

# Anforderungsanalyse

Echtzeitsystemlabor – Vorlesung/Übung

---

**Peter Ulbrich**  
**Fabian Scheler**  
**Wolfgang Schröder-Preikschat**

Lehrstuhl für Informatik 4  
Verteilte Systeme und Betriebssysteme  
Friedrich-Alexander Universität Erlangen-Nürnberg

<http://www4.cs.fau.de/~{scheler,ulbrich,wosch}>  
[{scheler,ulbrich,wosch}@cs.fau.de](mailto:{scheler,ulbrich,wosch}@cs.fau.de)



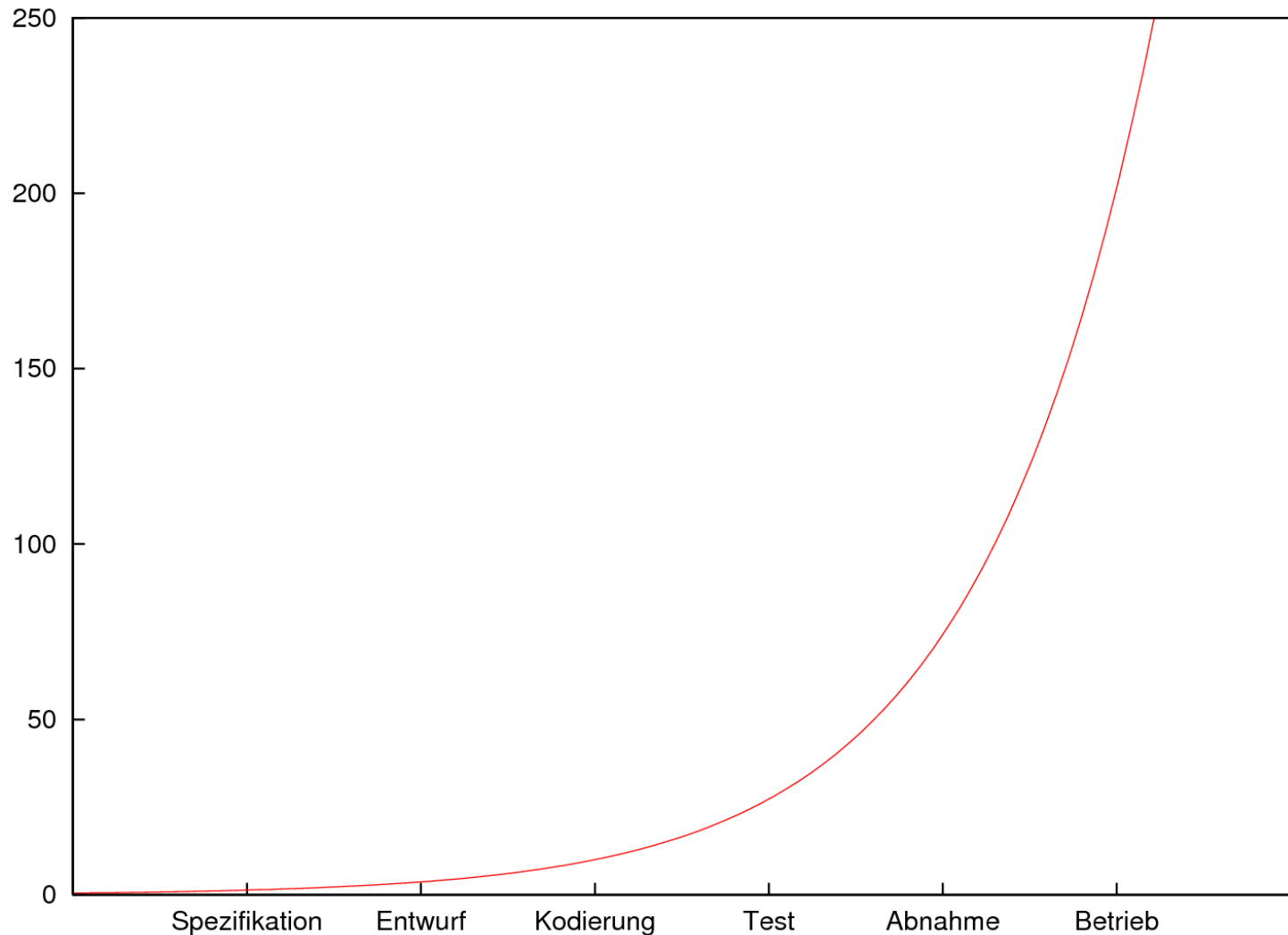
# Übersicht

---

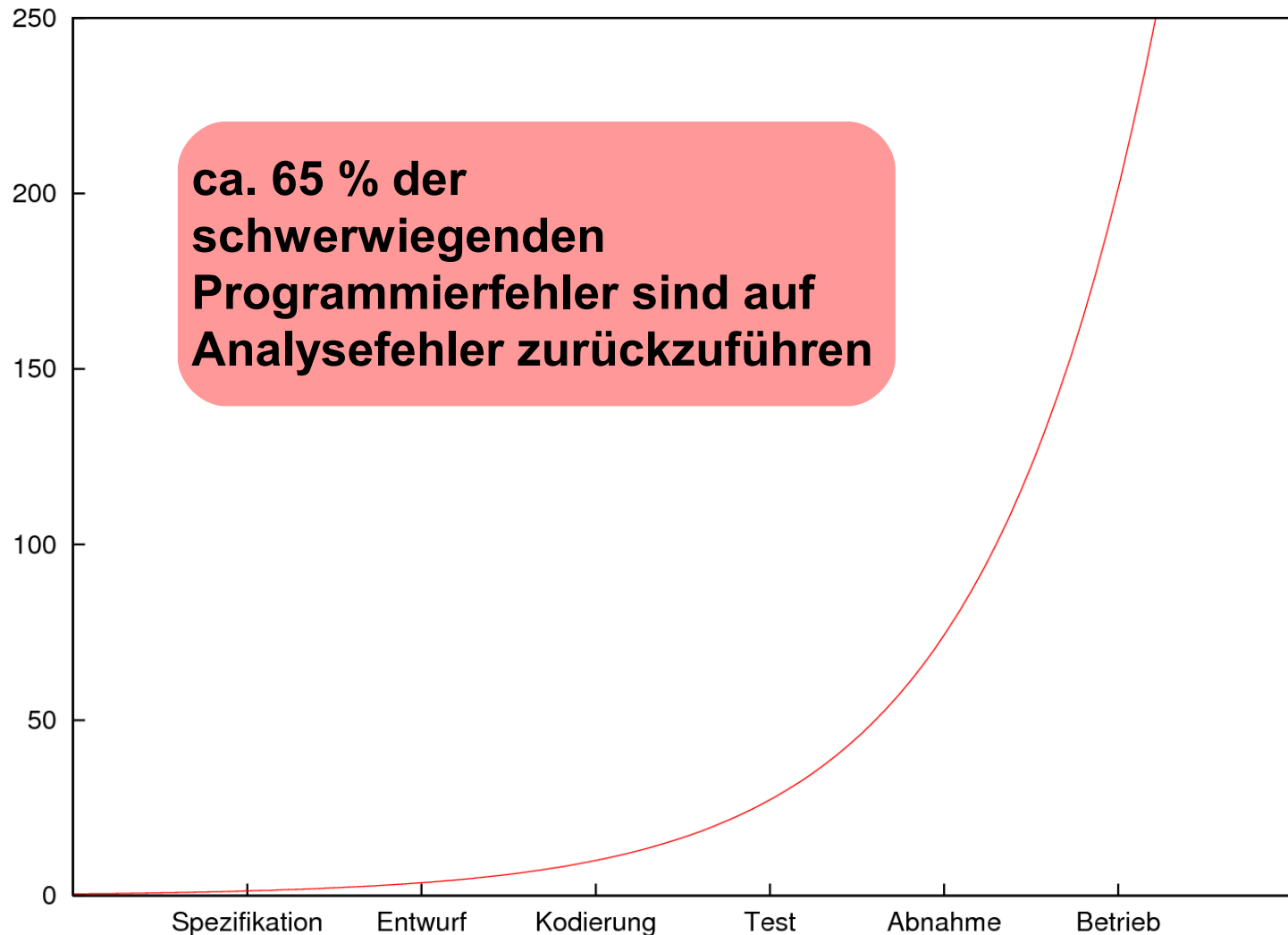
- Einleitung
- Aufgabenfelder
- Darstellungsmethoden
- Zusammenfassung



# Wozu? - Relative Kosten von Fehlern



# Wozu? - Relative Kosten von Fehlern



# Analyse der Problemstellung

---

- methodisch gestütztes Aufstellen von Anforderungen
- **Anforderung** (engl. *requirements*)
  - Aussage über eine zu erbringende Leistung
    - eines Produkts oder eines Systems
  - eine Eigenschaft, die erfüllt sein muss,
    - damit ein bestimmter Vorgang gelingen kann
  - ein Leistungsmerkmal (nicht nur) von Software
- Zusammenfassung im **Lasten-/Pflichtenheft**
  - als Bestandteil eines zu erstellenden Anforderungsdokuments, das
    - die durch das System zu lösende Aufgabe beschreibt
    - die im Projekt zu erreichenden Ziele definiert
    - den Benutzerkreis des zu entwickelnden Systems festlegt
  - ... in **Zusammenarbeit mit dem Kunden**



# Anforderungen → Lasten & Pflichten

---

- **Lastenheft** (Anforderungsspezifikation)
  - beschreibt unmittelbare Anforderungen, Erwartungen, Wünsche
  - legt fest, **was** und **wofür** etwas gemacht werden soll
- **Pflichtenheft**  
(Sollkonzept, Fachfeinkonzept, fachliche Spezifikation)
  - detaillierte Beschreibung einer zu erfüllenden Leistung
    - liegt am Ende als schwarzer Kasten (engl. *Black Box*) vor
    - enthält i.d.R. nicht die Problemlösung (**keine** Implementierung)
    - präzise, vollständige, nachvollziehbare Inhalte
  - gibt an, **wie** und **womit** etwas realisiert werden soll
    - verknüpft mit techn. Festlegungen der Betriebs-/Wartungsumgebung



# Anforderungen → Lasten & Pflichten

- **Lastenheft** (Anforderungsspezifikation)
  - beschreibt unmittelbare Anforderungen, Erwartungen, Wünsche
  - legt fest, **was** und **wofür** etwas gemacht werden soll
- **Pflichtenheft**  
(Sollkonzept, Fachfeinkonzept, fachliche Spezifikation)
  - detaillierte Beschreibung einer zu erfüllenden Leistung
    - liegt am Ende als schwarzer Kasten (engl. *Black Box*) vor
    - enthält i.d.R. nicht die Problemlösung (**keine** Implementierung)
    - präzise, vollständige, nachvollziehbare Inhalte
  - gibt an, **wie** und **womit** etwas realisiert werden soll
    - verknüpft mit techn. Festlegungen der Betriebs-/Wartungsumgebung

Nach DIN 69905 enthält das **Pflichtenheft** die vom **Auftragnehmer** erarbeiteten **Realisierungsvorgaben**, die sich aus der Umsetzung des vom **Auftraggeber** vorgegebenen **Lastenheftes** ergeben haben.



# Gliederung: Lasten- & Pflichtenheft

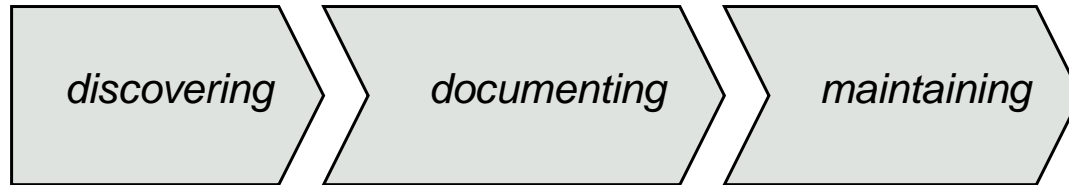
---

1. Allgemeines
  1. Einführung
  2. Referenzen
2. Systembeschreibung
  1. Funktionelles Zusammenwirken
  2. Funktionelle Arbeitsweise
  3. Aufteilung in Hard-/Software
3. Softwareanforderung
  1. Daten: Name, Typ, Struktur, Wertevorrat, Dimension, Genauigkeit, Zeitbedingungen, Bedeutung
  2. Funktionen: Ergebnis, Bedingungen, Initialisierung, Sonderfälle, Wiederholfrequenz/Durchlaufzeit, Bedeutung
4. Sonstiges: Programmiersprache, Verfahrensvorschriften

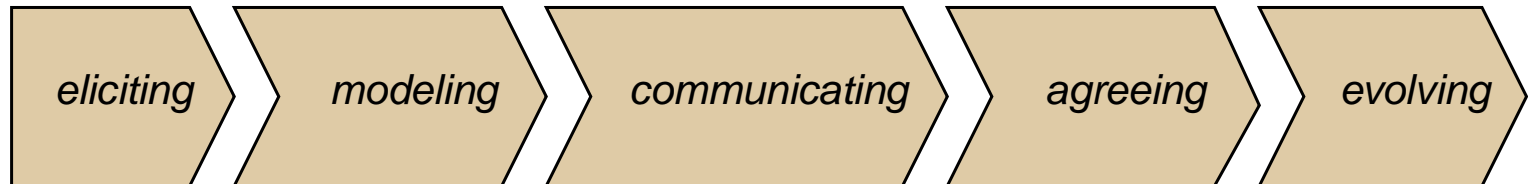


# Einzel Schritte

---



Sommerville & Sawyer



Nuseiheb & Easterbrook





# Anforderungsanalyse → Anforderungsdefinition

---

- **Anforderungstechnik** (engl. *req. engineering*, RE)
  - wichtige Voraussetzung zur Ermittlung von Anforderungen
    - Interessenvertreter identifizieren
    - d.h. die richtigen zu befragenden Institutionen/Personen ...
  - oft auch als Synonym zu *Anforderungsanalyse*
- **Anforderungspflege** (engl. *req. managment*, RM)
  - umfasst die Anforderungsanalyse und geht darüber hinaus
    - Maßnahmen zur Anforderungssteuerung, -kontrolle und -verwaltung
    - d.h. Risiko-, Änderungs- und Umsetzungsmanagement
  - elementare Prozess der Software- und Systemreifegrad-Modelle
    - **CMMI** – Capability Maturity Model Integration
    - **SPICE** – Software Process Improvement and Capability Determination
  - auch bekannt als **Software Requirements Specification** (SRS)



# Abgrenzung: RE vs. RM

---

## ■ Req. Engineering

- Erfassung
- Analyse
- Prüfung
- Abstimmung

## ■ Req. Management

- Strukturierung
- Bewertung
- Verfolgung
- Berichtswesen



# Abgrenzung: RE vs. RM

---

## ■ Req. Engineering

- Erfassung
- Analyse
- Prüfung
- Abstimmung

## ■ **Generierung** von Anforderungen

## ■ Req. Management

- Strukturierung
- Bewertung
- Verfolgung
- Berichtswesen

## ■ **Verwaltung** von Anforderungen



# Einzelsschritte

---

## 1. Anforderungserhebung

- Kriterien zur **Aufnahme** von Anforderungen
  - vollständig, eindeutig definiert/abgegrenzt, verständlich
  - atomar, identifizierbar, dokumentiert, notwendig
  - nachprüfbar, rück- und vorwärtsverfolgbar
- abschließende **Erfassung** der Anforderungen im Lastenheft

## 2. Anforderungsdefinition

- Kriterien zur **Strukturierung** der Anforderung
  - abhängig, zusammengehörig, rollenbezogen
  - funktional/nichtfunktional, fachlich/technisch motiviert
- abschließende **Abstimmung** zwischen Kunde und Entwickler

## 3. Anforderungsbewertung → **Prüfung** und **Bewertung**

- Qualitätssicherung der Anforderungen
  - korrekt, machbar, notwendig, priorisiert, nutzbar, benutzerfreundlich
- Ergebnis dieses Schritts ist Basis für das Pflichtenheft



# Qualitätsmerkmale von Anforderungen

---

## ■ Adäquatheit

- beschreiben, was der Auftraggeber fordert, was benötigt wird

## ■ Vollständigkeit

- alles beschreiben, was der Auftraggeber fordert, was benötigt wird

## ■ Widerspruchsfreiheit

- ansonsten ist die Spezifikation nicht realisierbar

## ■ Verständlichkeit

- für den Auftraggeber und den Auftragnehmer

## ■ Eindeutigkeit

- um Fehler durch Fehlinterpretationen zu vermeiden

## ■ Prüfbarkeit

- ob das erstellte System den Anforderungen entspricht



# Typen von Anforderungen (nach SRS)

---

1. funktionale Anforderungen
    - Beschreibung des kompletten, deterministischen Systemverhaltens
  2. externe Schnittstellen
  3. Performanz (statisch/dynamisch)
  4. logische Datenbasis
    - Nutzungsfrequenz, Zugriffsfähigkeiten, Daten inkl. Beziehungen
  5. Entwurfseinschränkungen
    - Einhalten von Normen, Systemattributen (von Software)
  6. Systemattribute von Software
    - Zuverlässigkeit, Verfügbarkeit, Sicherheit, Wartbarkeit, Übertragbarkeit
- Anforderungen 2. - 6. gelten als **nicht funktional**



# Typen von Anforderungen - Beispiele

---

- Name des Elements/Postens
- Gegenstandsbeschreibung
- Quelle der Eingabe und Ziel der Ausgabe
- Gültigkeitsbereich, Genauigkeit, Abweichung
- Maßeinheit
- Zeitvorgabe
- Beziehung zu anderen Ein-/Ausgaben
- Bildschirmformate/-organisation
- Fensterformate/-organisation
- Daten- und Befehlsformate



# Herausfinden

---

- ... was der Kunde will bzw. was machbar ist
- **Erhebung** (engl. *elicitation*)
  - Identifizierung von Anforderungen, Auflagen und Einschränkungen
    - Fragebögen, offene Interviews, Besprechungen
  - Wiederverwendung von Anforderungen aus früheren Projekten
- **Abstimmung** (engl. *negotiation*)
  - Auflösung bestehender Konflikte ...
    - zwischen Fähigkeiten und Einschränkungen
    - zwischen Anforderungen und Betriebsmitteln (Ressourcen)
    - wegen inkompatibler Merkmale verschiedener Interessenvertreter
  - Verhandlung mit den Interessenvertretern (Auftraggebern)
    - Konsensfindung, Kompromisswege/-lösungen herausarbeiten



# Formulieren

---

- ... des Problems und ggf. auch einer Lösungsidee
- **Analyse** (engl. *analysis*)
  - Grenze des Systems und Interaktion mit der Umgebung erläutern
    - ggf. verschiedene Sichten (engl. *viewpoint*) einnehmen
    - z.B. unterschiedliche Entwicklerrollen oder Beschreibungstechniken
  - widersprüchliche Anforderungen identifizieren und ggf. auflösen
- **Spezifikation** (engl. *specification*)
  - vollständige Menge zusammenhängender Anforderungen gestalten
  - Subsysteme/Komponenten definieren und Anforderungen zuordnen
- **Modellierung** (engl. *modelling*)
  - Systemeigenschaften durch **konzeptionelle Modelle** untersuchen
    - Daten-/Kontrollfluss-, Zustands-, Objekt-, Anwendungsfallmodelle
  - die operative Umgebung samt Daten und Kommunikation verstehen



# Organisieren

---

- ... um die Problemkomplexität zu beherrschen
- **Dokumentation** (engl. *documenting*) → **Anforderungsdok.**
  - die Menge aller beschriebenen Anforderungen zusammenstellen
  - Lastenheft erzeugen, das später ins Pflichtenheft überführt wird
- **Strukturierung** (engl. *structuring*)
  - Anforderungen nach versch. Kriterien klassifizieren
    - Gruppierung nach z.B. Priorität (bei der Erfüllung der Gesamtziele), Herkunft, Gültigkeitsbereich, Stabilität usw. vornehmen
    - in funktional und nicht-funktionale Anforderungen einstufen
  - Attribute für jede Anforderung festlegen
    - Beschreibung, Grund, Urheber, Status, Akzeptanzkriterien, Implikationen, Abhängigkeiten, ...
    - dient u.a. auch der weiteren Gruppierung (s.o.)
  - den Anforderungen eindeutige Bezeichner zuordnen



# Hinterfragen

---

- ... ob das Problem richtig verstanden wurde
- **Validierung** (engl. *validating*)
  - sicherstellen, dass das beschriebene System die ursprüngliche Intention (des Auftraggebers) adäquat wiedergibt
    - ein sich zu verschiedenen Prozesszeitpunkten wiederholender Vorgang
  - das Anforderungsdokument untersuchen, in Form von Inspektionen oder formalen Besprechungen durch Gutachtergruppen
    - Fehler, irrtümliche Annahmen, unklar bestimmte Begriffe, Abweichungen von üblichen Vorgehensweisen identifizieren
    - Gutachter sind u.a. auch Beauftragte der Benutzer des Systems
  - ggf. einen Prototypen zeigen, um die ursprüngliche Intention (s.o.) mit der eigenen Interpretation des Systems zu konfrontieren
    - manchmal genügen bereits einfache Papierskizzen



# Vorbereiten

---

- ... für die Phasen der Systementwicklung danach
- weder **RE** noch **RM**
  - **Entwurf** (engl. *design*)
    - überlegen, wie die Anforderungen umgesetzt werden können
  - **Implementierung** (engl. *implementation*) und **Integration**
    - es tun, d.h. die Anforderungen umsetzen
  - **Verifikation** (engl. *verification*) und **Testen**
    - das Ergebnis mit dem ursprünglichen „Plan“ vergleichen
  - **Einführen** (engl. *rollout*)
    - das „Produkt“ ausliefern



# Spezifikationstechniken

---

- allgemeine Klassifikation bzw. Ansätze
- **formal** (engl. *formal*)
  - rigorose, mathematische Grundlage → **formale Notation**
- **informell** (engl. *informal*)
  - wenn die **Transkription** („Umkodierung“) in eine formale Notation mit zugeordneten Regeln nur eingeschränkt möglich ist
    - z.B. ein Ablaufdiagramm (engl. *flowchart*)
  - bestenfalls werden Anforderungsverletzungen/-konflikte sichtbar
- **halbförmlich** (engl. *semiformal*)
  - Ansätze, die formale und informelle Züge zeigen, z.B. UML:
    - das Zustandsdiagramm (engl. *statechart*) ist formal
    - andere Konzepte sind jedoch eher pseudomathematischer Natur

Echtzeitsysteme (mit strikt einzuhaltenden Anforderungen) **erfordern** eine **formale Begründung** der Leistungscharakteristiken von Anforderungen



# Natürliche Sprache

---

- weit verbreitete Technik
- Strukturierung durch Nummerierungs- und Gliederungsschemata
- Qualitätsverbesserung durch linguistische Methoden
  - Sätze mit Standardstruktur
  - kein Passiv
  - beschränkte Mengen von Verben mit festen Bedeutungen



# Natürliche Sprache

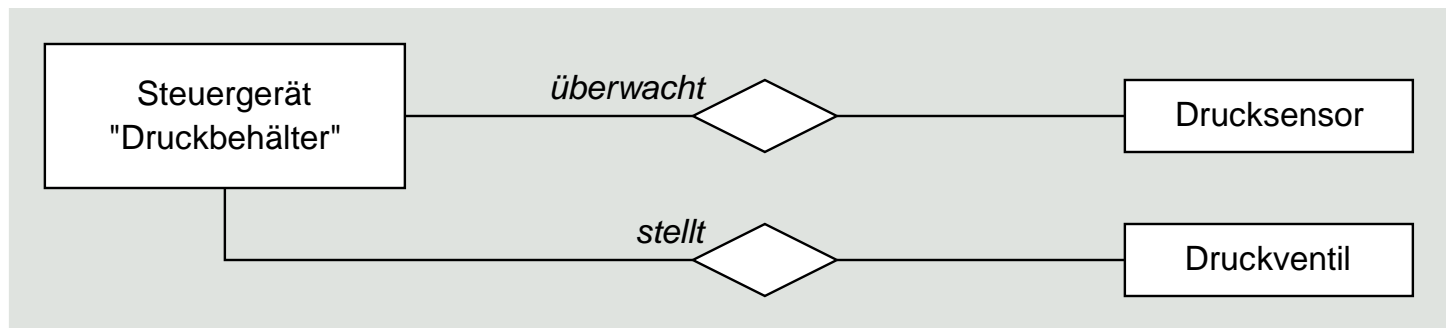
---

- weit verbreitete Technik
- Strukturierung durch Nummerierungs- und Gliederungsschemata
- Qualitätsverbesserung durch linguistische Methoden
  - Sätze mit Standardstruktur
  - kein Passiv
  - beschränkte Mengen von Verben mit festen Bedeutungen
- ✓ leicht zu lesen/schreiben, ausdrucksmächtig
- x unübersichtlich, fehleranfällig, mehrdeutig
- ungeeignet als alleiniges Beschreibungsmittel



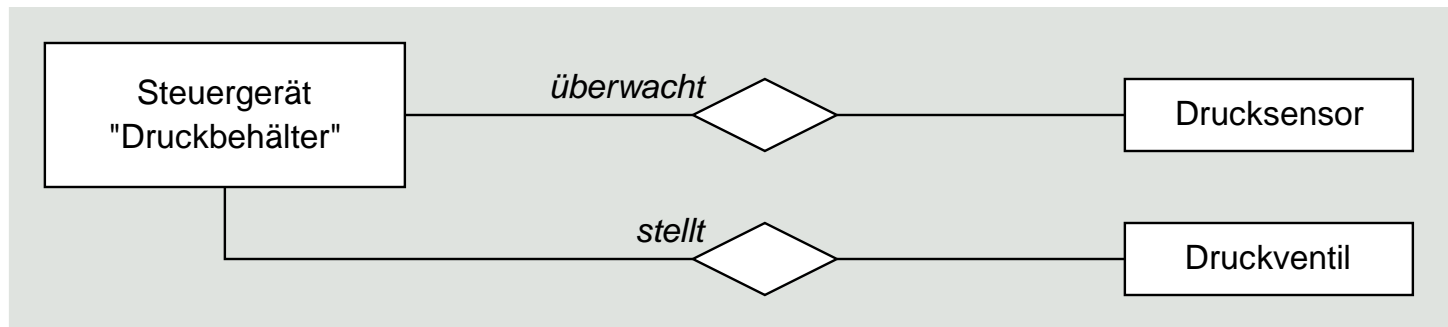
# Datenmodellierung

- Grundlage ist der **Entity-Relationship**-Ansatz
- modelliert werden Ausschnitte der Realität durch ...
  - Gegenstandstypen (engl. *entity types*)
  - Beziehungstypen (engl. *relation types*)
  - Attribute (engl. *attributes*)



# Datenmodellierung

- Grundlage ist der **Entity-Relationship**-Ansatz
- modelliert werden Ausschnitte der Realität durch ...
  - Gegenstandstypen (engl. *entity types*)
  - Beziehungstypen (engl. *relation types*)
  - Attribute (engl. *attributes*)

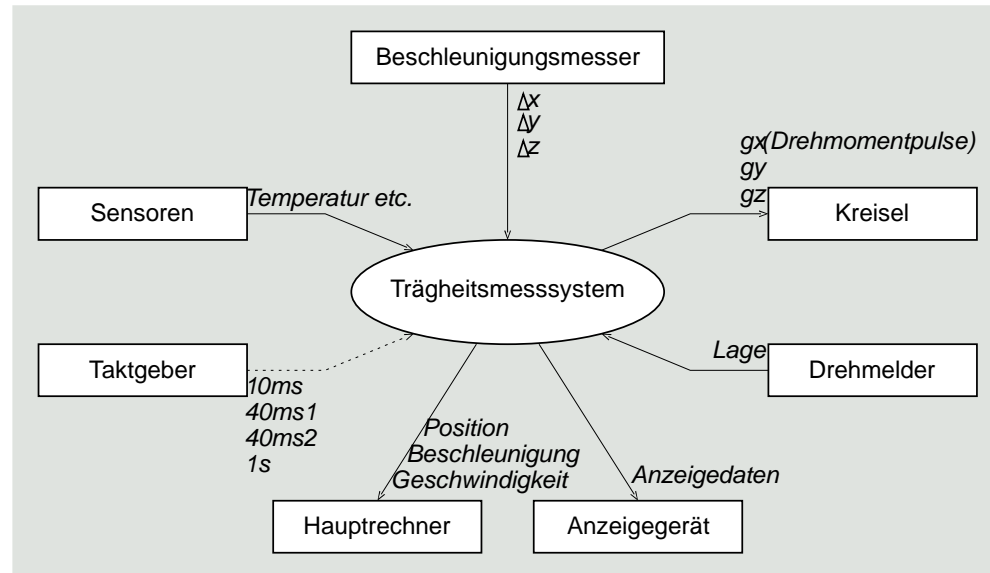


- ✓ vergleichsweise einfach und klar, ideal für Datenbanken
- ✗ weder Funktionalität noch Verhalten von Systemen  
keine Dekomposition bzw. Datenkapselung



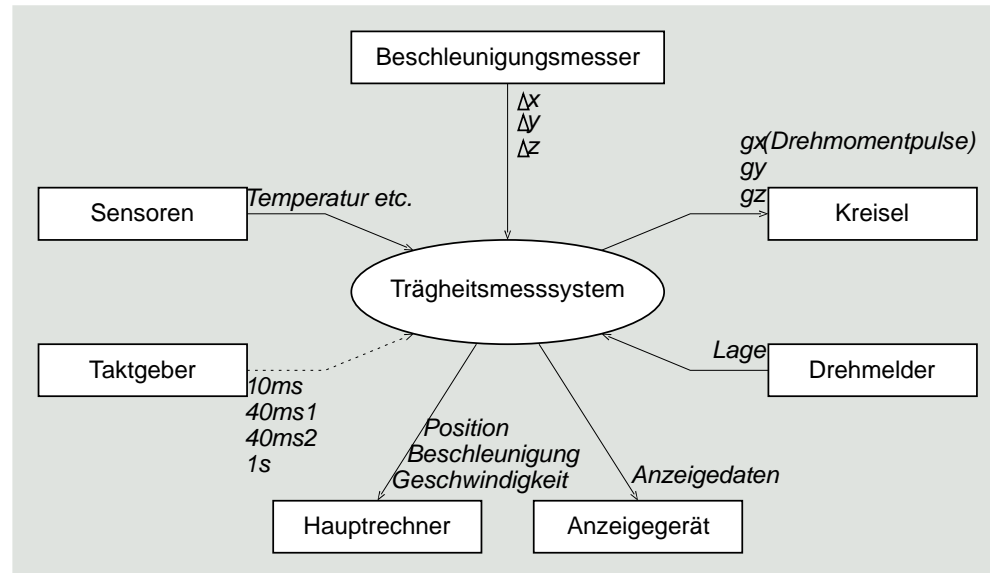
# Strukturierte Analyse

- Grundlage:  
Datenflussdiagramme
- Modellierung von  
Systemfunktionalität
- Beschreibung des  
Systemkontextes
  - Interaktion
  - Ein-/Ausgabe



# Strukturierte Analyse

- Grundlage:  
Datenflussdiagramme
- Modellierung von  
Systemfunktionalität
- Beschreibung des  
Systemkontextes
  - Interaktion
  - Ein-/Ausgabe



- ✓ vergleichsweise anschaulich, Dekomposition
- ✗ keine Lokalität, begrenzte Kapselungsfähigkeit, nicht-funkt. Eigenschaften nicht adäquat beschreibbar, „Strukturbruch“: Spezifikation ↔ Implementierung



# Objektorientierte Spezifikation

---

- Modellierung der statischen Struktur eines Systems unter Verwendung von Objekt- und Klassendiagrammen
- Objekte/Klassen beschreiben Daten, Funktionen und zeitliches Verhalten



# Objektorientierte Spezifikation

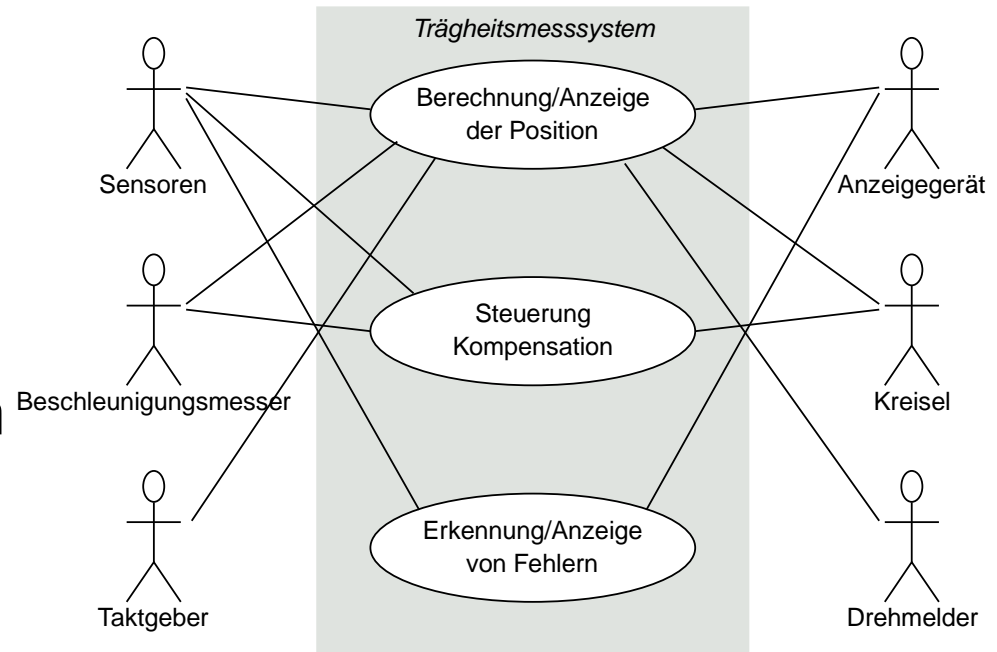
---

- Modellierung der statischen Struktur eines Systems unter Verwendung von Objekt- und Klassendiagrammen
- Objekte/Klassen beschreiben Daten, Funktionen und zeitliches Verhalten
  
- ✓ Beschreibung der Systemstruktur, Lokalität von Daten und Kapselung, motiviert strukturähnliche Implementierungen, Dekomposition
- x nicht-funkt. Anforderungen nicht adäquat beschreibbar



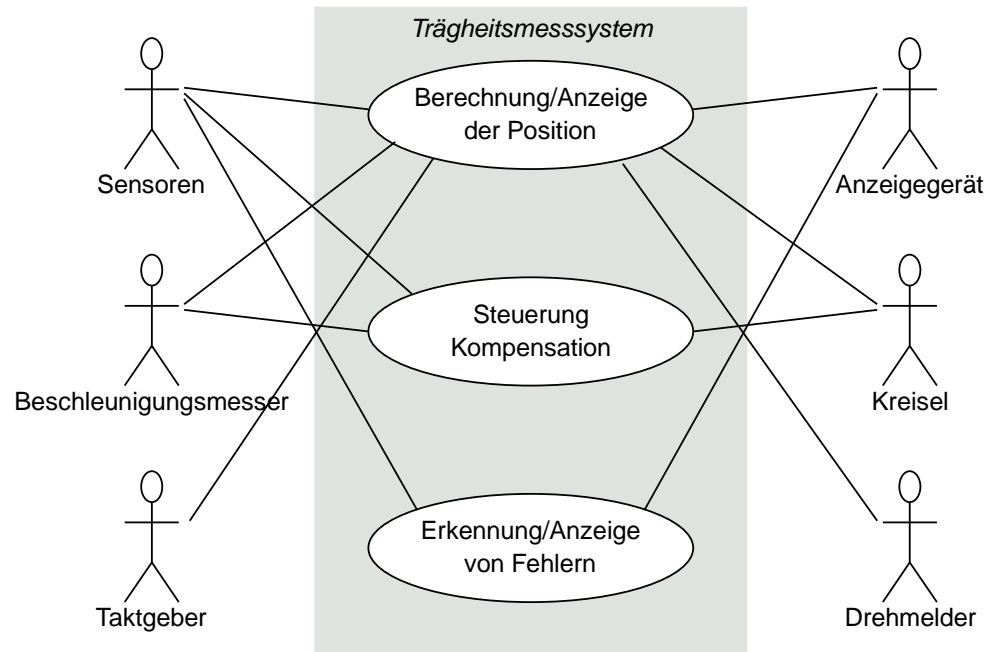
# Szenarien und Anwendungsfälle

- Modellierung der Interaktion zwischen System und Umwelt
  - d.h. Akteure
- Interaktionssequenzen entsprechen Szenarien
  - Anwendungsfall
  - engl. *use case*



# Szenarien und Anwendungsfälle

- Modellierung der Interaktion zwischen System und Umwelt
  - d.h. Akteure
- Interaktionssequenzen entsprechen Szenarien
  - Anwendungsfall
  - engl. *use case*



- ✓ leicht versteh- und prüfbar, modelliert Funktionalität aus Benutzersicht, Abgrenzung des Systems vom Kontext, Dekomposition
- ✗ keine Erfassung von Zusammenhängen/Abhängigkeiten von Szenarien, statische Struktur, keine Datenmodellierung



# Formale Methoden

---

- Grundlage bilden mathematische Formalismen
  - formal definierte Syntax und Semantik
- große theoretische Vorteile, praktisch selten zu finden
  - punktueller Einsatz: [sicherheitskritische Systeme](#)



# Formale Methoden

---

- Grundlage bilden mathematische Formalismen
  - formal definierte Syntax und Semantik
- große theoretische Vorteile, praktisch selten zu finden
  - punktueller Einsatz: **sicherheitskritische Systeme**
- ✓ Eindeutigkeit (formal definierte Semantik), Widerspruchsfreiheit, formal prüfbar, Nachweisbarkeit der Erfüllung von Anforderungen, Lösungsneutralität
- x aufwendige Erstellung, Prüfung der Adäquatheit schwierig, umfangreiche Spezifikation auch für Fachleute schwer verständlich



# Zusammenfassung

---

## ■ Einleitung

- Anforderung, Qualitätsmerkmal, Typen von Anforderungen
- Anforderungsanalyse (-technik) vs. Anforderungspflege
- Einzelschritte bzw. Prozess der Anforderungsanalyse
- Anforderungsspezifikation: Lasten- und Pflichtenheft

## ■ **Aufgabenfelder** → herausfinden, formulieren, organisieren, hinterfragen

- Erhebung, Abstimmung
- Analyse, Spezifikation, Modellierung
- Dokumentation, Strukturierung
- Validierung

## ■ **Darstellungsmethoden**

- formal, informell, halbformliche Spezifikationstechniken
- natürliche Sprachen, Datenmodellierung, strukturierte Analyse, objektorientierte Spezifikation, Anwendungsfälle. formale Methoden

