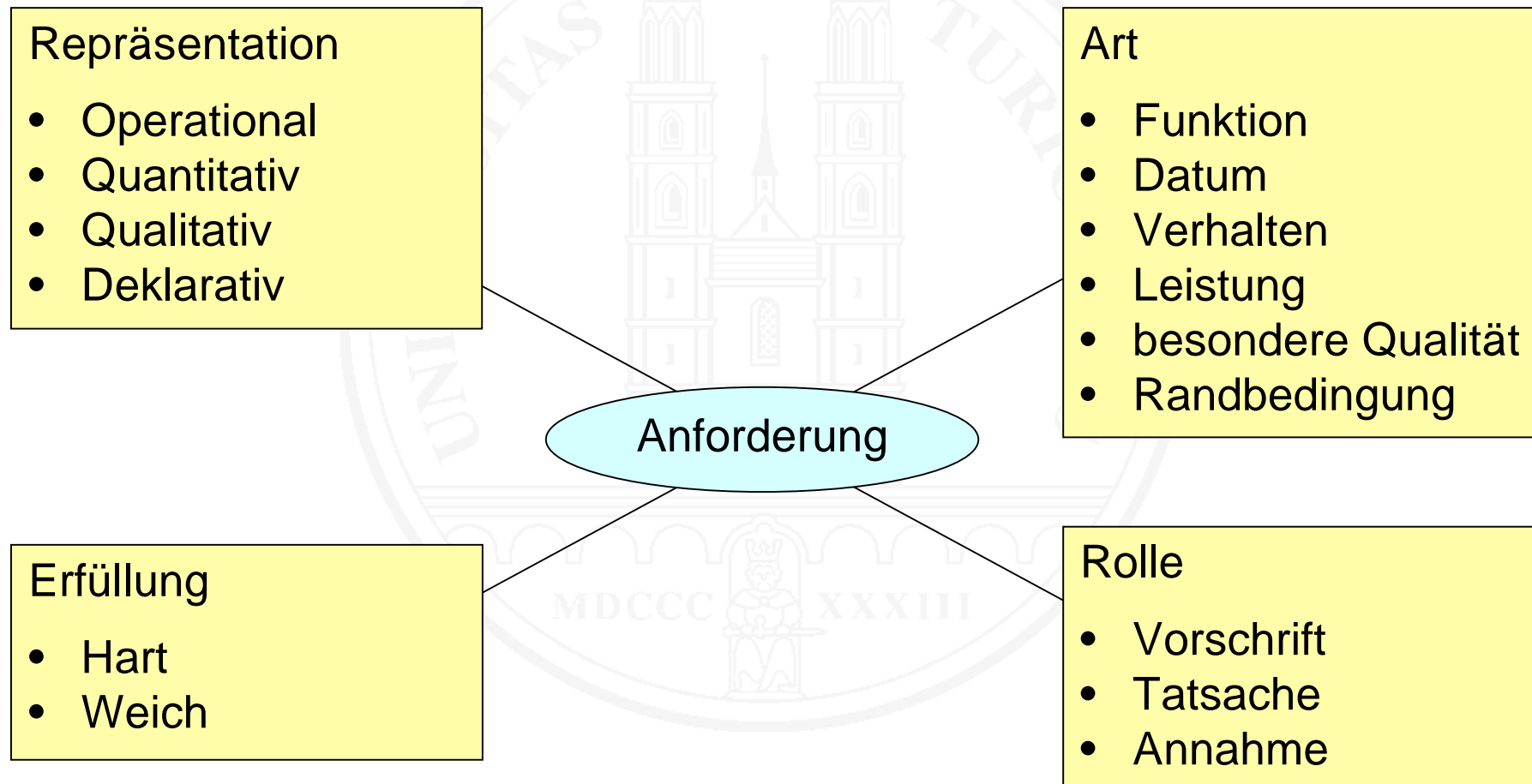
Eine solche Anforderung kann operationalisiert werden, zum Beispiel durch

„Der Zugriff auf die Kundenstammdaten muss über eine Login-Prozedur mit Passwörtern geschützt werden“. „Die Kundenstammdaten müssen verschlüsselt gespeichert werden.“

Dies sind **funktionale Anforderungen**.

Facettierte Klassifikation von Anforderungen

Besser wäre es, Anforderungen nach vier Facetten zu klassifizieren:



Erläuterungen zu den Facetten

Repräsentation

- ✧ **Operational:** „Der Kontostand wird um den eingezahlten Betrag erhöht“
- ✧ **Quantitativ:** „Antwortzeit < 0,5 s“
- ✧ **Qualitativ:** „Das System muss eine hohe Verfügbarkeit aufweisen“
- ✧ **Deklarativ:** „Das System soll auf einer Linux-Plattform laufen“

Erfüllung (vgl. harte und weiche Zielerfüllung in Kapitel 2)

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

Erläuterungen zu den Facetten – 2

Art

- ✧ siehe Darstellungsaspekte in Kapitel 4.4

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?

4.1 Grundlagen

4.2 Der Spezifikationsprozess

4.3 Anforderungsgewinnung und -analyse

4.4 Dokumentation von Anforderungen

4.5 Spezifikationsmethoden und -sprachen

4.6 Prüfen von Anforderungen

4.7 Verwalten von Anforderungen

Was gibt es zu tun im Requirements Engineering?

Zwei Hauptprozesse:

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

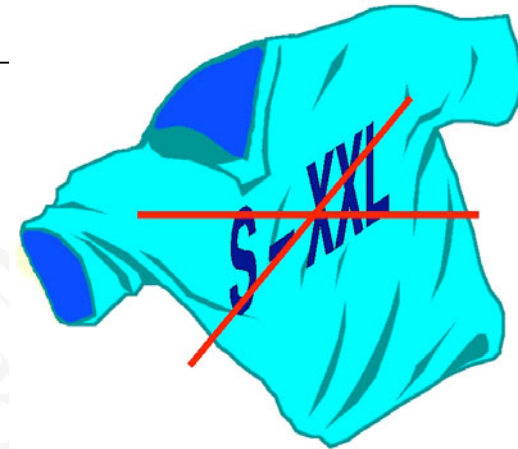
Keine Einheitsgröße für alles

Es gibt keinen idealen RE-Prozess

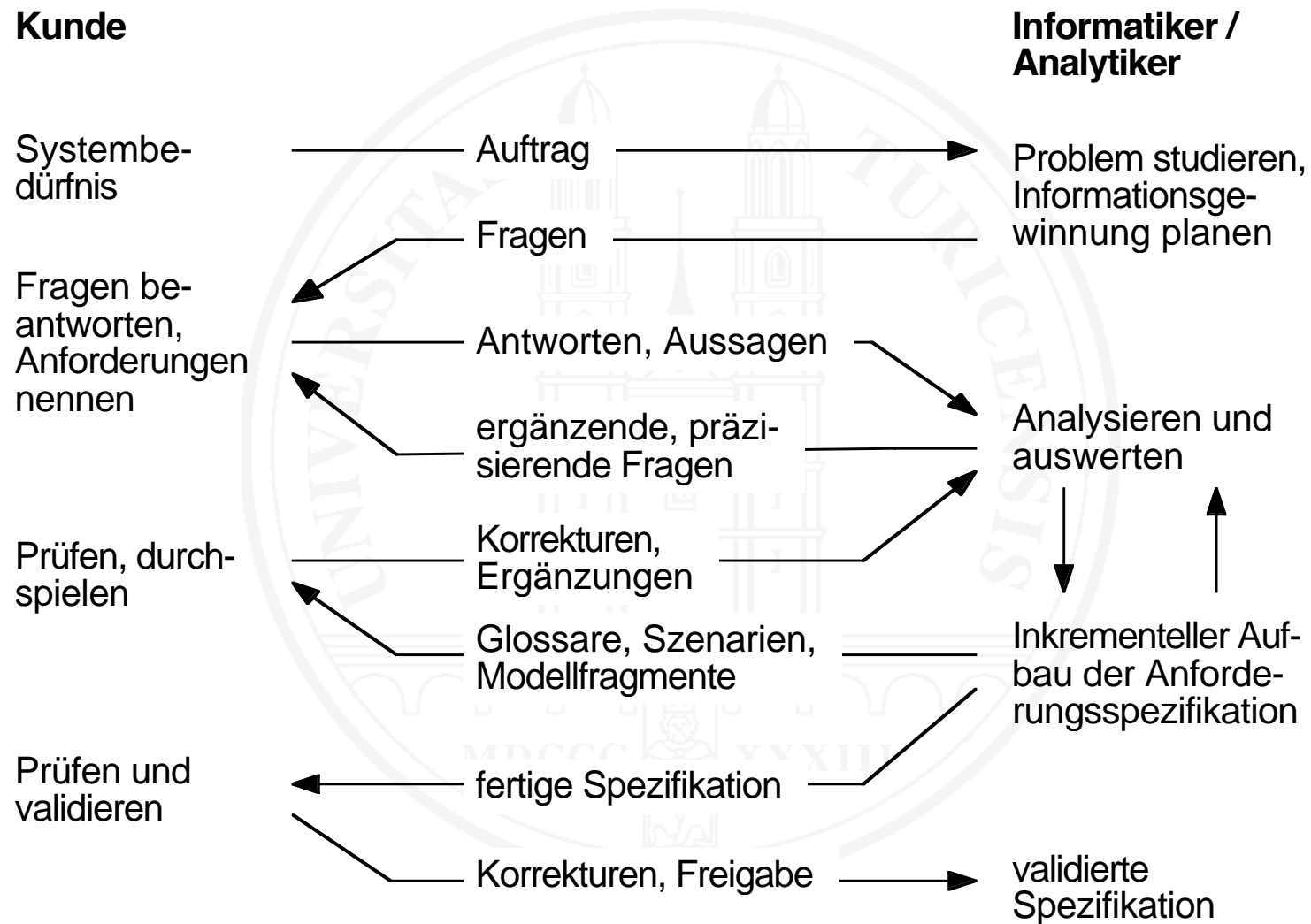
⇒ **zuschneiden** auf konkrete Projektsituation

○ zu berücksichtigende Faktoren:

- lineares oder inkrementelles Vorgehen im Projekt?
- muss die Spezifikation wasserdicht sein (Vertrag; Realisierung durch Dritte)?
- sind die Kunden/Benutzer bekannt und können sie in die Erstellung der Spezifikation einbezogen werden?
- wird das zu spezifizierende System im Kundenauftrag oder für den Markt entwickelt?
- soll Standardsoftware zum Einsatz kommen?



Möglicher Ablauf des Spezifikationsprozesses



4.1 Grundlagen

4.2 Der Spezifikationsprozess

4.3 Anforderungsgewinnung und -analyse

4.4 Dokumentation von Anforderungen

4.5 Spezifikationsmethoden und -sprachen

4.6 Prüfen von Anforderungen

4.7 Verwalten von Anforderungen

Informationsquellen

- Mit Leuten reden / Leute befragen
 - Welche **Beteiligten (stakeholders)** gibt es, die Anforderungen stellen können?
 - Häufig nicht Individuen, sondern **Rollen**
 - Typische Beteiligtenrollen: Endbenutzer, Auftraggeber, Betreiber, Entwickler, Projektleitung, ...
 - Diverse Gesprächs- und Befragungstechniken
- Prozessabläufe beobachten
 - Wie wird heute gearbeitet?
 - Was ist gut und was soll anders werden?
 - Wer verwendet wo welche Daten?
- Unterlagen studieren
 - Ausschreibungstexte, Visionsdokumente, ...

Mini-Übung 4.2: Beteiligtenanalyse

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.

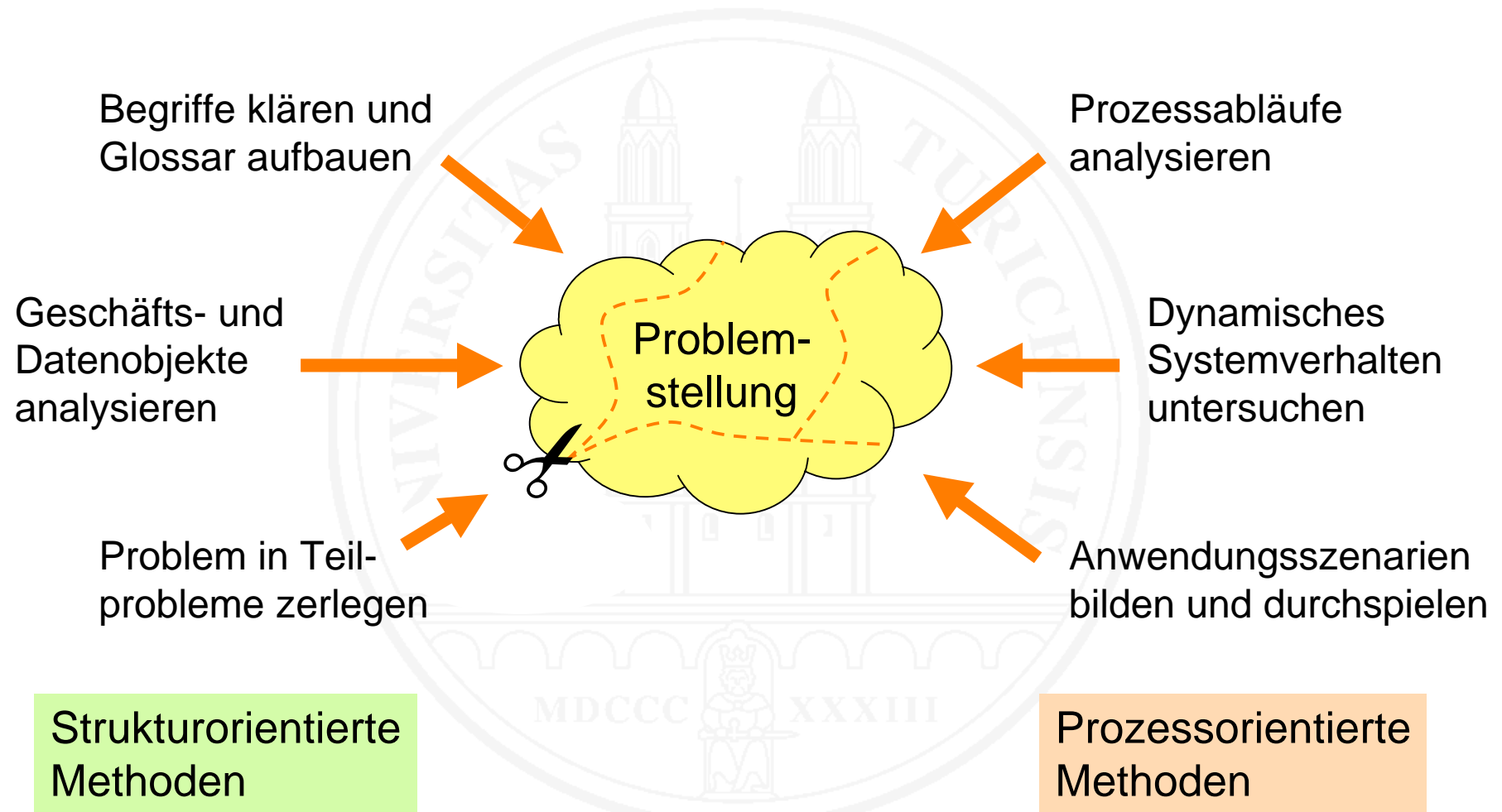
Ferner ist ein Diebstahlsicherungssystem zu installieren, das verhindert, dass jemand die Bibliothek mit nicht ausgeliehenen Büchern verlässt.

Identifizieren Sie Beteiligte, die im Rahmen dieses Projekts Anforderungen stellen können.

Techniken der Informationsbeschaffung

- **Interviews**
Beteiligte werden einzeln oder in Gruppen befragt
- **Umfragen/Fragebogen**
Erfassung von Begriffswelt und Bedürfnissen einer größeren Gruppe von Beteiligten
- **Beobachtung**
Beteiligte bei der Arbeit beobachten
- **Gemeinsame Arbeitstagen**
Gemeinsame Erarbeitung von Anforderungen in einer Gruppe ausgewählter Beteiligter und Anforderungsingenieure
- **Prototypen**
Gewinnen von Anforderungen durch Erprobung möglicher Lösungen
- ...

Gewinnung und Analyse: Methodische Ansätze



Regeln

- Informationen über den Anwendungsbereich gewinnen und analysieren
 - Begriffswelt
 - Gegenstände und Prozesse
- Konkrete Bedürfnisse und Wünsche erfassen und analysieren
- Anforderungsspezifikation fortlaufend inkrementell aufbauen
 - Keine großen Materialsammlungen
 - Gewinnung, 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

Risiken und Probleme

- **Erwartungs-** und **Begriffsdiskrepanzen** bei den Beteiligten
- Beteiligte wissen zwar, was sie wollen, **können** ihre **Vorstellungen** aber **nicht formulieren**
- Beteiligte **wissen nicht, was sie wollen**
- Beteiligte haben **verdeckte Ziele**, die sie absichtlich nicht offen legen
- Beteiligte sind **auf bestimmte Lösungen fixiert**
- ⇒ Requirements Engineering ist immer auch
 - Aufgabenklärung
 - Risikoanalyse
 - Konsensbildung
 - Konflikterkennung und -auflösung
 - Anregung von Kreativität bei den Beteiligten

Anforderungen priorisieren

- Nicht alle Anforderungen sind gleich wichtig:
 - **Muss**-Anforderungen – unverzichtbar
 - **Soll**-Anforderungen – wichtig, aber bei zu hohen Kosten verzichtbar
 - **Wunsch**-Anforderungen – schön zu haben, aber nicht essenziell
- Priorisierung nötig bei
 - harten **Terminen**
 - harten **Preisobergrenzen**
 - **Beschaffung**
- Priorisierung nützlich bei
 - Festlegung von Inhalt und Umfang der Inkremente bei **inkrementeller Entwicklung**
 - **Releaseplanung** bei der Weiterentwicklung bestehender Systeme

4.1 Grundlagen

4.2 Der Spezifikationsprozess

4.3 Anforderungsgewinnung und -analyse

4.4 Dokumentation von Anforderungen

4.5 Spezifikationsmethoden und -sprachen

4.6 Prüfen von Anforderungen

4.7 Verwalten von Anforderungen

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

- Datenmengen (durchschnittlich/im Extremfall)
- Verarbeitungs- /Reaktionsgeschwindigkeit (durchschnittlich/im Extremfall)
- Verarbeitungszeiten und -intervalle
- Wo immer möglich: messbare Angaben!

○ Qualitätsaspekt

- Geforderte besondere Qualitäten (z.B. Benutzerfreundlichkeit, Zuverlässigkeit)

Inhalt einer Anforderungsspezifikation – 3

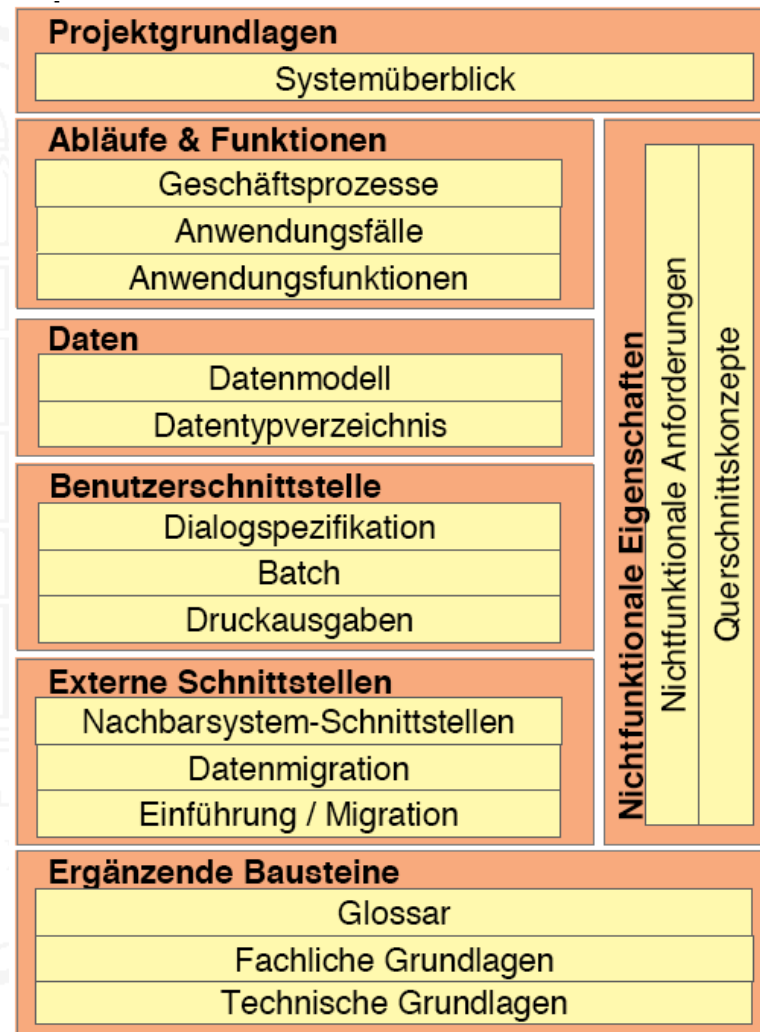
- **Randbedingungsaspekt**
 - Einzuhaltende/zu verwendende **Schnittstellen**
 - Normen und **Gesetze**
 - Datenschutz, **Datensicherung**
 - Explizite **Vorgaben** des Auftraggebers

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&m¹⁾

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



Darstellungsregeln

- Einzelanforderungen in **kleinen Einheiten** fassen: eine Kernaussage pro Einzelanforderung
- Zu jedem geforderten **Resultat** die **Funktion** und die **Eingabedaten**, welche das Resultat erzeugen, spezifizieren
- Mögliche **Ausnahmesituationen** spezifizieren
- **Implizite Annahmen** aufdecken und explizit formulieren
- Leistungs- und Qualitätsanforderungen **quantitativ** spezifizieren
- **All-Quantifizierungen** kritisch hinterfragen
- Anforderungen **strukturieren** (zum Beispiel durch Kapitelgliederung)
- **Redundanz** nur dort, wo für leichtes Verständnis notwendig
- **Präzise** spezifizieren

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: $\text{Real} \rightarrow \text{Real}$;
Pre: $x \geq 0$;
Post: $|\text{Sqrt}^2(x) - x| < \varepsilon \wedge \varepsilon \leq 10^{-16} \wedge \varepsilon \leq 10^{-6}x$.

Deskriptive Darstellung – 2

- **Vorteile:**

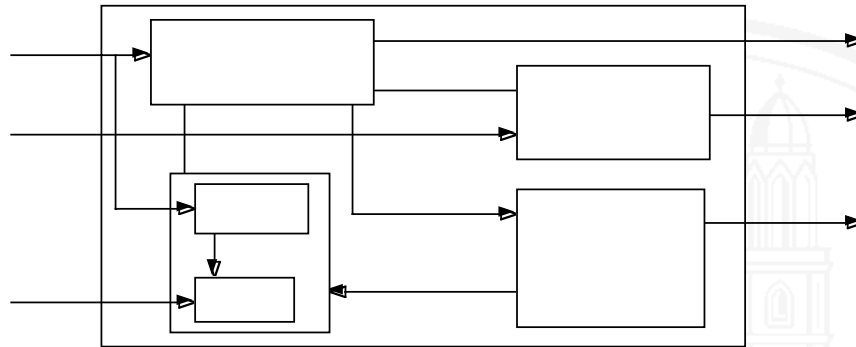
- + Nur **äußeres Verhalten** spezifiziert
- + **Lösungsneutral**

- **Nachteile:**

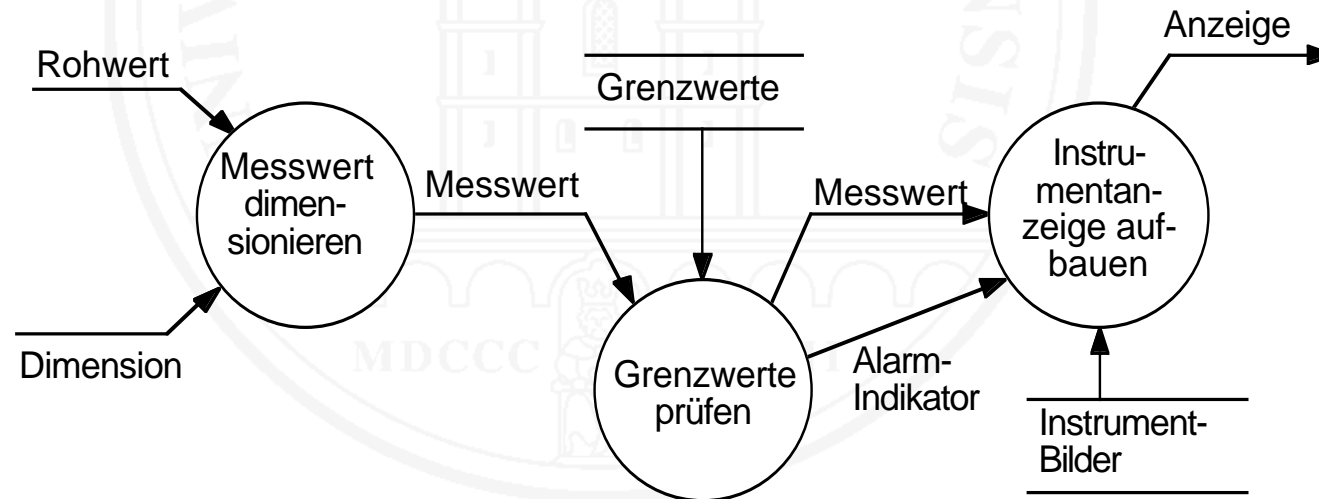
- Bei Verwendung von Text: umfangreich und **wenig strukturiert**; Zusammenhänge nicht erkennbar; **fehlerträchtig** und schwierig zu prüfen
- Bei Verwendung formaler Mitteln: **sehr schwierig zu erstellen**; **Prüfung auf Adäquatheit oft fast unmöglich**

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

Konstruktive Darstellung – 1



Beispiel



Konstruktive Darstellung – 2

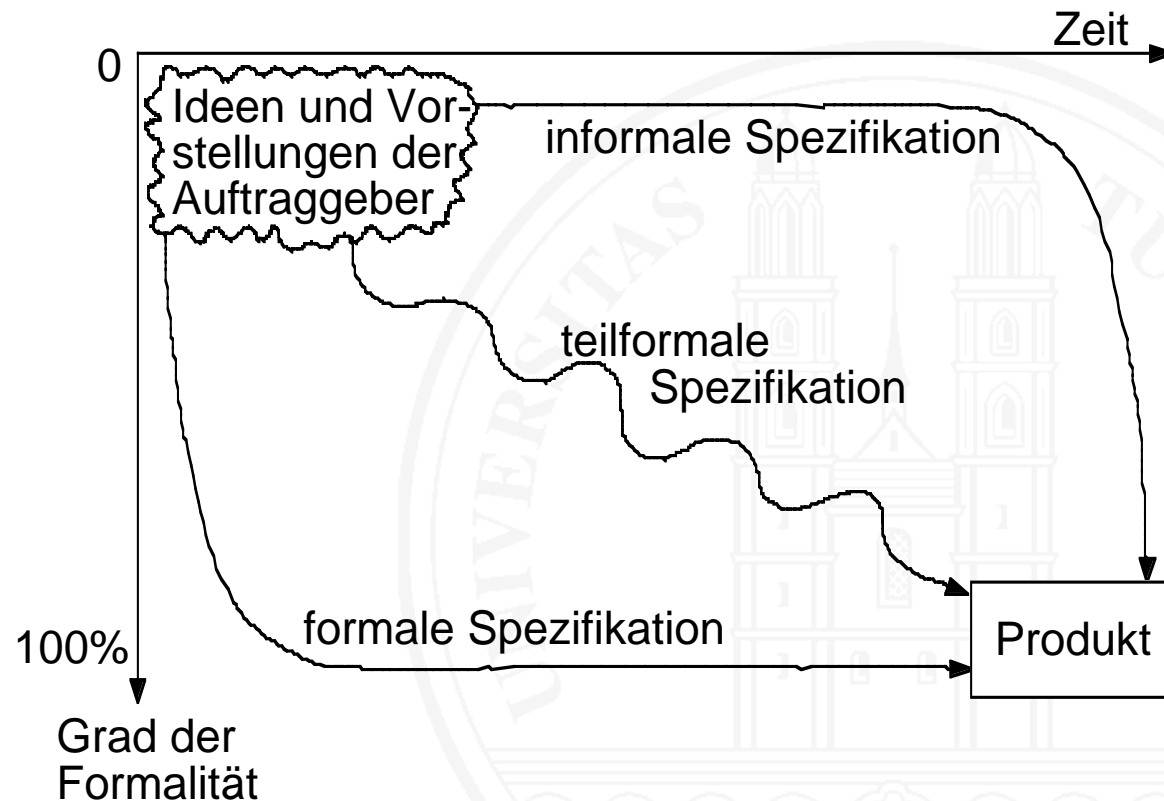
Vorteile:

- + **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 4.3: 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 nebenstehenden 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

4.1 Grundlagen

4.2 Der Spezifikationsprozess

4.3 Anforderungsgewinnung und -analyse

4.4 Dokumentation von Anforderungen

4.5 Spezifikationsmethoden und -sprachen

4.6 Prüfen von Anforderungen

4.7 Verwalten von Anforderungen

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

- Weit verbreitet
- Nummerierung und Gliederungsschemata als Strukturierungsmittel
- Linguistische Methoden zur Verbesserung der Qualität
- + Leicht zu schreiben und zu lesen
- + Ausdrucksmächtig
- Unübersichtlich
- Fehlerträchtig
- Schwierig zu prüfen
- ⇒ Weniger geeignet als alleiniges Mittel zur Beschreibung von Spezifikationen

Regeln für natürlichsprachliche Anforderungen

- 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
- Für **Verben**, die Prozesse beschreiben, **feste Bedeutungen** festlegen
- **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**

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

$\forall s \in \text{Stack}, e \in \text{elem}$

(1) $\neg \text{full}(s) \rightarrow \text{pop}(\text{push}(s,e)) = s$

-- Pop hebt den Effekt von Push auf

(2) $\neg \text{full}(s) \rightarrow \text{top}(\text{push}(s,e)) = e$

-- Top liefert das zuletzt gespeicherte Element

(3) $\text{empty}(\text{new}) = \text{true}$

-- ein neuer Keller ist leer

(4) $\neg \text{full}(s) \rightarrow \text{empty}(\text{push}(s,e)) = \text{false}$

-- nach Push ist ein Keller nicht mehr leer

(5) $\text{full}(\text{new}) = \text{false}$

-- ein neuer Keller ist nicht voll

(6) $\neg \text{empty}(s) \rightarrow \text{full}(\text{pop}(s)) = \text{false}$

-- nach Pop ist ein Keller niemals voll

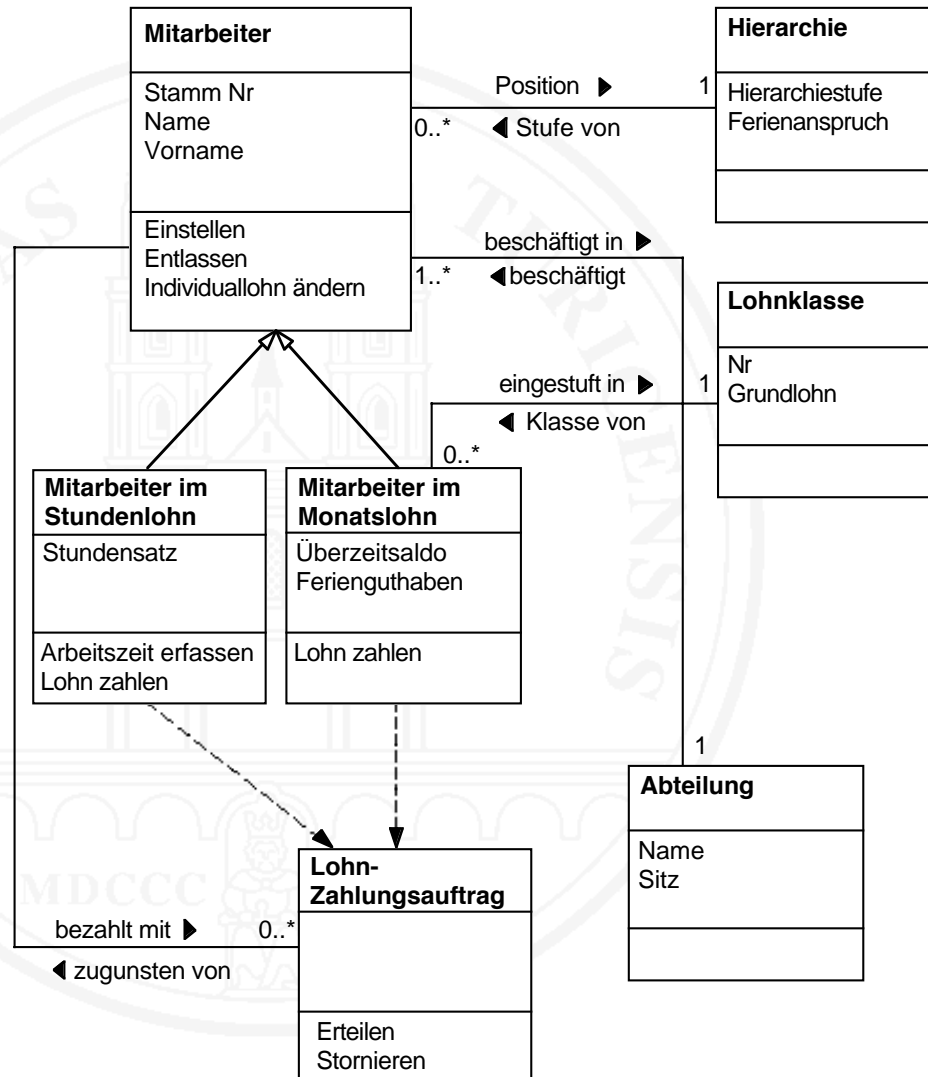
Objektorientierte Spezifikation

- 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

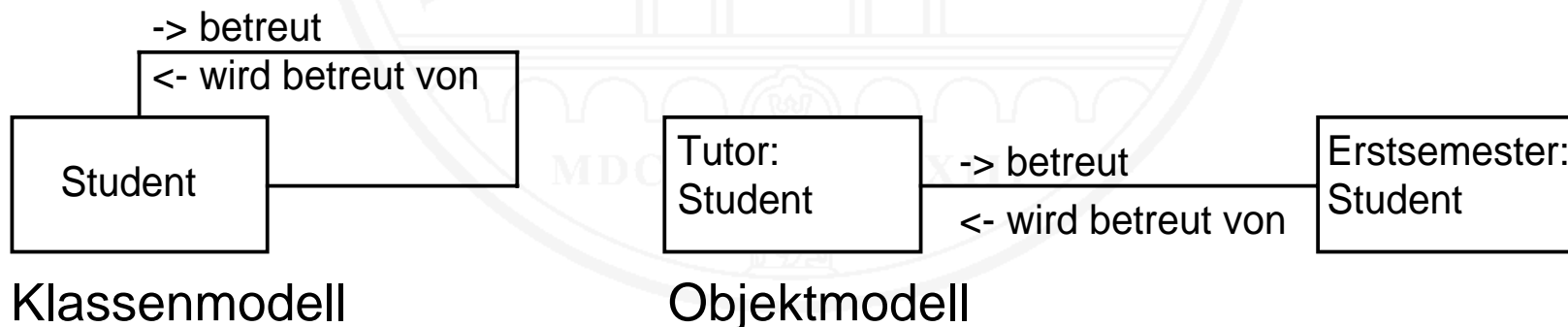
Beispiel eines Klassenmodells...

...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



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

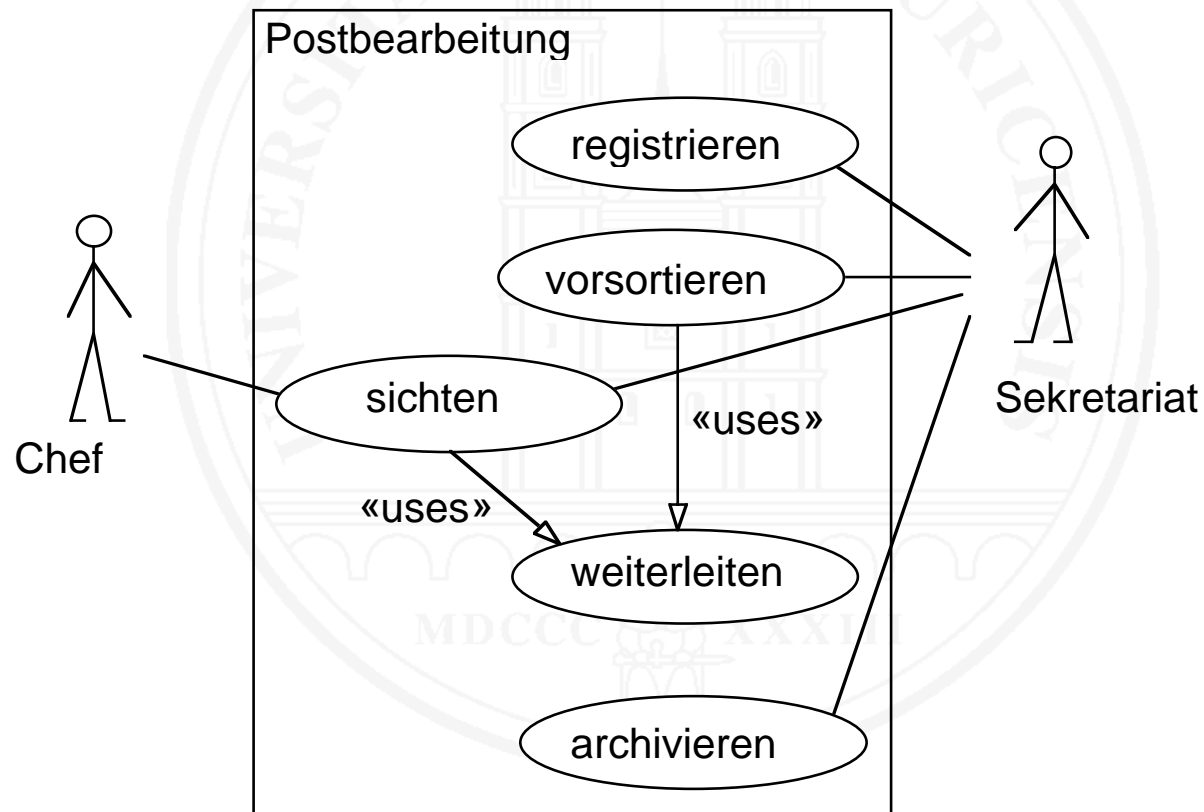
+ Dekomposition möglich

– Zusammenhänge / Abhängigkeiten zwischen Szenarien nicht modelliert

– Statische Struktur nicht modelliert

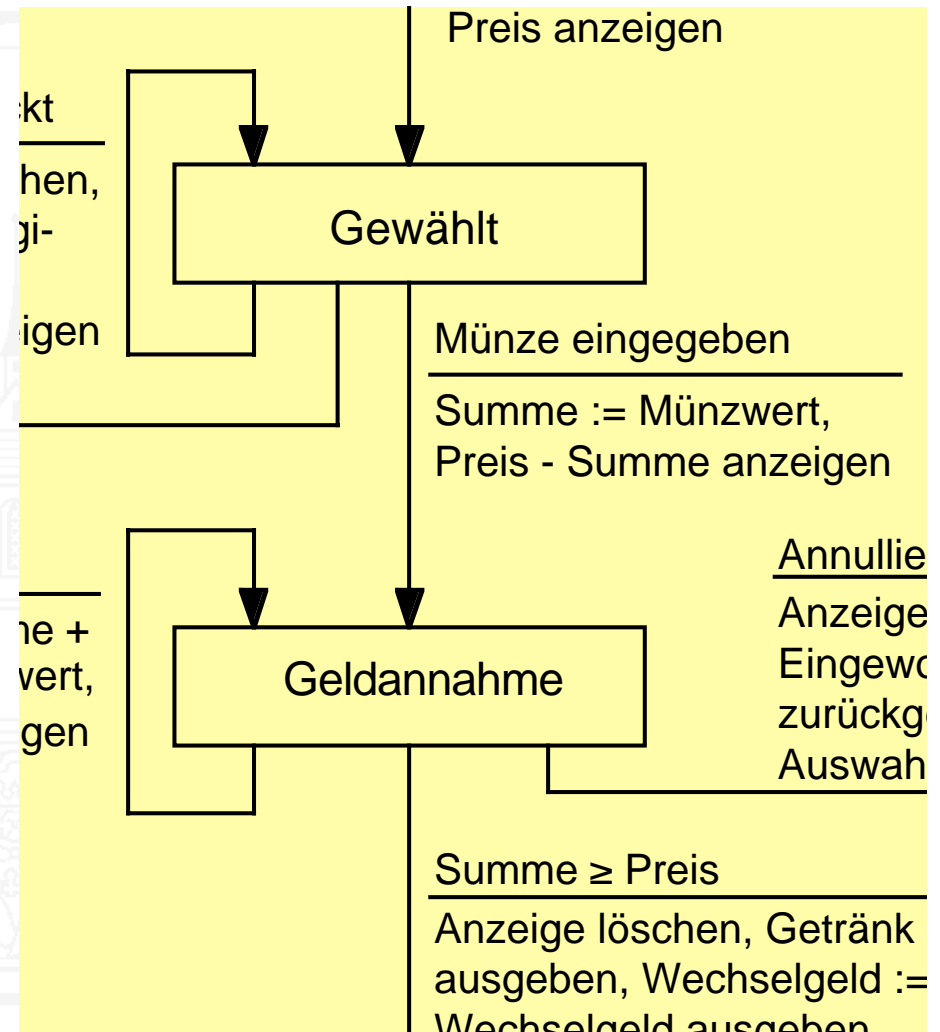
Spezifikation mit Anwendungsfällen – 3

- 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 genannt, aber nicht modelliert
- Statische Struktur nicht modelliert



4.1 Grundlagen

4.2 Der Spezifikationsprozess

4.3 Anforderungsgewinnung und -analyse

4.4 Dokumentation von Anforderungen

4.5 Spezifikationsmethoden und -sprachen

4.6 Prüfen von Anforderungen

4.7 Verwalten von Anforderungen

Prinzipien



- **Abweichungen** von der geforderten **Qualität** der Spezifikation **feststellen**
- möglichst viele **Fehler, Lücken, Unklarheiten, Mehrdeutigkeiten**, etc. **finden** und **beheben**
- **Konflikte** zwischen den Wünschen / Forderungen verschiedener beteiligter Personen **erkennen** und **lösen**
- **Verdeckte Wünsche / Erwartungen / Befürchtungen** **erkennen** und thematisieren

Organisation

- **Vertreter aller Beteiligten einbeziehen**

- Endbenutzer
- Auftraggeber
- Betreiber
- Entwickler
- Projektleitung
- ...

Zeitpunkt(e):

- (1) **Fortlaufend**, d.h. begleitend zur Erstellung der Spezifikation, z. B. Autor-Kritiker-Zyklus
- (2) **Wenn** die Spezifikation **fertig** ist (aber noch genug Zeit bleibt, die gefundenen Mängel zu beheben)

Prüfverfahren

- **Review** (vgl. Kapitel 9)
 - 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**

Prüfverfahren – 2

- **Simulation/Animation**
 - Untersuchung der **Adäquatheit** des Systemverhaltens
 - Dynamisches Verhalten des spezifizierten Systems wird durch Simulator ausgeführt und/oder durch Animator visualisiert
- **Prototyp** (vgl. Kapitel 13)
 - Beurteilung der Adäquatheit / praktischen Brauchbarkeit des spezifizierten Systems anhand der im Prototyp realisierten Systemteile
 - **Erprobung** eines Systems in der geplanten **Einsatzumgebung**
 - Modell für das zu schaffende Produkt
 - Mächtigstes (aber auch aufwendigstes) Mittel zur Beurteilung der **Adäquatheit**

4.1 Grundlagen

4.2 Der Spezifikationsprozess

4.3 Anforderungsgewinnung und -analyse

4.4 Dokumentation von Anforderungen

4.5 Spezifikationsmethoden und -sprachen

4.6 Prüfen von Anforderungen

4.7 Verwalten von Anforderungen

Requirements Management

- Anforderungen **stabil halten** und
- Veränderung **kontrolliert zulassen**

- ⇒ **Konfigurationsmanagement** für Anforderungen
 - Anforderungen einzeln **identifizierbar**
 - Geordneter **Änderungsprozess**
 - Klare **Zuständigkeiten** und **Verantwortlichkeiten**
 - **Rückverfolgbarkeit** aller Entscheide und Änderungen

- ⇒ **Verfolgbarkeit** (traceability)
 - **Rückwärts**: Wo kommt welche Anforderung her?
 - **Vorwärts**: Wo ist welche Anforderung entworfen bzw. implementiert?
 - Wie **hängen** Anforderungen voneinander **ab**?

- ⇒ Beherrschen der **Evolution** von Anforderungen

Literatur

Siehe Literaturverweise im Kapitel 7 des Skripts.

Im Skript [M. Glinz (2005). *Software Engineering*. Vorlesungsskript, Universität Zürich] lesen Sie Kapitel 7.

Im Begleittext zur Vorlesung [S.L. Pfleeger, J. Atlee (2006). *Software Engineering: Theory and Practice*, 3rd edition. Upper Saddle River, N.J.: Pearson Education International] lesen Sie Kapitel 4 und Kapitel 6.3-6.4.

