Software Engineering I

Prof. Dr. Martin Glinz

Kapitel 7

Spezifikation von Anforderungen

Universität Zürich
Institut für Informatik

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 – 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

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.

Pflichtenheft ➜ verschiedene Begriffe:

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

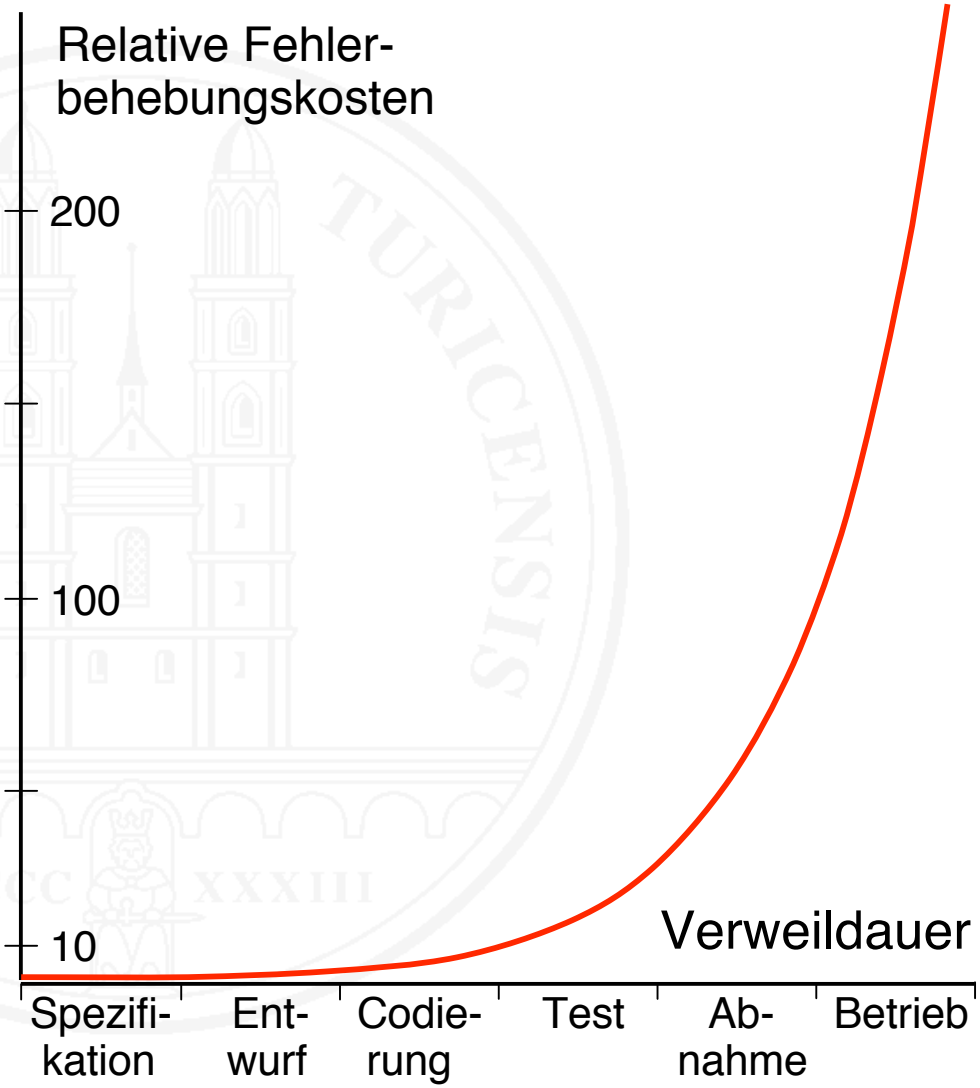
Anforderungsspezifikation – Warum (1)

- Mehr verdienen
  - Kosten senken
    - geringere Herstellungskosten (Senken der Fehlerkosten!)
    - geringere Pflegekosten
  - Mehr verkaufen
    - zufriedenere Kunden
    - früher am Markt
- Projektrisiken beherrschen
  - Risiken verkleinern
  - Zuverlässige Prognosen für Termine und Kosten

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

Anforderungsspezifikation – Warum (2)

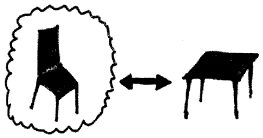
Kosten für die Behebung von Fehlern abhängig von ihrer Verweildauer in der Software



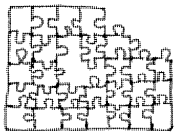
Wirtschaftlichkeit der Anforderungsspezifikation

- Wenig Anforderungs-fehler machen
- Möglichst viele der den-noch gemachten Fehler früh finden

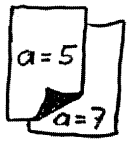
Merkmale einer guten Spezifikation



Adäquatheit – das beschreiben, was der Kunde will bzw. braucht



Vollständigkeit – alles beschreiben, was der Kunde will bzw. braucht



Widerspruchsfreiheit – sonst ist die Spezifikation nicht realisierbar

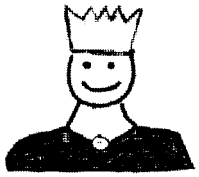
$$\bigwedge_{n\in\mathbb{N}} \bigwedge_{m_i \wedge s_i \in \Sigma} \sum_{i=1}^{n} |m_i| \geq P \wedge$$
$$\sum_{i=1}^{n-1} |m_i| < P \leftrightarrow W(m_1 \circ \ldots \circ m_n) \wedge \neg W(m_1 \circ \ldots \circ m_{n-1})$$

Verständlichkeit – für den Kunden und für die Informatiker

Eindeutigkeit – damit Fehler durch Fehlinterpretationen vermieden werden

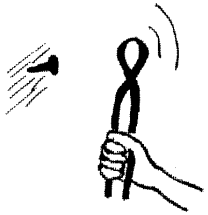
Prüfbarkeit – feststellen können, ob das realisierte System die Anforderungen erfüllt

Merkmale eines guten Spezifikationsprozesses

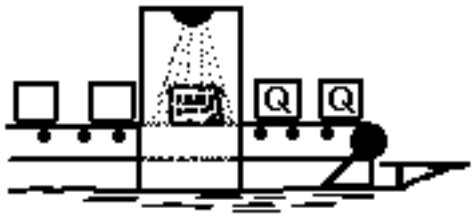


Kundenorientierung

Methodisches und zielgerichtetes Vorgehen



Verwendung geeigneter Mittel



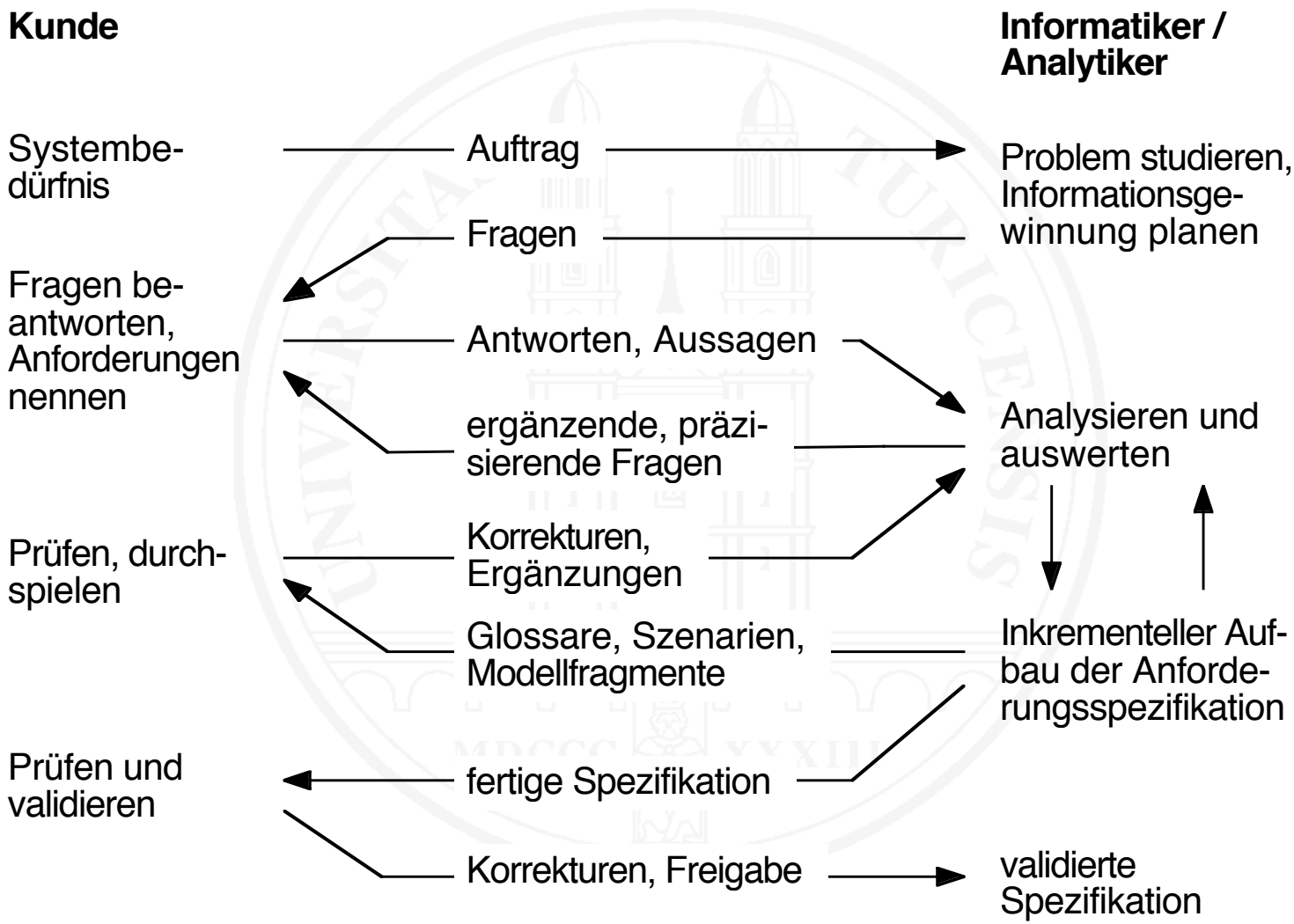
Integration von Erstellung und Prüfung von Anforderungen

Der Spezifikationsprozess

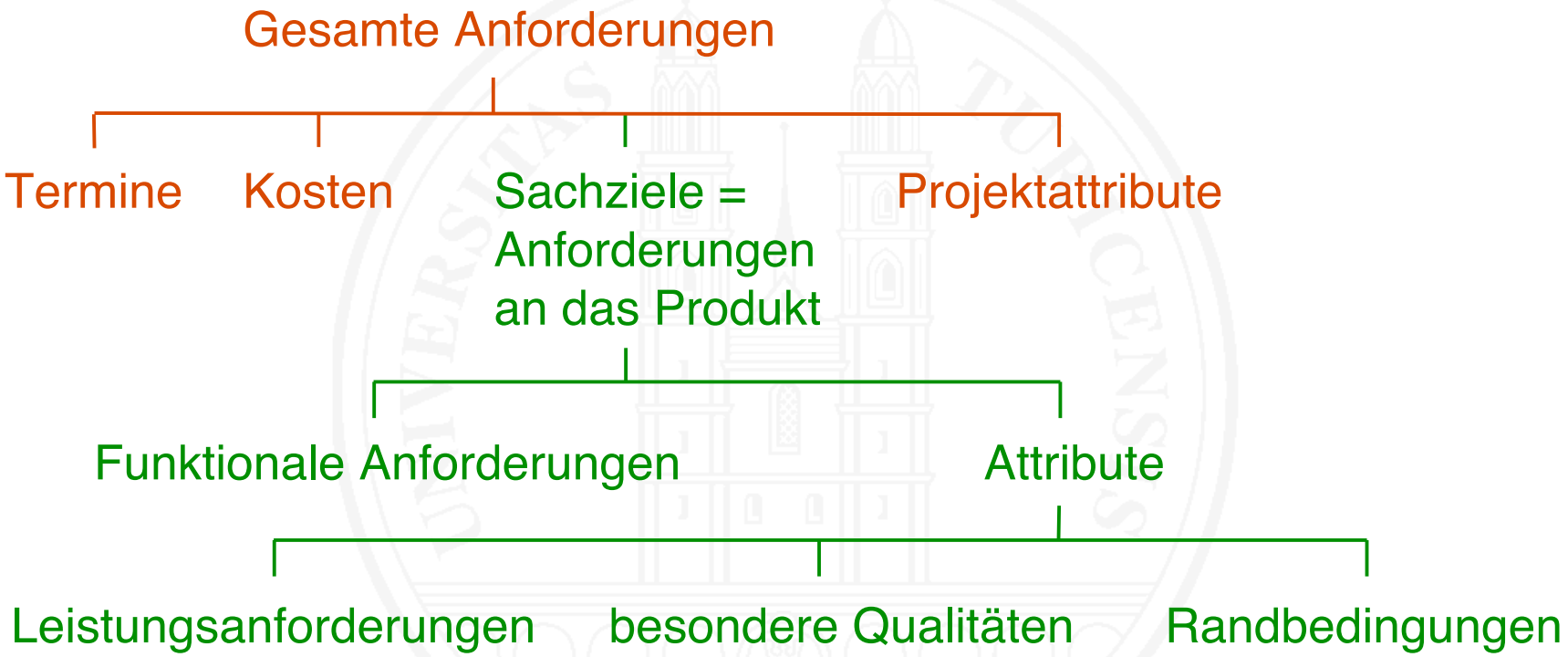
„Der“ Spezifikationsprozess besteht aus einer Reihe von Teilprozessen:

- Anforderungen spezifizieren
  - gewinnen
  - dokumentieren
  - prüfen
- Anforderungen verwalten
  - freigeben
  - ändern
  - rückverfolgen (wo kommt eine Anforderung her)
  - vorwärtsverfolgen (wo wird eine Anforderung verwendet)

Möglicher Ablauf des Spezifikationsprozesses



Klassifikation von Anforderungen



[siehe auch Kapitel 2]

Priorisierung von Anforderungen

- Muss-Anforderungen – unverzichtbar
- Soll-Anforderungen – wichtig, aber bei zu hohen Kosten verzichtbar
- Wunsch-Anforderungen – schön zu haben, aber nicht essenziell

Nötig bei

- harten Preisobergrenzen
- Beschaffung

Nützlich bei

- Festlegung von Inhalt und Umfang der Inkremente bei inkrementeller Entwicklung
- Releaseplanung bei der Weiterentwicklung bestehender Systeme

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 (nicht-funktionale) 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 (zum Beispiel IEEE 830-1993)
- Wahl der Gliederung hat großen Einfluss auf Verständlichkeit

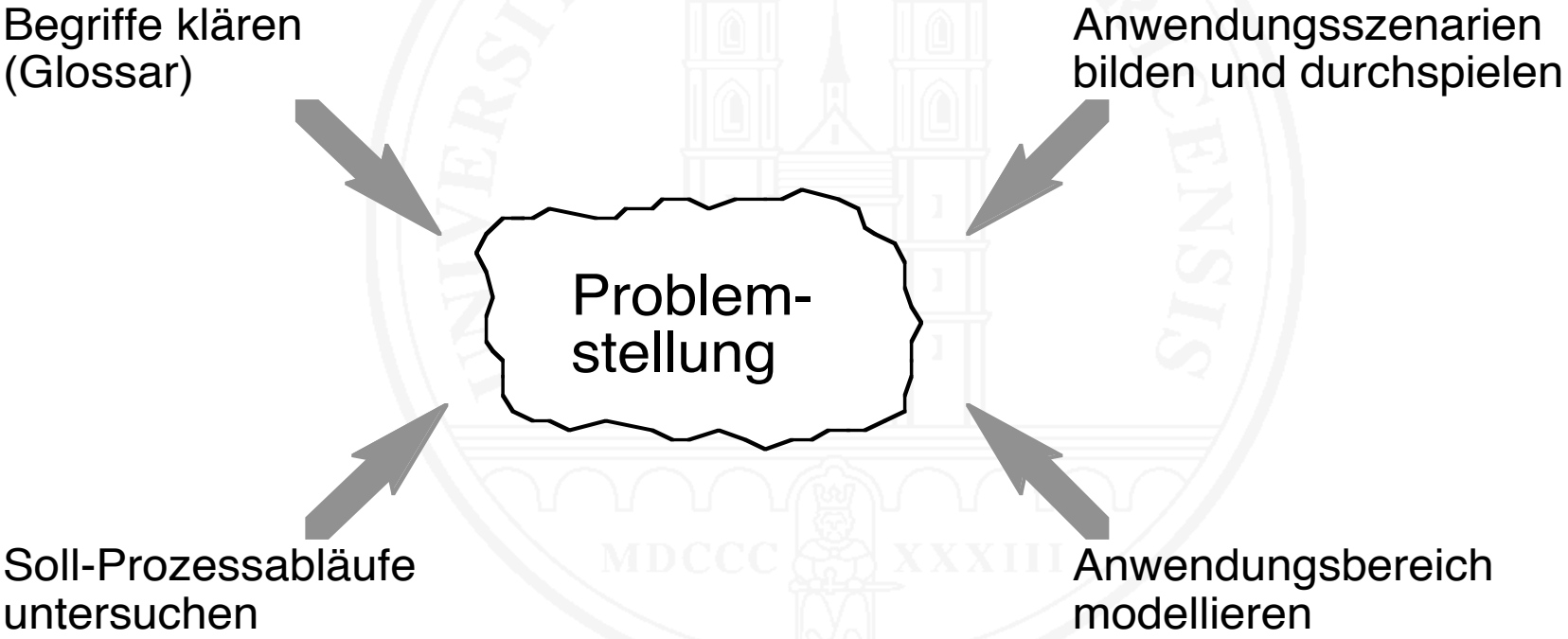
Gewinnung von Anforderungen – 1: Probleme

- Erwartungs- und Begriffsdiskrepanzen beim Kunden
- Kundenvertreter können ihre Vorstellung nicht formulieren
- Kundenvertreter wissen nicht, was sie wollen

⇒ Requirements Engineering ist immer auch Konsensbildung

Gewinnung von Anforderungen – 2: Ansätze

Mögliche Ansätze für die Anforderungsgewinnung:



Techniken zur Informationsgewinnung

- Interviews
  Vertreter des Kunden werden einzeln oder in Gruppen zu höchstens drei Leuten befragt

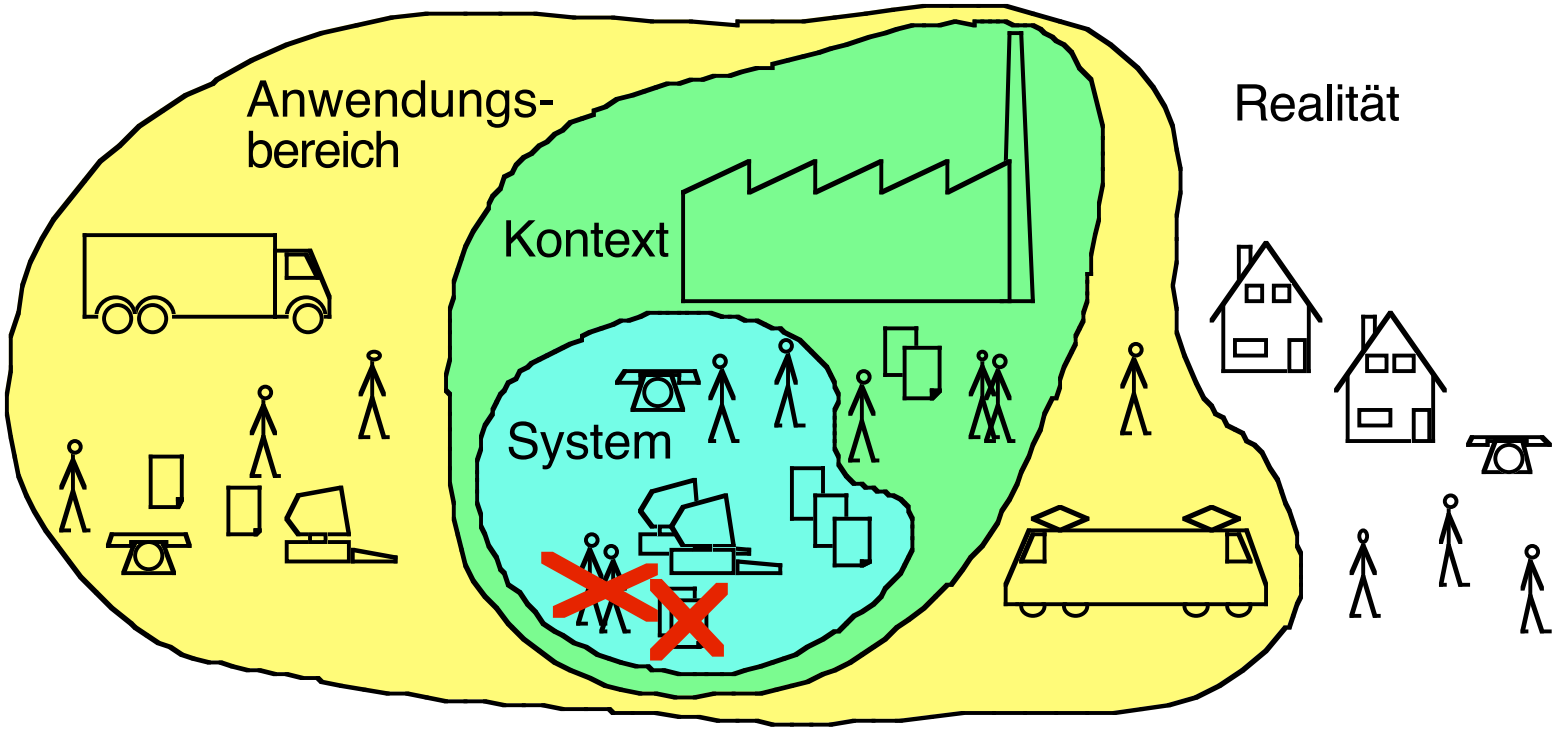
- Fragebogen
  Erfassung von Begriffswelt und Bedürfnissen einer größeren Gruppe von Kundenvertretern

- Gemeinsamen Arbeitstagungen
  Gemeinsame Erarbeitung von Anforderungen durch eine Gruppe ausgewählter Kundenvertreter und Informatiker

Einbettung und Abgrenzung

Einbettung eines Informatiksystems in seinem Umfeld:

Realität ➔ Anwendungsbereich ➔ Kontext ➔ (Informatik-)System



Die Rolle der IST-Analyse

- Traditionelles Vorgehen
  (vor allem bei Informationssystemen; McMenamin und Palmer 1984):
  (1) Physischen IST-Zustand erheben
  (2) Essentiellen IST-Zustand extrahieren
  (3) Essentiellen SOLL-Zustand (= Anforderungen) ermitteln
- Heute nicht mehr zeitgemäß
- Beschäftigung mit IST-Zustand nur wenn notwendig
  - zum Verstehen der Arbeit und der Bedürfnisse der Benutzer
  - zur Ermittlung von Stärken und Schwächen des IST-Systems
  - zur Übernahme von Teilen eines IST-Systems

Darstellung von Anforderungen

Die möglichen Darstellungsformen unterscheiden sich konzeptionell vor allem in drei Aspekten:

- Art der Darstellung: konstruktiv oder deskriptiv
- Formalitätsgrad der Darstellung
- Verfügbare Gliederungs- und Abstraktionsmittel

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: $x \geq 0$;

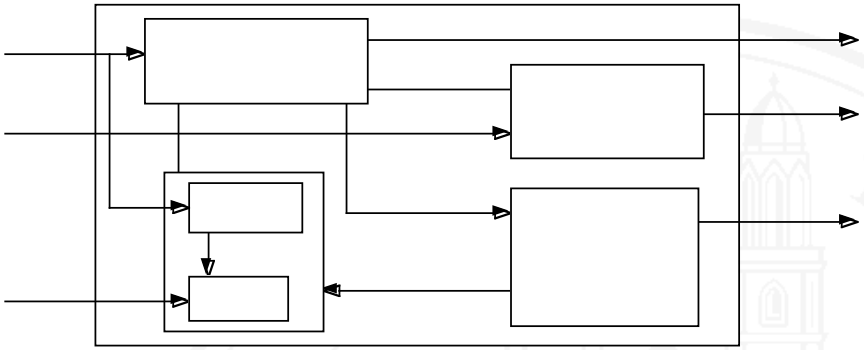
  Post: $|Sqrt^2(x) - x| < \varepsilon \wedge \varepsilon \leq 10^{-16} \wedge \varepsilon \leq 10^{-6}x$.

Deskriptive Darstellung – 2

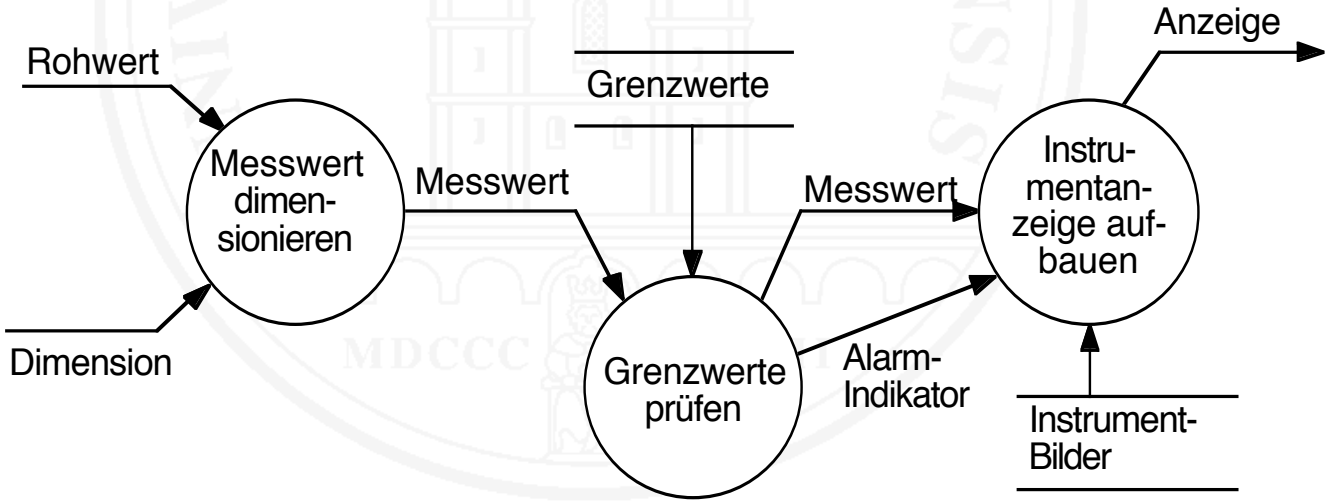
- Vorteile:
  - \+ Nur äußeres Verhalten spezifiziert
  - \+ Lösungsneutral
- Nachteile:
  - – Bei Verwendung von Text: umfangreich und wenig strukturiert; Zusammenhänge nicht erkennbar; fehlerträchtig und schwierig zu prüfen
  - – Bei Verwendung formaler Mitteln: sehr schwierig zu erstellen; Prüfung auf Adäquatheit oft fast unmöglich

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

Konstruktive Darstellung – 1



Beispiel



Konstruktive Darstellung – 2

Vorteile:

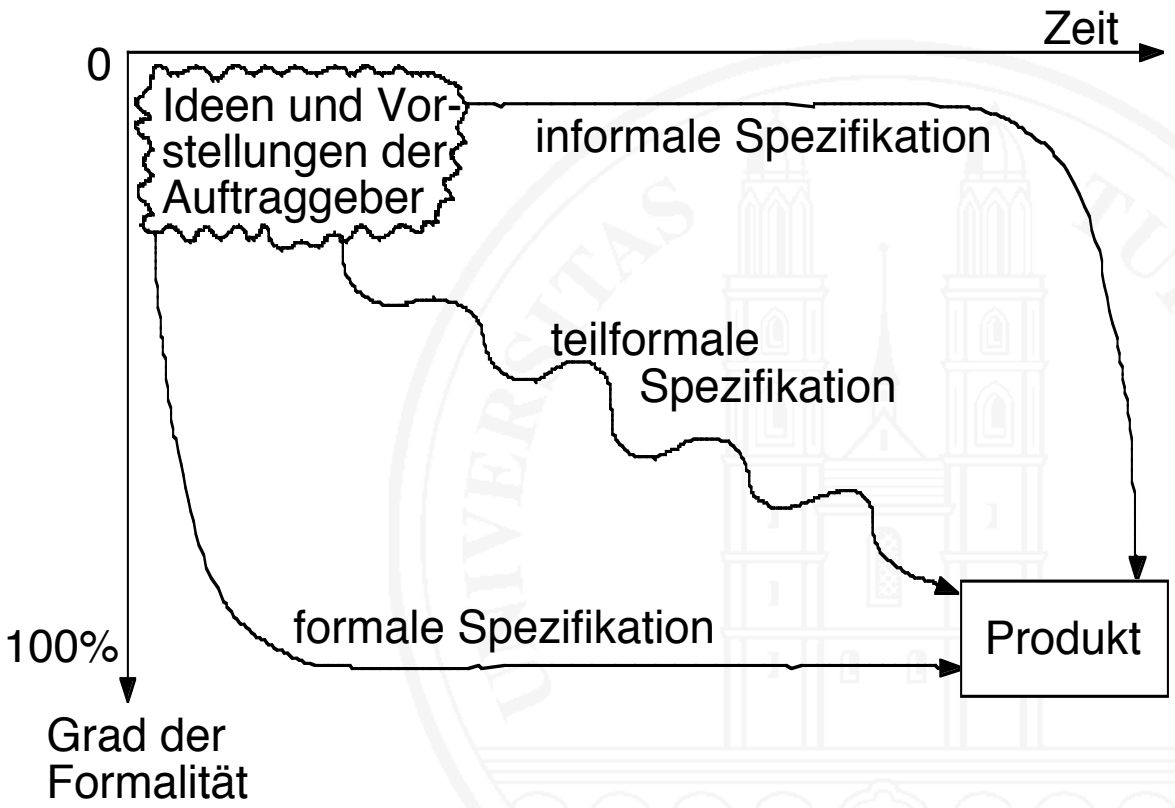
+ Anschauliches Modell der Problemstellung; leicht verstehbar und nachvollziehbar
+ Ermöglicht die Zerlegung der Gesamtaufgabe in kleinere, besser überschaubare Teilaufgaben
+ Kombination unterschiedlich stark formalisierter Teile möglich
+ Das Modell ist idealisierte Lösung: Tatsächliche Lösung oft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

Beispiel einer 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 iederzeit 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.»

Unklarheit: Reihenfolge von Auswahl und Bezahlung?

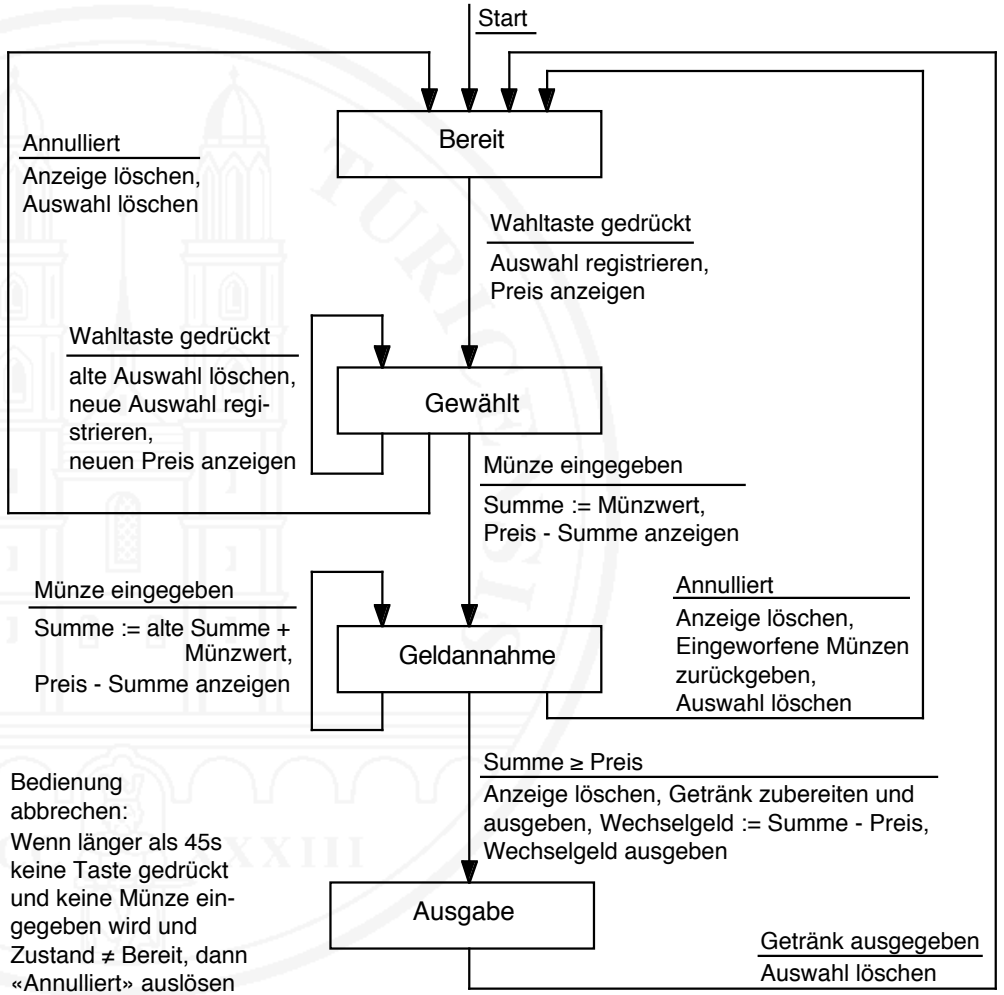
Fehler: Was ist bei Betragsgleichheit?

Mehrdeutigkeit: Ist «und» oder «oder» gemeint?

Unklarheit: Abbruch während der Ausgabe des Getränks?

Widerspruch: Annullierung wird ausgeschlossen

Beispiel einer teilformalen Spezifikation



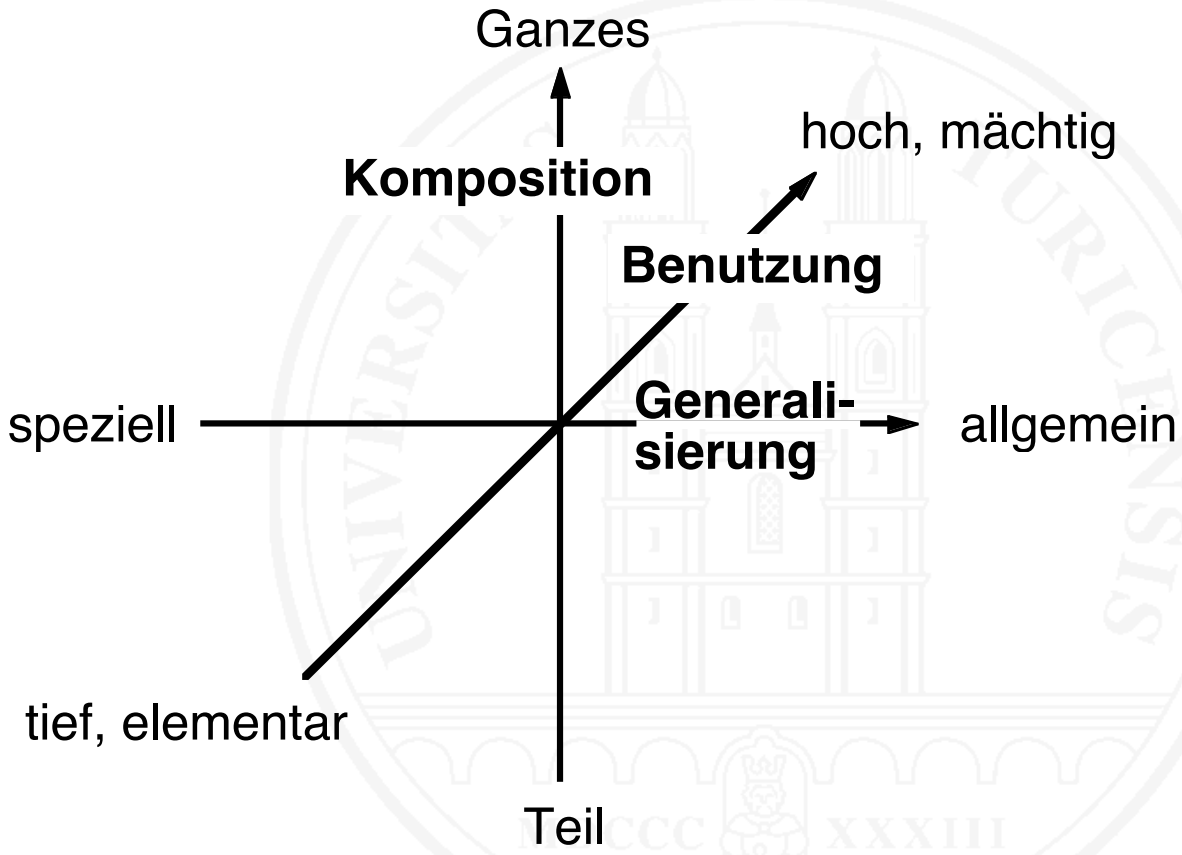
Beispiel einer formalen Spezifikation

Sei m_i die i-te eingegebene Münze, $|m_i|$ der Wert dieser Münze, P der geforderte Preis und G die Funktion, welche die Getränkezubereitung auslöst. Dann muss gelten:

$\forall n \in \mathbb{N} \; \forall m_i, 1 \leq i \leq n$

$$\left[\sum_{i=1}^{n} |m_i| \geq P \wedge \sum_{i=1}^{n-1} |m_i| < P \right] \leftrightarrow \left[G(m_1 \circ m_2 \circ \ldots \circ m_n) \wedge \neg G(m_1 \circ m_2 \circ \ldots \circ m_{n-1}) \right]$$

Gliederungs- und Abstraktionsmittel



siehe Vorlesung Informatik II, Teil Modellierung, Kapitel 8

Ausgewählte Spezifikationsmethoden

- Natürlichsprachige Spezifikation
- Algebraische Spezifikation
- Datenflussorientierte Spezifikation (Strukturierte Analyse)
- Verhaltensspezifikation mit Automaten
- Objektorientierte Spezifikation
- Spezifikation mit Anwendungsfällen

Natürlichsprachige Spezifikation

- Nur wenige spezifische Methoden,
  hauptsächlich zur Erkennung bzw. Vermeidung von
  - unvollständigen Sätzen
  - Mehrdeutigkeiten
  - vagen Aussagen
- Strukturierungshilfsmittel:
  - Nummerierung der Anforderungen
  - Gliederungsschemata

Algebraische Spezifikation – 1

- 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)

Algebraische Spezifikation – 2

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 Stack zu speichern sind.

```
TYPE Stack
FUNCTIONS
new:    ()              →  Stack;  -- neuen (leeren) Stack 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 Stack kein Element enthält
full:   Stack           →  bool;   -- wahr, wenn Stack voll ist
```

Algebraische Spezifikation – 3

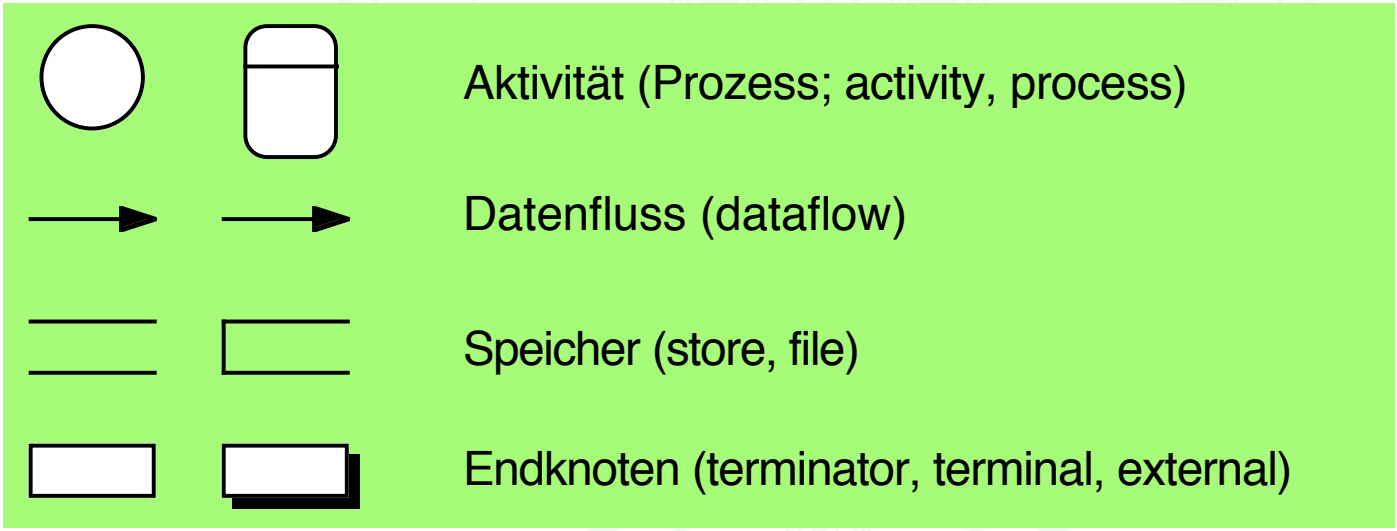
AXIOMS

$\forall s \in Stack, e \in elem$

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

Strukturierte Analyse – 1

- Teilformale, konstruktive Spezifikationsmethode
- Modellierung durch eine Hierarchie von *Datenflussdiagrammen*
- Basiert auf dem Prinzip der *datengesteuerten Verarbeitung*

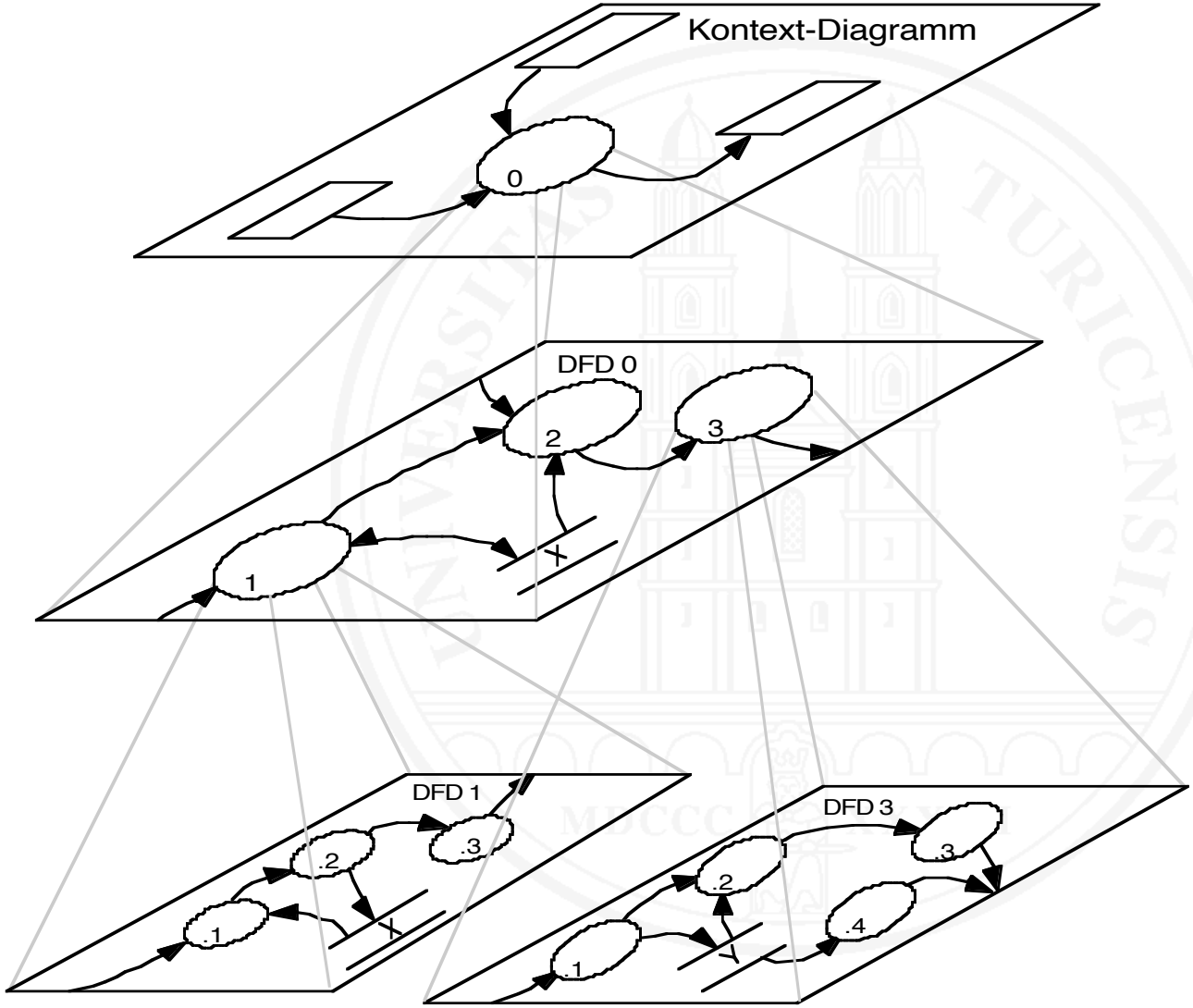


Strukturierte Analyse – 2

- Datenflussdiagramme sind Übersichtsmodelle
- Zur Präzisierung müssen definiert werden:
  - die Namen aller Datenflüsse und Speicher → Datenlexikon
  - die Funktionalität jeder nicht hierarchisch zerlegten Aktivität → Mini-Spezifikationen

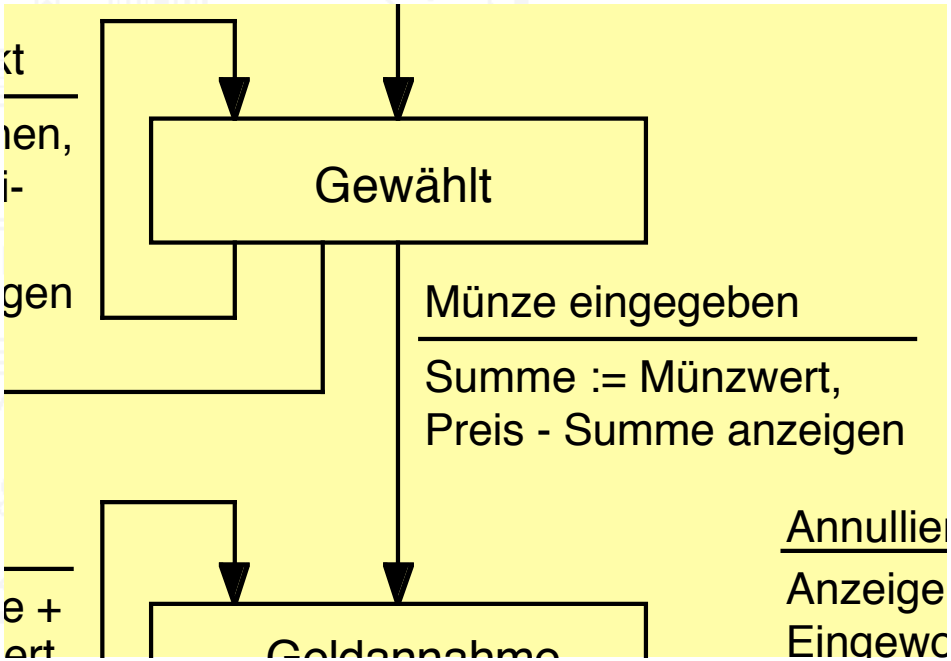
- Datenflussdiagramme eines Systems können hierarchisch in Schichten angeordnet werden
- Jede Ebene fasst die Datenflussdiagramme der darunterliegenden Ebenen zu je einer Aktivität zusammen

Strukturierte Analyse – 3



Verhaltensspezifikation mit Automaten

- Verhaltensspezifikationen mit Automaten sind teilformale, konstruktive Modelle
- Anforderungen werden als eine Menge von Zuständen, Zustands-übergängen, auslösenden Ereignissen und ausgelösten Aktionen beschrieben
- Das Modell spezifiziert die Reaktionen des Systems auf eine gegebene Folge äußerer Ereignisse



Objektorientierte Spezifikation – 1: Grundlagen

- 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: Objekte

Objekt (object) – Ein individuell erkennbares, von anderen Objekten eindeutig unterscheidbares Element der Realität

Beispiel: Die konkrete Person Eva Müller, 36 Jahre alt, Dr. oec. publ., Leiterin Fertigung in der Firma AGP, verheiratet, ein Kind, ...

Objekteigenschaften:

- Attribute und Werte dazu
  Eva Müllers Geschlecht: weiblich
- Beziehungen zu anderen Objekten
  Eva Müller leitet die Abteilung Fertigung
- Operationen auf Objekten
  EvaMüller befördern

Objektorientierte Spezifikation – 3: Klassen

Klasse (class) – eine eindeutig benannte Einheit, welche eine Menge gleichartiger Objekte beschreibt

Beschreibt den Aufbau, die Bearbeitungsmöglichkeiten und das mögliche Verhalten von Objekten dieser Klasse

Beispiel: Die Klasse Mitarbeiter mit den Attributen Name, Vorname, Geschlecht, Titel, Zivilstand, Anzahl Kinder, ... und den Beziehungen arbeitet in und leitet zur Klasse Abteilung beschreibt Personen der Art von Eva Müller

Klassendefinition

- Definition der Attribute der Klasse und der Wertebereiche dazu (lokale Merkmale)
- Definition der Beziehungen zu anderen Klassen (referenzierte Merkmale)
- Definition der Operationen (auf Objekten der Klasse / auf der Klasse)

Objektorientierte Spezifikation – 4: Klassen

- Zwei Sichten
  - Intensional: Klasse ist ein Typ.
  - Extensional: Klasse ist eine Menge von Objekten
- Häufig werden beide Sichten miteinander vermischt

- Klassen stehen in Beziehung zueinander
  - Assoziation (mit Kardinalitäten)
  - Benutzung
  - Vererbung

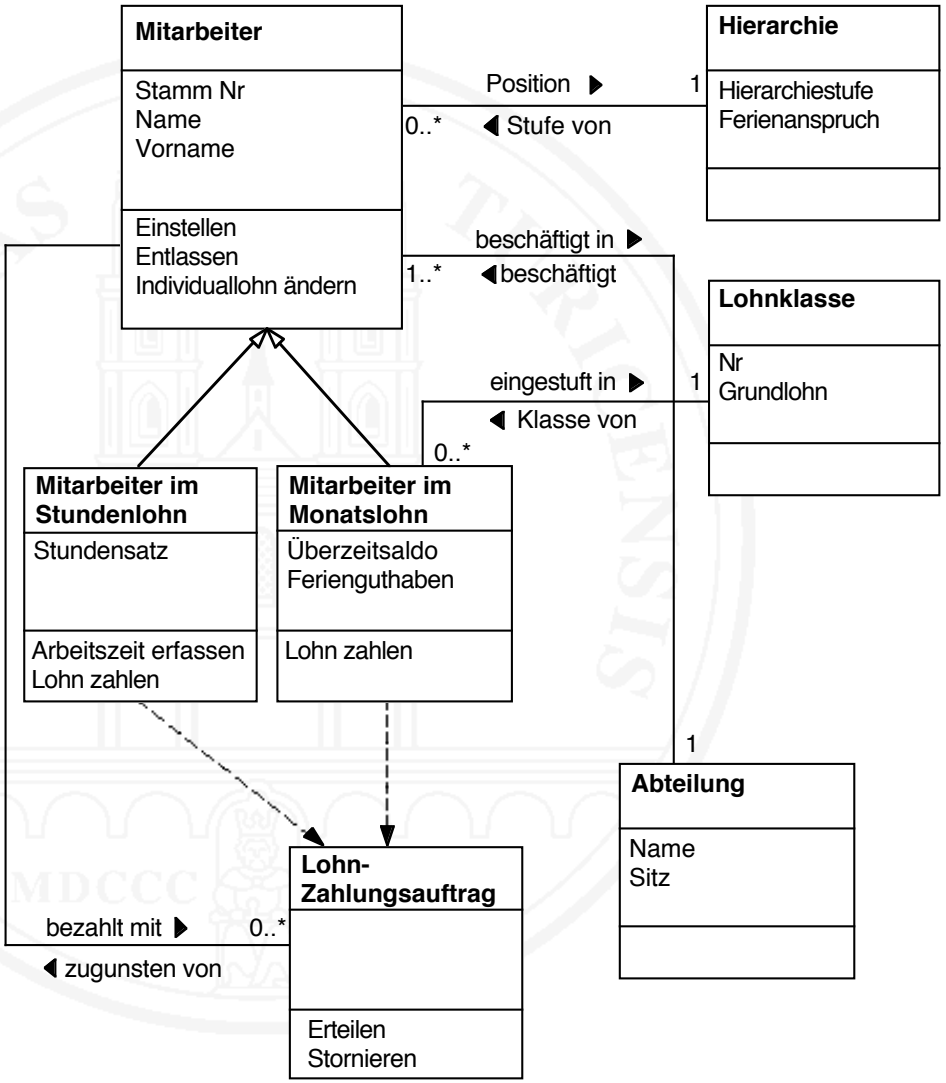
Objektorientierte Spezifikation – 5: Klassenbeschreibung

KLASSE Mitarbeiter im Monatslohn
UNTERKLASSE von Mitarbeiter
ATTRIBUTE (Name, Kardinalität, Wertebereich)
    Leistungslohnanteil [1..1] CHF
    Überzeitsaldo[1..1] Stunde
    Ferienguthaben [1..1] Tag
    ...
BEZIEHUNGEN (Name, Kardinalität, mit Klasse)
    eingestuft in [1,1] Lohnklasse
OPERATIONEN
    Lohn zahlen
        Voraussetzung: Mitarbeiter ist aktiv
        Ergebniszusicherung: Zahlungsauftrag zugunsten des Mitarbeiters ist erteilt
                mit Grundlohn aus Lohnklasse und Leistungslohnanteil
BENUTZT (Klasse.Operation)
    Zahlungsauftrag.Erteilen

Objektorientierte Spezifikation – 6: Klassenmodell

Beispiel eines Klassenmodells...

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

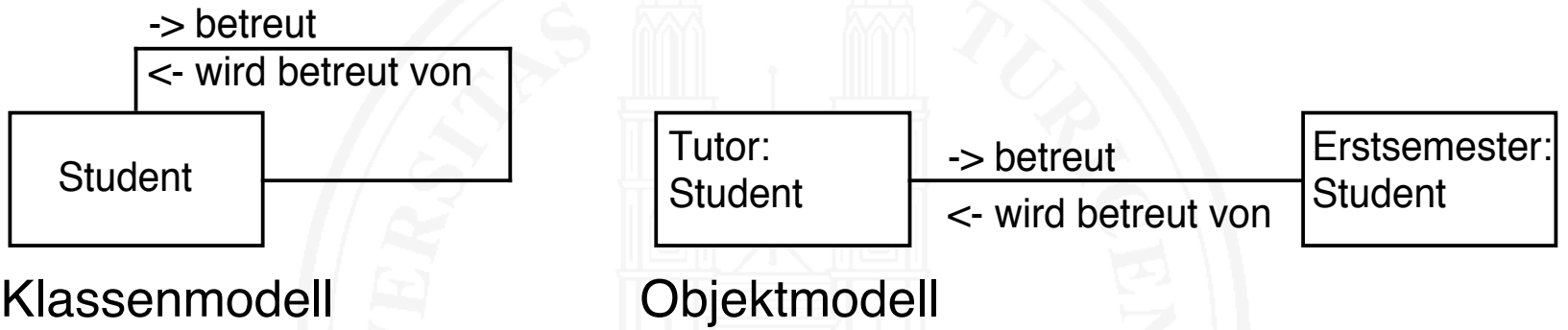


Objektorientierte Spezifikation – 7: Objektmodelle

- Alternative oder Ergänzung zu Klassenmodellen
- Modellierung abstrakter Objekte
- Stehen als Muster bzw. Repräsentanten für konkrete Objekte
- In den meisten heute verwendeten Ansätzen
  - nicht verwendet oder
  - nur als Ergänzung zu Klassendefinition verwendet (Modellierung des Arbeitskontextes)
- Objektmodelle anstelle von Klassenmodellen: große Vorteile, wenn
  - verschiedene Objekte der gleichen Klasse zu modellieren sind
  - ein Modell hierarchisch in Komponenten zerlegt werden soll
  - Aber: Modellierung von Assoziationen und Vererbungsbeziehungen umständlicher

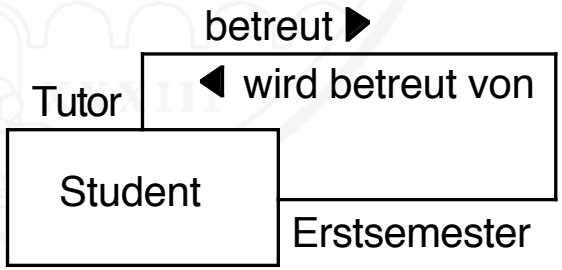
Objektorientierte Spezifikation – 8: Objektmodelle

- Situation, in der ein Objektmodell überlegen ist:



Klassenmodell Objektmodell

- ⇨ Behelfskonstruktion in vielen Klassenmodellen z. B. auch UML: Beschriftung der Beziehungen mit Rollen:



Spezifikation mit Anwendungsfällen – 1

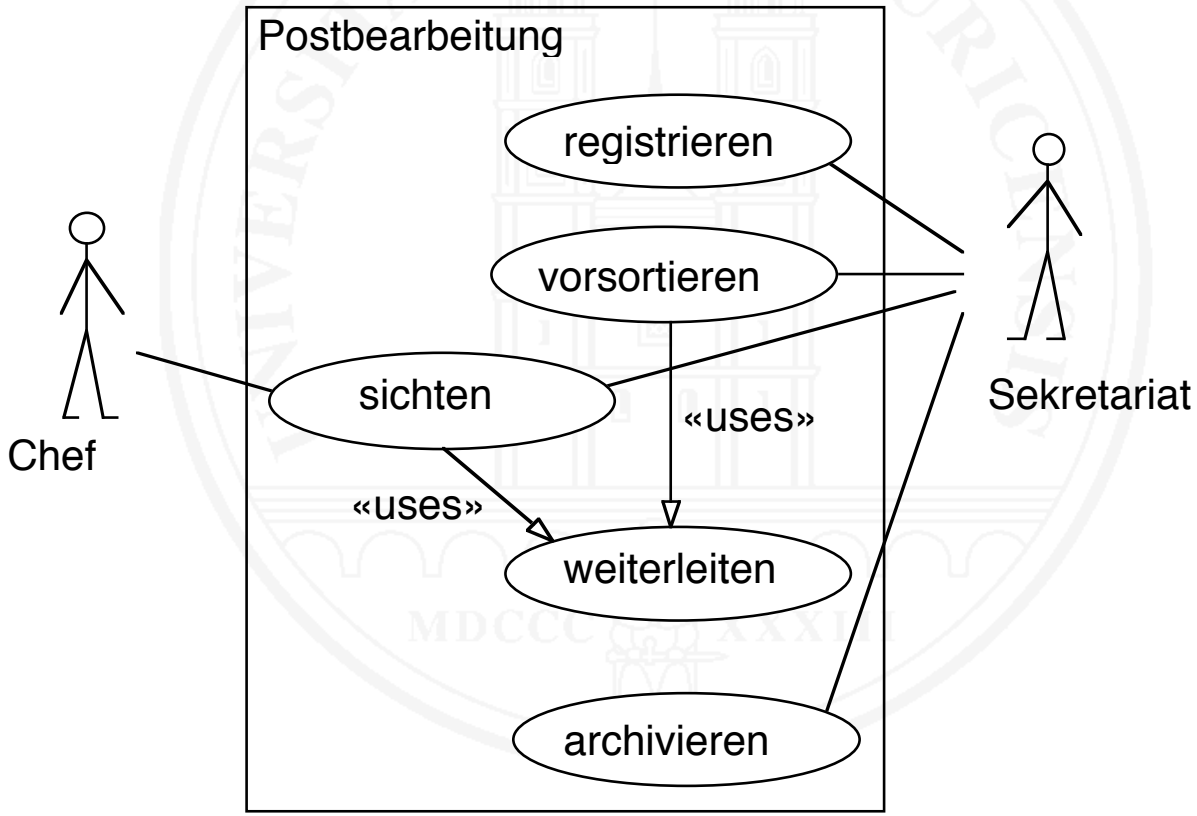
- Klassenmodelle modellieren nicht
  - den Systemkontext
  - die Interaktionen zwischen System und Umgebung
- Diese sind aber wichtig. Darum
- ➪ Ergänzung durch Anwendungsfallmodelle

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.

- Modellierung der einzelnen Anwendungsfälle
  - informal durch Text, z.T. formatiert durch Schlüsselworte
  - teilformal, z.B. mit Zustandsautomaten

Spezifikation mit Anwendungsfällen – 2

- Modellierung des Systemkontextes und der Menge aller Anwendungsfälle: Anwendungsfalldiagramm



Prüfen von Anforderungen – 1: 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

Prüfen von Anforderungen – 2: Organisation

Beteiligte:

- Informatiker
- Kunde(n)
- beide gemeinsam

} je nach Verfahren

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)

Prüfen von Anforderungen – 3: Prüfverfahren

- Reviews
- Prüf- und Analysemittel in Werkzeugen
- Simulation/Animation
- Prototypen

- 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

Prüfen von Anforderungen – 4: Prüfverfahren

- Prüf- und Analysemittel in Werkzeugen
  - Einsatz bei werkzeuggestützter Erstellung der Spezifikation
  - Auffinden von Lücken und Widersprüchen
- Simulation/Animation
  - Untersuchung der Adäquatheit des Systemverhaltens
  - Dynamisches Verhalten des spezifizierten Systems wird durch Simulator ausgeführt und/oder durch Animator visualisiert

Prüfen von Anforderungen – 5: Prüfverfahren

- Prototyp
  - Beurteilung der Adäquatheit / praktischen Brauchbarkeit des spezifizierten Systems anhand der im Prototyp realisierten Systemteile
  - Erprobung eines Systems in der geplanten *Einsatzumgebung*
  - Modell für die weitere Entwicklung oder für das zu schaffende Produkt
  - Mächtigstes (aber auch aufwendigstes) Mittel zur Beurteilung der Adäquatheit