# IT Konzepte & Modelle

FS 2012

Flavio De Roni
Studiengang Wirtschaftsingenieur | Innovation 4. Semester

HSLU-T&A
24.05.2012

# Änderungsverzeichnis

| Version | Datum | Autor | Änderung | Status |
|---|---|---|---|---|
| 0.1 | 24.02.12 | fdr | Einführung | fertig |
| 0.2 | 25.02.12 | fdr | Requirements Engineering | in Arbeit |
| 0.3 | 03.02.12 | fdr | Bestandteile eines Grobkonzepts | fertig |
| 0.4 | 04.03.12 | fdr | Effiziente Textanalyse Block 1 | fertig |
| 0.4.1 | 09.03.12 | fdr | Requirements Engineering | fertig |
| 0.4.2 | 10.03.12 | fdr | Effiziente Textanalyse Block 2 | fertig |
| 0.4.3 | 17.03.12 | fdr | Effiziente Textanalyse Block 3 sowie Scope und Resultate | fertig |
| 0.5 | 30.03.12 | fdr | Modellierung | fertig |
| 0.6 | 05.04.12 | fdr | UML-Übersicht: Struktur erstellt | fertig |
| 0.6.1 | 06.04.12 | fdr | UML-Übersicht, ohne Notationselemente | fertig |
| 0.7 | 12.04.12 | fdr | Strukturierter Aufbau IT Konzept | fertig |
| 0.8 | 13.04.12 | fdr | Prozessmodellierung | in Arbeit |
| 0.8.1 | 18.04.12 | fdr | Prozessmodellierug | fertig |
| 0.9 | 27.04.12 | fdr | Funktionsmodellierung | in Arbeit |
| 0.9.1 | 06.05.12 | fdr | Funktions- und Datenmodellierung | fertig |
| 0.9.2 | 06.05.12 | fdr | Kontextdiagramm | fertig |
| 0.9.3 | 11.05.12 | fdr | Systemdesign | in Arbeit |
| 0.9.4 | 17.05.12 | fdr | Überarbeitung der Struktur | fertig |
| 1.0 | 25.05.12 | fdr | Architekturkonzepte | fertig |

# Inhalt

# Einführung

Konstruktionsfehler können katastrophale Auswirkungen haben – auch bei Software!

Konsequenzen:
- Systembau durch Konzept vorbereiten
- Zugrunde liegende Modelle genau recherchieren
- Konzept sorgfältig und nachvollziehbar ausarbeiten

Damit es nicht so weit kommt wie zum Beispiel hier:

Schiefer Turm von Pisa

| | |
|---|---|
| *Welchen Zweck sollte das Bauwerk erfüllen?* | Der Turm war als freistehender Glockenturm (Campanile) für den Dom in Pisa geplant. |
| *Welchem Faktor trug der Architekt in seinem Modell (Pläne) von dem Bauwerk nicht genügend Rechnung?* | lehmigen & sandigen Untergrund |
| *Welche waren die Auswirkungen dieses Modellierungsfehlers?* | Untergrund hat sich durch Gewicht abgesenkt, dadurch hat sich Turm geneigt. |

Tay-Brücke

| | |
|---|---|
| *Welchen Zweck sollte das Bauwerk erfüllen?* | Eisenbahnbrücke |
| *Welchem Faktor trug der Architekt in seinem Modell (Pläne) von dem Bauwerk nicht genügend Rechnung?* | Luftdruck (Kraft) des Seiten-Windes → notwendiger Sicherheitskoeffizient fehlte |
| *Welche waren die Auswirkungen dieses Modellierungsfehlers?* | Während eines furchtbaren Windsturmes brach am 29. nachts die große Eisenbahnbrücke über den Taystrom in Schottland zusammen, im Moment, als der Zug darüber fuhr. → viele Tote |

Ariane 5-Absturz

*Welche Fehlentscheidungen führten letztlich zur Katastrophe?*
*Welches war der entscheidende Konstruktionsfehler, ohne welchen der Flug wahrscheinlich trotz den anderen Fehlern hätte fortgesetzt werden können?*

Die Anforderungen der Ariane 5 wurden zu wenig genau spezifiziert, es wurde einfach bei den Anforderungen der Ariane 4 abgeschaut (Flugverhalten war z.B. gar keine Anforderung). Die Anforderungen wurden nicht sauber definiert. So war diese operative Funktion (Dauer 50 Sek ab Startsequenz), die den Fehler und damit das Unglück verursacht hatte, eigentlich gar nicht notwendig bei der Ariane 5 (nur

vor dem Start). Wurde nicht vollständig getestet, weil man angenommen hat, dass identisch mit Ariane 4.

Hauptfehler: ErrorHandling (Exception) hat zu Totalabschaltung geführt, anstelle korrekterweise Werte anzunehmen, um Flug weiterzuführen (Konzeptfehler)

- Fehler in der Software: Konversion einer 64-bit floating point zu einer 16-bit signed integer Zahl führte zu falschen Steuerbefehlen → wären die Tests sauber durchgeführt worden, so würde der Fehler aufgedeckt

# 1 Bestandteile eines Grobkonzepts

Ein Grobkonzept hat grundsätzlich folgenden Aufbau:

1. Ausgangslage
2. Ziel(e)
3. Scope
4. Resultate
5. Business Case / Nutzenanalyse
6. Grobplanung (Aufwand und Zeit)
7. Approach / Lösungsskizze (optional)
8. Kontext Diagramm
9. Informationsmodell
10. Prozesslandkarte (optional)
11. Funktionale Zerlegung
12. Systemdesign
13. Projektstruktur

## 1.1 Ausgangslage

Die Ausgangslage beschreibt die Problemstellung respektive Probleme, die zum Projekt führen.

WARUM will man das Projekt machen?

## 1.2 Ziele

WAS will man mit dem Projekt erreichen?
Was soll NACH dem Projekt „besser" sein als vorher?

Das Ziel beschreibt einen Zustand / Situation.

## 1.3 Scope

Abgrenzung, Anwendungs- und Gültigkeitsbereich

Der Scope beschreibt, was in ein Projekt „hineingehört", was nicht zu dazugehört und die Einschränkungen (constraints).

## 1.4 Resultate / Ergebnisse

Mit den Resultaten meint man die Arbeitsergebnisse, welche das Projekt abzuliefern hat. Man beschreibt, was der Auftraggeber oder die Benutzer „in die Hand" kriegen.

Resultate sind nicht Aktivitäten!
Also: Nicht das Durchführen eines Workshops ist interessant, sondern z.B. die daraus hervorgegangenen Anforderungen.

Resultate sind präzis und klar zu umschreiben.

präzise Beschreibung der Projektresultate ist DIE Grundlage für die Projektplanung

Ziele → Hauptresultat → Teilresultat → Arbeitspaket

### 1.4.1 Warum gute Resultatformulierungen?

Präzise formulierte und gut strukturierte Projektresultate benötigt man während des ganzen Projektablaufs:

- Für die Detaillierung der Arbeitspakete: Resultate → Teilresultate → Arbeitspakete
- Für die Aufwandschätzung
- Für das Projekt-Controlling
  - Aufwandskontrolle
  - Kontrolle der Erreichung der Resultate
- Für das Projektreporting

Alle wichtigen Projektplanungs- und –Reporting- Instrumente hängen direkt von der klaren Formulierung der Resultate und Arbeitspakete ab.

### 1.4.2 Ergebnisraster

Die Ergebnisse eines Projekts können (fast) immer in folgende Klassen unterteilt werden:

1. Applikationssoftware
2. Applikationsdokumentation
3. Prozesse / BO (Betriebsorganisation)
4. Migration
5. Rollout
6. Projektmanagement-Ergebnisse

Gemäss DiMA-Projekt können IT-Projekte immer in diese fünf Gruppen innerhalb der Resultate gegliedert werden:

1. Applikationssoftware (Proof of Concept, Release 1)
2. Dokumentation (Konzept, Detailspezifikation, Benutzerhandbuch, Betriebshandbuch)
3. Businessorientierte Resultate
4. Rollout (Einführung und Verteilung, Schulungskonzept)
5. Projekt Management (Planung, Scope, Business Case, Risikoanalyse)

## 1.5 Business Case / Nutzenanalyse

Der Business Case oder Nutzenanalyse beschreibt den Nutzen eines Projekts:

- Qualitativ: Beschreibung des Nutzens in Prosa
- Quantitativ: Berechnung des Nutzens in Fr. (optional)

Der Nutzen lässt sich von den Zielen ableiten und ist in drei Kategorien unterteilbar:

- Erzeugung von Mehrwert (Einnahmen)
- Reduktion von Kosten
- Externe Zwänge: z.B. Gesetze

## 1.6 Grobplanung (Aufwand und Zeit)

Mit der Grobplanung skizziert man

- A) einen groben Zeit- und Meilensteinplan
- B) einen grobe Aufwand- und Kostenschätzung

Voraussetzung für diese Grobplanung sind die Projektresultate und eine grobe Lösungsskizze (Approach)

## 1.7 Approach / Lösungsskizze (optional)

Die Lösungsskizze oder englisch Approach beschreibt, wie man gedenkt mit dem Projekt die Probleme zu lösen.

Dies kann eine Optimierung eines Prozesses sein, die Entwicklung oder der Einkauf einer oder mehrere Applikationen und vielleicht der Umbau einer Organisation.

## 1.8 Kontext Diagramm

Das Kontextdiagramm veranschaulicht in der Konzeptionsphase das Umfeld des zu realisierenden IT-Systems. Insbesondere wird klar abgegrenzt, was zum Aufgabenbereich des IT-Systems gehört und was nicht.

Weiterhin veranschaulicht es grob, welche Informationen in das System hinein fließen und welche Informationen das System liefern kann. Das Kontextdiagram wird zu einer sehr frühen Phase der Systemkonzeption eingesetzt.

## 1.9 Informationsmodell

Das Informationsmodell in einem Grobkonzept beschreibt die wichtigsten Informationsobjekte und deren Zusammenhänge, welche für das Projekt, das zu bauende SW-System und für das Projektverständnis eine zentrale Rolle spielen.

## 1.10 Prozesslandkarte (optional)

Eine Prozessskizze oder eine Prozesslandkarte gibt einen Überblick über die zu automatisierenden Geschäftsprozesse. Es wird beschrieben, für welchen Geschäftsbereich eine Applikation erstellt werden soll.

## 1.11 Funktionale Zerlegung

Mit der Funktionalen Zerlegung (Funktionen-Diagramm) beschreibt man die wichtigsten Funktionen oder Funktionsblöcke eines ITSystems.
→ Klarheit über die Funktionalität des Systems

## 1.12 Systemdesign

Mit dem Systemdesign im Grobkonzept wird der physische Aufbau eines IT-Systems beschreiben.

Mit physischem Aufbau meint man einzusetzende Software- und Hardwarekomponenten und deren Aufbau.

## 1.13 Projektstruktur

Organisation des Projektes. Ansprechpersonen

# 2 Requirements Engineering

## 2.1 Was ist das?

Requirements Engineering ist diejenige Disziplin, welche die Anforderungen an das Zielsystem aktiv durch ihren gesamten Lebenszyklus begleitet.

Das Requirements Engineering als erster Schritt der Systementwicklung entscheidet massgeblich über den Erfolg oder Misserfolg eines Projektes.

Viele weitere Disziplinen hängen von den Ergebnissen des Requirements Engineering ab. Diese sind:

- 
- 
- 
- 
- 
- 

## 2.2 Requirements Management

Anforderungen können im Projektverlauf ändern:

- Veränderte Bedürfnisse von Kunden
- Veränderung von Märkten / neue Märkte
- Änderung der Organisation (kundenseitig oder entwicklerseitig)
- Neue / geänderte politische oder rechtliche Randbedingungen
- oder der Kunde wird gescheiter, merkt dass etwas anders laufen sollte

Ziele:

- Anforderungen stabil halten und Veränderungen kontrolliert zulassen
- Beherrschen der Evolution von Anforderungen und Traceability

Massnahmen:

- Konfigurationsmanagement für Anforderungen

Traceability:
Nachvollziehbarkeit der Änderungen, sodass man später weiss, warum eine Anforderung neu dazugekommen ist
→ Process Improvement, man kann daraus lernen,

save my ass (reiner Selbstschutz

## 2.3 Warum?

60% der Fehler passieren in der Analyse, in Design und Implementierung nur 40%. Die Kosten der Fehlerbehebung steigen aber exponentiell mit dem Zeitpunkt der Entdeckung
→ bei den Requirements investieren!!

Ausschlaggebend sind eigentlich nur sogenannte Softfactors. Man spricht nicht miteinander.

Requirements Engineering sorgt dafür, dass

- die Anforderungen mit dem Auftraggeber abgestimmt werden,
- die Anforderungen so weit ausgearbeitet werden,
- dass das System gebaut werden kann,
- die Anforderungen und mögliche Änderungen daran über das Projekt hinweg verfolgt werden.

Requirements Engineering ermöglicht die Verfolgbarkeit von Anforderungen über die Spezifikation und die Module bis hin zu den Testfällen des Systems („Traceability")!

**Die drei Risiken im RE – und für Ihr konkretes Projekt**

**1 Fehlende Anforderungen**

- Kunde nicht involviert; unklarer Bedarf and Nutzen
- Kritische Anforderungen übersehen
- Nur funktionale Anforderungen

**2 Falsche Anforderungen**

- Vermischung von was (Bedarf) und wie (Lösung)
- Keine Verifikation / Validierung

**3 Sich ändernde Anforderungen**

- Unbekannte / unzureichende Baseline, auf die aufgebaut wird
- Unzureichendes Änderungsmanagement

Baseline:
Grundstock von Anforderungen

*Basiert auf: B.Lawrence, K.Wiegers and C.Ebert: The Top Risks of Requirements Engineering. IEEE Software, Vol. 18, No. 6, pp. 62-63, Nov. 2001.*

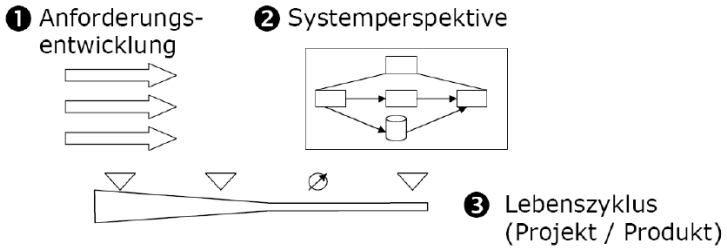
### 2.3.1 Ebenen der Anforderungen

**Marktanforderungen** /
Kunden- / Benutzer- / Geschäfts-anforderungen / Ziele / Bedürfnisse

**Produkt (Service-) anforderungen** /
Eigenschaften / SLA /
Systemanforderungen

**Komponenten-anforderungen** /
Softwareanforderungen

### 2.3.2 RE im Systemkontext




- Anforderungen werden aus Bedürfnissen heraus entwickelt
- Anforderungen werden im Systemzusammenhang analysiert
- Anforderungen werden lebenszyklus-übergreifend umgesetzt

- ☑ Korrekte Funktion im Produkt „end to end"
- ☑ Systematische und konsistente Umsetzung.
- ☑ Beherrschte Komplexität: Qualität, Termin- und Kosteneinhaltung

## 2.4 Hauptaktivitäten

*Die Hauptaktivitäten im Requirements Engineering sind:*

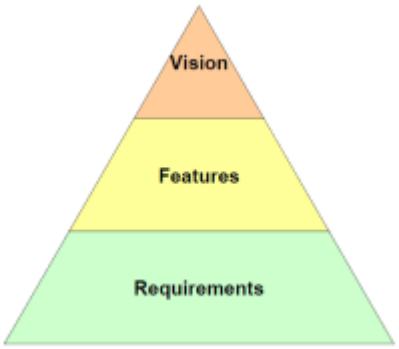
- Sammeln
  - *Gewinnung*
  - *Analyse*
- Ausarbeiten
  - *Spezifikation*
  - *Validierung*
- Verwalten
  - *Freigabe*
  - *Änderung (Change Management!)*
  - *Verfolgung*

*Aus den Aktivitäten „Gewinnen", „Analyse" und „Spezifikation" ergibt sich eine Anforderungspyramide:*

*Vision:*
*Systembeschreibung in einem Satz*

*Features:*
*Grobe Beschreibungen der Anforderungen*

*Requirements:*
*Ausgearbeitete Features*




## 2.5 Sammeln

*Kontext des Zielsystems bestimmen → Kontextdiagramm*

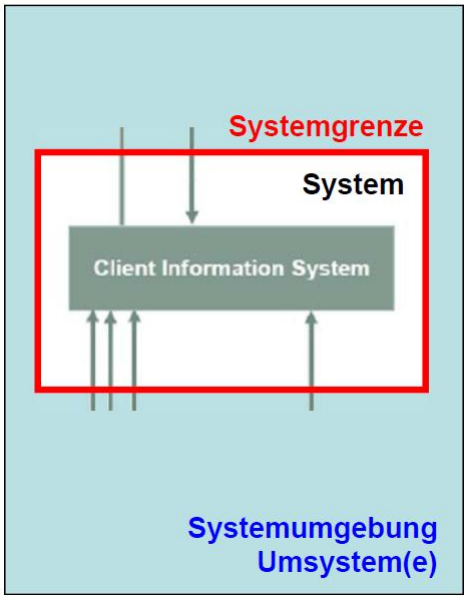
### 2.5.1 Kontextdiagramm

*Das Kontextdiagramm zeigt die Systemgrenze*

- *Was gehört zum System das ich entwickeln muss, das ich beeinflussen kann*

- Was gehört zur Systemumgebung
  Umsysteme, die vorgegeben sind, schon bestehen oder in einem andern Projekt von Dritten entwickelt werden
- Welche Schnittstellen hat mein System die mein System anbieten muss oder die es von den Umsystemen benötigt

Die Anforderungen beschreibendas System und seine Schnittstellen




### 2.5.2 Geschäftsereignisse und -prozesse

- Betroffene Geschäftsereignisse und Geschäftsprozesse bestimmen
  (erst IST, dann SOLL)
  - Diejenigen Teile der Geschäftsprozesse herausfinden, die von dem Zielsystem unterstützt werden sollen
    → fliessen in die Use Cases (UCs) ein.

Beispiel: Organisation einer Kundenveranstaltung im CIS. Dazu werden die Profile der Kunden benötigt, um die richtigen Leute einzuladen. Das ändert sich mit CIS.

### 2.5.3 Aus welchen Quellen schöpfen?

Quellen für Anforderungen:

- Stakeholder:
  - offizielle Ansprechpartner beim Auftraggeber
  - noch inoffizielle (aber betroffene!) Ansprechpartner beim Auftraggeber:
    - Hotline-MA
    - Umsystem-Betreuer
    - BO
    - ...
- Vorhandene Dokumentation
- Gesetze und Verordnungen usw.

### 2.5.4 Know your user

| | |
|---|---|
| Level of knowledge and experience<br>• computer literacy<br>• system experience<br>• experience with similar applications<br>• education / reading level | Psychological characteristics of the user<br>• Attitude towards job<br>• Motivation<br>• Cognitive style ( verbal vs. spatial; analytic vs. intuitive; concrete vs. abstract) |
| Characteristics of the user's tasks and jobs<br>• Frequency of use<br>• Turnover rate for employees<br>• Importance of task<br>• Repetitiveness of task<br>• Training anticipated | Physical characteristics of the user<br>• Age<br>• Gender<br>• Physical handicaps (color-blind; deaf; ...) |

### 2.5.5 Fallstricke

Es ist Vorsicht geboten!

- „People hate change"
- Politische Interessen, Machtspielchen etc.
- Manche Leute sagen nicht, was sie denken
- Manche Leute wissen nicht, was sie wollen

### 2.5.6 Techniken

| Technik | Wann einsetzen? | Nutzen |
|---|---|---|
| Interview | Mindestens am Anfang; oft! | Liefert bei versierter Interviewtechnik viele und nützliche Daten |
| Fragebögen | Fragen sind bekannt, aber viele Leute | Exakte Antworten, Klärung der „Lager" |
| Workshops | Komplexe Themen | Intensiv (aber auch aufwändig!) |
| Szenarien durchspielen | Erster Entwurf steht | Liefert die nötigen Details, offenbart Lücken im Verständnis, holt Nutzer ins Boot |
| Prototyp | Prosa-Spezifikation für Nutzer schwer verdaulich | Gibt ein gutes Gefühl über das spätere System (wenn nicht im Rahmen iterativer Entwicklung sehr aufwändig!!) |
| Beobachten (vor Ort) | Prozesse und Umfeld fremd für Auftragnehmer | Reduziert Risiko implied needs zu übersehen |

Für ein Informatiker ist bereits ein GUI ohne jegliche Funktionalität ein Prototyp...

implied needs = ??

## 2.6 Ausarbeiten

### 2.6.1 Sichtweisen

- Auftraggeber will die Spezifikation verstehen und abnehmen
  → darf nicht zu detailliert, nicht zu technisch sein
- Entwickler müssen wissen, was sie bauen sollen
  → muss detailliert und auch technisch sein – je nach Fähigkeiten des Teams unterschiedlich
  → unterschiedliche Dokumente erzeugen:
    - z.B. HTAgil Kundenanforderung als Spezifikation gegenüber Auftragnehmer
    - z.B. HTAgil Systemspezifikation zum Festhalten technischer Vorgaben: Kap. 3.(Schnittstellen) / 4.(SW-Anforderungen) / 5.(Environment)

Management-Folie, die knapp die wichtigsten Eigenschaften und den Kontext eines Systems darstellt.

### 2.6.2 Hilfsmittel

Techniken zur Verfeinerung der Anforderungen siehe oben.

Zur Ausarbeitung der fachlichen Spezifikation (Objektmodell etc.) dienen:

- UCs für die funktionalen Anforderungen
- Formulare für die nichtfunktionalen Anforderungen
- Formalere Mittel, wo nötig:
  - Ablaufdiagramme
  - Informationsmodell
  - Zustandsdiagramme

Fachliche Spezifikation mittels:

- Zustandsdiagramm
- Use Case (üblicherweise Formularbasiert)

### 2.6.3 Kategorien von Anforderungen

- Funktional/Nichtfunktional
- Vorschrift/Tatsache/Annahme
- Hart (Sperrung nach Ablauf) / Weich (einfach nutzbar)
- Repräsentation:
  - Operational: Kontostand = Kontostand + Buchungsbetrag
  - Quantitativ: Antwortzeit < x
  - Qualitativ: Hohe Verfügbarkeit
  - Deklarativ: Soll unter LINUX laufen

### 2.6.4 Checklisten

**PROJECT DRIVERS:**

1. The Purpose of the Project
2. Client, Customer, Stakeholders
3. Users of the Product

**PROJECT CONSTRAINTS:**

4. Mandated Constraints
5. Naming Conventions and Definitions
6. Relevant Facts and Assumptions

**FUNCTIONAL REQUIREMENTS:**

7. The Scope of the Work
8. The Scope of the Product
9. Functional and Data Requirements

**NON-FUNCTIONAL REQUIREMENTS:**

10. Look and Feel
11. Usability and Humanity
12. Performance
13. Operational
14. Maintainability and Support
15. Security
16. Cultural and Political
17. Legal

**PROJECT ISSUES:**

18. Open Issues
19. Off-the-shelf Solutions
20. New Problems
21. Tasks
22. Cutover
23. Risks
24. Costs
25. User Documentation and Training
26. Waiting Room
27. Ideas for Solutions

### 2.6.5 Goldene Regeln

- Formuliere Anforderungen, keine Lösungen!
- Schreibe klar und verständlich!
- Definiere die Subjekte, vermeide Passivsätze!
- Definiere die Bedeutung von „muss“, „soll“, „kann“, wenn Du sie benutzen willst!

### 2.6.6 Validierung

Die Anforderungen werden vom Auftraggeber abgenommen und damit gültig! Am besten richtet man ein Quality Gate ein, durch das alle Anforderungen hindurch müssen.

Qualitätskriterien für Anforderungen:

- Vollständigkeit
- Korrektheit
- Klassifizierbarkeit (rechtliche Relevanz; muss/soll/kann)
- Konsistenz
- Prüfbarkeit (Testfälle!!)
- Eindeutigkeit (können nicht anders verstanden werden)
- Klare Verständlichkeit
- Gültigkeit und Aktualität
- Realisierbarkeit, Kalkulierbarkeit der Kosten
- Notwendigkeit zur Erreichung des Ziels
- Gewichtbarkeit (Prioritäten)

## 2.7 Verwalten

### 2.7.1 Anforderungs-Datenbasis

Alle Anforderungen kommen in eine gemeinsame Datenbasis.

Attribute:

- Urheber
- Status: erhoben, akzeptiert, zurückgestellt, gestrichen, ausgeliefert.

    - bei iterativem Vorgehen immer mit Bezug zu einer bestimmten Iteration.
- Komplexität
- Verbindung zu den UCs

### 2.7.2 Evolution von Anforderungen

Anforderungen unterliegen einer Evolution

- Fortschritte der Technologie
- Änderung der unternehmensinternen Organisation
- Veränderte Bedürfnisse von Kunden
- Veränderung von Märkten / neue Märkte
- Neue / geänderte politische oder rechtliche Randbedingungen

Problem:

- Anforderungen stabil halten und
- Veränderung kontrolliert zulassen

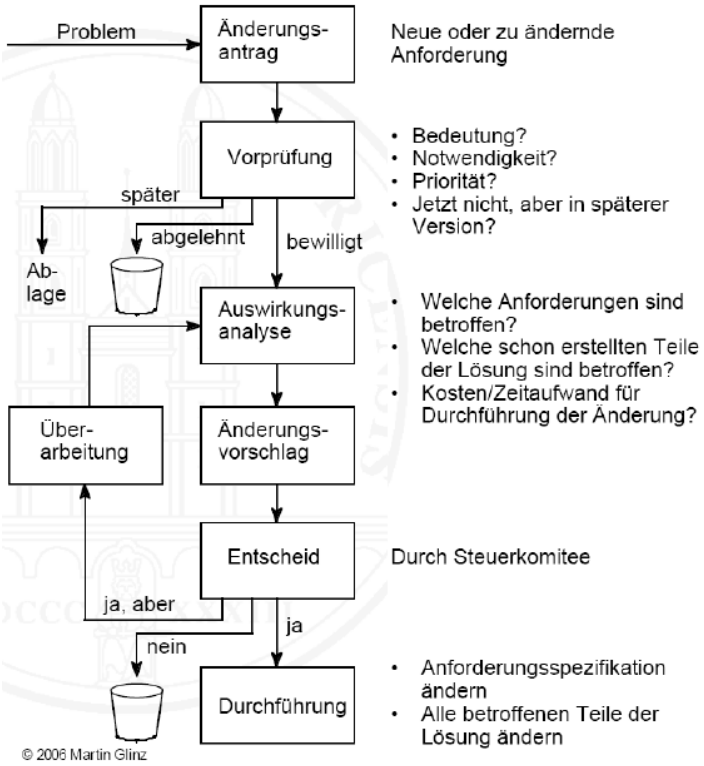
Es braucht

- Konfigurationsmanagement für Anforderungen
- Verfolgbarkeit von Anforderungen

### 2.7.3 Änderungsprozess

- Geregeltes Prozedere
- Entscheidung durch Steuerkomitee *)
    - Mitglieder: Vertreter vonAuftraggeber und Auftragnehmer
    - Vorsitz: Projektleiter

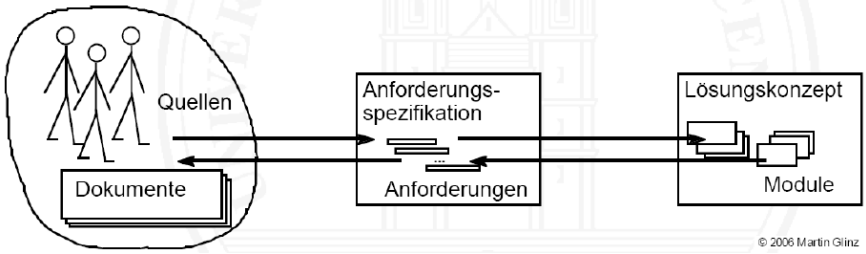
*) Synonyme: Lenkungsausschuss, Change Control, Board



*Anmerkung:*
*Reorganisationsprojekte werden oftmals als IT-Projekte dargestellt. Dann ist die IT der Sündenbock für die Reorganisation.*

### 2.7.4 Verfolgbarkeit (Traceability)

- *Rückwärts: Wo kommt welche Anforderung her?*
- *Vorwärts: Wo ist welche Anforderung entworfen bzw. implementiert?*
- *Wie hängen Anforderungen voneinander ab?*
- *Aufwand und Ertrag für Verfolgbarkeit gegeneinander abwägen*
- *Rückverfolgungsbeziehungen pflegen, sonst sind sie nutzlos*
- *Benötigt Werkzeugunterstützung*




## 2.8 Requirements Management und Werkzeuge

- *Rational/IBM Requisite Pro*
- *Telelogic DOORs*
- *Excel*

# 3 Effiziente Textanalyse und –umsetzung

## 3.1 Textverständnis

Beim Textverständnis geht es in erster Linie darum, Wesentliches zu erkennen.

Drei Bestandteile gehören dazu, das Wesentliche eines Informationsblocks zu erkennen:

- Eigene Wahrnehmungskanäle richtig einsetzen,
- auf die Form der Information achten,
- den Aufbau der Information erkennen
- Vor dem Lesen Leseziel setzen: was ist das Resultat dieser Leseaktivität?

Mit diesem Vorgehen kann **Zeit gespart** werden.

Es ist sehr wichtig, die eigenen Wahrnehmungskanäle richtig einzusetzen:

- Je nach Botschaft und Lernziel
- Kanäle kombinieren:
  - Beim Lesen mitschreiben
  - Fremdwörter nachsprechen
  - Verstandenes anderen vortragen, auch sich selbst vorlesen

<u>Form der Information</u>

- Explizite Hinweise auf Wesentliches
- Strukturangaben (z. B. Titel, Kapitelunterteilung, Inhaltsverzeichnis)
- Aufzählungen
- Gesetzte Ziele
- Zusammenfassungen, Abstracts o. ä.
- Farbcodes
- Layout-Codes (Fettdruck, Kursivdruck, ...)
- Bilder, Tabellen

Der Aufbau der Information trägt viel zum Verständnis bei

Häufigste Struktur einer Informationseinheit: T-H-U-N

## 3.2 Textstrukturanalyse mit THUN

T Thema
H Hauptgedanken
U Unterstützende Einzelheiten
N Nebensächliches

| Strukturelement | Merkmale |
|---|---|
| Thema | Titel, 2-3 Schlagworte |
| Hauptgedanken | Grundsätze, Verallgemeinerungen, Modelle, Kernaussagen, Problemlösung, Hauptargumente |

| | |
|---|---|
| Unterstützende Einzelheiten | Erklärungen, Beispiele, Daten, Beweise, Motivationen |
| Nebensächliches | Ähnliche Beispiele, „Zierrat", Abschweifungen |

### 3.2.1 Thema

Fragen, um das Thema herauszubekommen:

- Worum geht es überhaupt?
- Wie lautet das (Lern-) Ziel?
- Welches ist das Kernproblem?

Das Thema wird oft am Anfang des Blocks genannt.

Regeln, um das Thema zu formulieren:

- Hauptsachen in Hauptsätze
- Mehrere Hauptsachen: Beiordnung
- Das Verb möglichst früh
- Kurze Sätze, gerne auch elliptisch
- Einfach und verständlich schreiben
- Telegrammstil mit Vorsicht anwenden

### 3.2.2 Hauptgedanken

Fragen, um die Hauptgedanken zu ermitteln:

- Was sind Botschaft/Anliegen des Autors?
- Was wird im ersten/letzten Satz eines Abschnitts gesagt?
- Welche Schlussfolgerungen werden aus der Erörterung des Themas gezogen?

### 3.2.3 Unterstützende Einzelheiten

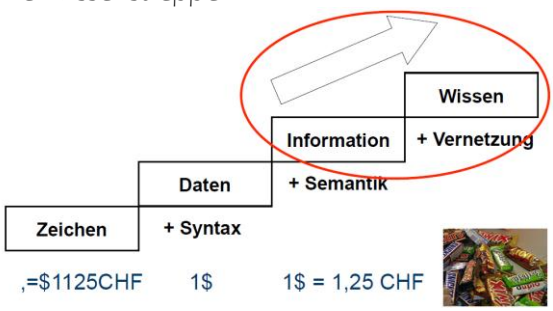
Fragen, um die unterstützenden Einzelheiten zu ermitteln:

- Womit werden die Hauptgedanken veranschaulicht?
- Welche Beispiele kommen?
- Womit werden Behauptungen konkretisiert oder bewiesen?

## 3.3 Schriftliche Texte verarbeiten

### 3.3.1 Von der Information zum Wissen

Die Wissenstreppe:



„THUN" → Informationen.
Information → Wissen: Verankerung im Erfahrungsschatz.

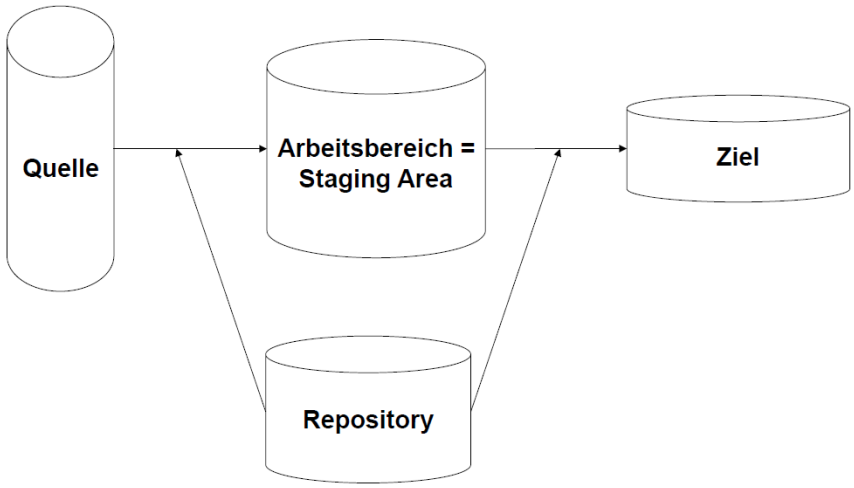
| Aktion | Nutzen |
|---|---|
| Anreichern, vertiefen | Gutes Verständnis |
| Ordnen, strukturieren | Brauchbarer Überblick |
| Wiederholen, üben | Langfristige Verfügbarkeit |

### 3.3.2 Anreichern, vertiefen

Techniken zum Anreichern und Vertiefen

- Gedankenstützen, Merksätze
- Eigene Zusammenfassung, Bilder, Zeichnungen
- Verwandtschaften suchen
- Anwendungsmöglichkeiten, Beispiele suchen
- Fragen an den Text stellen
- Mit anderen diskutieren

Quelle und Ziel sind sehr unterschiedlich. Ex- und Import sind zeitkritisch!




Repository = Kontrollzentrale

### 3.3.3 Ordnen, strukturieren

Ordnungsprinzipien

- Äussere Form (z.B. alphabetisch)
- Elemente, Merkmale
- Hierarchie: Über- und Unterordnung
- Vergleich, Beurteilung
- Beziehungen
- Kausal: Ursache – Wirkung
- Temporal: Reihenfolge, Ablauf

### 3.3.4 wiederholen, üben

Tipps:

- Lernziel(e) festlegen: Ist ganz THUN nötig oder genügen TH?
- „Kneipp-Kur": Wechselbad zwischen gründlichem und zusammenfassendem Wiederholen!
- „Früh übt sich …": Rechtzeitig beginnen!
- „Salami-Taktik": Öfter und in kleinen Portionen
- Kein „Überlernen"! Was sitzt, sitzen lassen.
- Soziale Motivation am wichtigsten: Konkurrenz, Vorbilder, Neugierde!
- Erfolgserlebnisse kurzfristig belohnen!
- Systematische Tricks („Becker-Faust") helfen – stellen den Organismus auf's Lernen ein.
- Vor dem Üben aufwärmen
- Teams sind gut zur Kontrolle und für's Coaching!

## 3.4 Informationen zur Verfügung stellen

Der folgende Ablauf stellt eine grundlegen Textarbeitstechnik dar:

1. bewusst lesen
2. richtig Notizen nehmen
3. Informationen recherchieren

### 3.4.1 bewusst lesen

Bewusstes Lesen einer Lektüre bedeutet:

- Vorbereitung
- Leseprozess
- Nachbereitung

<u>Vorbereitung der Lektüre</u>

- Welches Ziel verfolge ich beim Lesen?
  - eigene Arbeit vorbereiten
  - Prüfungsvorbereitung
  - seltene Anwendung
  - häufige Anwendung
  - Allgemeinbildung
  - Freizeitspass
- Wie gründlich muss ich Lesen (Abstract, ganze Arbeit, Teile davon etc.)
- Kontext?
- Umgebung herrichten, sich auf das Lesen einstimmen

<u>Leseprozess</u>

- Überblick über Lesestoff verschaffen!
- Inhaltsverzeichnis durchsehen, Grobgliederung herausfinden!
- Fragen an den Text stellen!
- Etappenweise lesen, z. B. „lesen -nachdenken - wiederholen - einprägen - lesen" usw.
- Bewusst, verstehend lesen! Anreichern, wo nötig.
- Erste Notizen nehmen – aber mit Überlegung (nicht besinnungslos alles mitschreiben)!

- Nur „TH" aufschreiben, „U" z. B. mit Textmarker hervorheben – weniger als 10% des Textes!

Nachbearbeitung
Nach jeder längeren Etappe:
- Zusammenfassende Wiederholung
- Bilanz: Was habe ich verstanden/gelernt, wo liegen noch Probleme?
- Aufgaben bearbeiten, sofern vorhanden
- Schwierige Stellen nochmals nachlesen

Am Ende:
- Hauptaussagen strukturieren, Kurzzusammenfassung schreiben
- Material vertiefen durch Anreichern
- Wichtige Texte von Zeit zu Zeit wiederholen

### 3.4.2 richtig Notizen nehmen

Notizen unbedingt selbst machen!
- Situation beurteilen → überhaupt Notizen nehmen? Wenn ja, wie?
- Auf Originalen schreiben, wenn möglich
- Notizen vom letzten Mal durchsehen!
- Nur „THU" notieren, „N" weglassen!
- Objektiv notieren!
- Stichwörter! Ausser bei Formeln, Zitaten, Definitionen etc.
- Hilfen:
  - Farbcodes
  - Abkürzungen
  - Lange Wörter kürzen
  - Top-Tipp: Stenografie!

Nachbearbeiten:
- Lücken ergänzen (keine neue Reinschrift)
- Anreichern, vertiefen etc.
- Erste Wiederholung zeitnah – es prägt sich besser ein!

### 3.4.3 Informationen recherchieren

Es gibt drei verschiedene Ebenen der Information
- präsent (Zeitung, Bücher etc.)
- Fachwissen

- schwer zugängliches Wissen (Unterlagen von Tagungen, Dissertationen usw.) – „graue Literatur"

Quellen für wissenschaftliche Recherchen sind:
- Fachliteratur → Bibliothekskataloge
- Internet/Intranet
- Datenbanken!

Vorgehen bei wissenschaftlichen Recherchen:

1. Text suchen zu übergeordnetem Thema! Schlagwortkatalog, Textsuche www, Datenbanken
2. Literaturhinweise auswerten → Grundlagentext
3. Literaturhinweise auswerten → weitere Texte
4. Weiter bei 3. bis zur Sättigung

Tipps für Recherche:

- Internet-Recherche:
    - Fragen: Seite aktuell? Impressum? Firmenseite? Werbestil? Gut aufgebaut?
    - pdf-Whitepapers!
    - Vorlesungsunterlagen!
    - Nicht nur Google
        - metacrawler.com
        - search.com
        - acm.org
        - ieee.org
        - scholar.google.com

Nicht nur nach Suchworten suchen, sondern auch Klassifikationen verwenden!

- Bibliographie anlegen
- Leute („Experten") fragen!
- Auch Stichwörter auf Englisch versuchen!

# 4 Strukturierter Aufbau eines IT Konzepts

Beispiele für „IT-Konzepte“:

- Technische Texte über Computer, Betriebssysteme, Programmiersprachen, Spezifikationsmethodiken, etc.
- Texte welche Anforderungen an IT-Systeme beschreiben
- Grobkonzept für ein IT-System

Texte, die nicht „IT-Konzepte“ sind

- Krimi
- Gedicht
- Tageszeitung allgemein
- Wirtschaftsmagazin

## 4.1 Arbeitsphasen auf dem Weg zum IT Konzept

1. Erste Orientierung über Thema, Thema eingrenzen und konkretisieren
2. Recherchieren und Material beschaffen
3. Strukturieren, Disposition entwerfen
4. Konzept-Rohfassung erstellen
5. Konzept redigieren, Endfassung erstellen

## 4.2 Ziele des zu verfassenden IT Konzepts festlegen

### 4.2.1 Ziele festlegen

Zu Beginn sollten folgende Fragen beantwortet werden:

- Was ist das Thema, das Scope des zu Behandelnden?
- Ziel: was will ich/mein Auftraggeber mit dem Konzept hauptsächlich erreichen?
- Wer ist das Zielpublikum?
- Was ist meine Rolle? Als Wer schreibe ich den Text?
- Art des Konzepts
- Erwarteter/sinnvoller Umfang?

### 4.2.2 Thema festlegen

- Klar und bestimmt: Ein Thema!
- Vorgabe beleuchten: Über- und Unterthemen ansehen
- Nicht 0815-mässig, sondern kreativ und pfiffig!
- Nicht abgegriffenes wieder aufwärmen, sondern Ungewohntes kombinieren!

## 4.3 Schreiben

### 4.3.1 Arbeit vorbereiten, Informationen zusammenstellen

Schreibsituation klären:

- Sind alle Aspekte des Ziels geklärt?
- Vordefiniertes Layout?
- Wieviel Zeit habe ich?
- Ist eine sinnvolle Anzahl von Informationsquellen vorhanden?
- Ist Koordination in einer Gruppe nötig?
- Absehbare Probleme?

- Sprache?

Informationen zusammenstellen:
- Sammeln: Recherchetechniken
- Sofort: Arbeitsbibliographie erstellen
- Auswertung:
  - Zentrale Quellen überfliegen (Selektion!)
  - Gründliches Quellenstudium bei Bezug zum Thema (möglicherweise 2× nötig!)

→ siehe vorhergehendes Kapitel

### 4.3.2 Material strukturieren, Disposition entwickeln

Folgende Fragen müssen für die Disposition geklärt werden:
- Thema → aus Zieldefinition
- Daraus abgeleitete Hauptgedanken?
- Roter Faden?
- Inhaltsstruktur?
- Zentrale Tabellen/Grafiken?

Qualität einer Arbeit = Struktur + Stil

klassische Grundeinteilung und Rollen
1. Titel → „Teaser", Attraktion
2. Management Summary, Zusammenfassung → das Wichtigste für sehr eilige Leser
3. Einleitung → Schaffen des Hintergrundes, Hilfe für das Lesen des Rests
4. Hauptteil, in etwa gleich grosse Kapitel gegliedert → der eigentliche Inhalt
5. Schluss → Resultat, Schlussfolgerungen, wie weiter?

### 4.3.3 Text schreiben

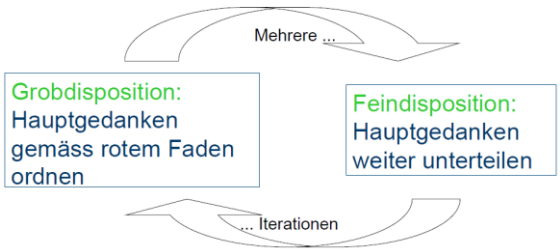
Empfohlene Reihenfolge des Verfassens
1. Hauptteil
2. Schluss
3. Einleitung
4. Management Summary, Zusammenfassung
5. Titel

Hauptteil
Hauptideen/-gedanken aus dem Material heraussuchen:
- Bezug zu dem Thema!
- Häufige Nennung in den Quellen

„Interessante?" Informationen weglassen
- »When in doubt, leave it out!«

Schemata für roten Faden:
(1) chronologisch: zeitliche Abfolge, Prozessschritte
(2) hierarchisch: vom Wichtigen zum weniger Wichtigen, vom Allgemeinen zum Besonderen, ...
(3) deduktiv: Behauptung, Beweis, Schlussfolgerung

(4) strukturell, nach Art: Barock, Klassik, Romantik
(5) argumentativ: Pro/Contra
(6) didaktisch: vom Leichten zum Schweren




Grob-/Feindisposition:

- Gliederungspunkte auf jeder Ebene: maximal 7±2.
- Am Ende jedes Abschnitts: Kurze Zusammenfassung!

## Zwischen Kapiteln Brücken schaffen!

- Am Ende des Kapitels kurzen Ausblick auf nächstes geben und Zusammenhang darstellen
- Am Anfang eines Kapitels kurz aufgreifen, wo das letzte Kapitel stehen geblieben ist und Brücke schlagen.

Brüche und „Löcher" in den Gedankengängen unbedingt vermeiden

- Daran denken, dass der Leser das, worüber der Autor Stunden und Tage nachgedacht hat, innert Minuten „wiederdenken" soll
- Komplett neue Gedanken nicht einfach in den Raum stellen, sondern ...
    - Leser von einem Gedanken zum nächsten führen, indem auch Text in den Zusammenhang dieser Gedanken investiert wird
    - Oft ist dem Leser das unklar, was dem Autor halb- oder unbewusst sonnenklar war. Darüber nachdenken!

## Schluss

„Den Sack zumachen", „Rausschmeisser"

Inhalt:

- Fazit
- Ausblick, wie weiter?
- (Frage an die Leser)
- Empfehlungen
- Handlungsalternativen mit Vor-/Nachteilen

## Einleitung

- Hintergrund, „Aufhänger", Basis, Existenzberechtigung des Texts!
- „To Be (Read) Or Not To Be (Read)" → muss extrem informativ sein:
    - Einführung in das Thema
    - Die ganz wesentlichen Fakten sehr gedrängt
    - Hauptproblem, Kernaussage, zentrale Frage
    - Schlussfolgerung des Autors „in a nutshell"

- Einleitung soll dem Leser Hilfe geben für das Lesen des Rests
- Oft am Ende der Einleitung Überblick geben über den Rest

Management Summary, Zusammenfassung

- In der Kürze liegt die Würze
- Sich Vorgabe für die Länge geben, z. B. 250 Wörter
- „Word count" des Word verwenden
- Nur der Kern soll vorkommen
- Welche drei Punkte will man unbedingt hinüberbringen?

Titel → siehe vorhergehendes Kapitel

### 4.3.4 Arbeit gestalten

Tipps für das Formale:

- Zwischentitel setzen, konsistent nummerieren!
- Pro Hauptabschnitt immer wieder „im Kleinen":
  - Einleitung
  - Hauptteil
  - Schluss
- Verzeichnisse
  - Inhaltsverzeichnis
  - Tabellenverzeichnis
  - Abbildungsverzeichnis
  - Abkürzungsverzeichnis
  - Begriffserklärungen
  - Referenzen/Quellenangaben

### 4.3.5 Sprachtipps

Tipps für die „Schreibe" (Stil):

- Spontan formulieren!
- Fussnoten vermeiden!
- Rechtschreibfehler vermeiden!
- Einfach schreiben, verständlich bleiben!
- Sachlich und nüchtern sein/bleiben!
- Genereller Tipp:»Read the Masters!«

Verständniskiller unbedingt vermeiden!

**Wörter**

- **Nicht erklärte Fachwörter, Abkürzungen, Spezialbegriffe**

**Satzbau**

- **Nominal- und Attributkonstruktionen**
- **Schachtelsätze, „Rekursion"**
- **Nebensätze am Anfang**

**Ganzer Text**

- **Keine Einführung in das Thema**
- **Struktur fehlt, kein roter Faden, keine Übergänge zwischen Kapiteln**
- **Unanschaulichkeit, Abstraktheit, keine Beispiele, kein Bezug zur menschlichen Realität**

## 4.4 Richtig zitieren

Es gibt viele Zitiersysteme. Wichtig ist Konsistenz im gesamten Text → z.B. APA

Tipp: Literaturverzeichnis von Word verwenden

## 4.5 Review

Review

- selber nochmals alles lesen
- Andere lesen lassen und systematisch Kommentare sammeln

Der Zustand, dass man nichts mehr findet, das man ändern könnte, erreicht man nie → Haltepunkt setzen
Punkte notieren, auf welche man bei der Review achten sollte

- Aufbau/Struktur logisch und sinnvoll?
- Roter Faden nachvollziehbar
- Brücken/Lesehilfen vorhanden?
- Sprache korrekt und verständlich? – Rechtschreibefehler sind peinlich
- Erreichen die gewählten Formulierungen die gesteckten Ziele, speziell in der Zusammenfassung?
- Alle Verzeichnisse und Referenzen vollständig?
- Form / Layout ansprechend?

# 5 Modellierung

Was verstehen Sie intuitiv unter „Modell"?

- Abstraktion aus der realen Wert
- Vereinfachung
- Abbild der Wirklichkeit
- „Zusammenhänge" darstellen, damit etwas klarer wird
- Zweck Simulation
- 2D/3D
- Geschäftsmodelle
- soziale Modelle, z.B. Bevölkerungsentwicklung

In der Informatik stellen Modelle Vorbilder (Pläne) dar.

Nicht wertneutral

- Modell – Konkretes oder gedankliches Abbild eines vorhandenen Gebildes oder Vorbild für ein zu schaffendes Gebilde in der Wahrnehmung der beteiligten Personen für einen bestimmten Verwendungszweck

Grösstmögliche Ähnlichkeit zwischen Original und Modell
kein Ziel

- Bewusste Abstraktion und Gestaltung des Modells
- Ausnahme: Anfertigung von Kopien

Validierung erforderlich

- Alle relevanten Eigenschaften des Originals müssen adäquat und vollständig auf Eigenschaften des Modells abgebildet sein

Wozu erstellt man Modelle? Warum sind Modelle sinnvoll?

- um zu verstehen → ein „System"
- Kommunikation über ein System
- Spezifikation eines Systems
  → wichtig

- Gedankliches Hilfsmittel zum Gestalten, Bewerten oder Kritisieren eines geplanten Systems oder Varianten davon.

Überlegen Sie, wie ein Prozess aussieht, dessen Resultat ein Modell ist, z.B. ein Datenmodell. In welchen Schritten gehen Sie vor?
2 Rollen:

- Modellierer
- Wissensträger

1. Reflektieren: Welches Modell der Ausschnitt
2. Informationen gewinnen (Was müssen diese wissen? z.B. mittels Workshop)
3. Modell beschreiben
4. Validieren

## 5.1 Merkmale eines Modell

Abbildungsmerkmal
Jedes Modell ist ein Abbild oder Vorbild

Verkürzungsmerkmal
Jedes Modell abstrahiert

Pragmatisches Merkmal
Jedes Modell wird im Hinblick auf einen Verwendungszweck geschaffen

Manchmal werden Modelle als Abstraktion eines Ausschnitts der Realität definiert

### 5.1.1 Abbildungsmerkmale

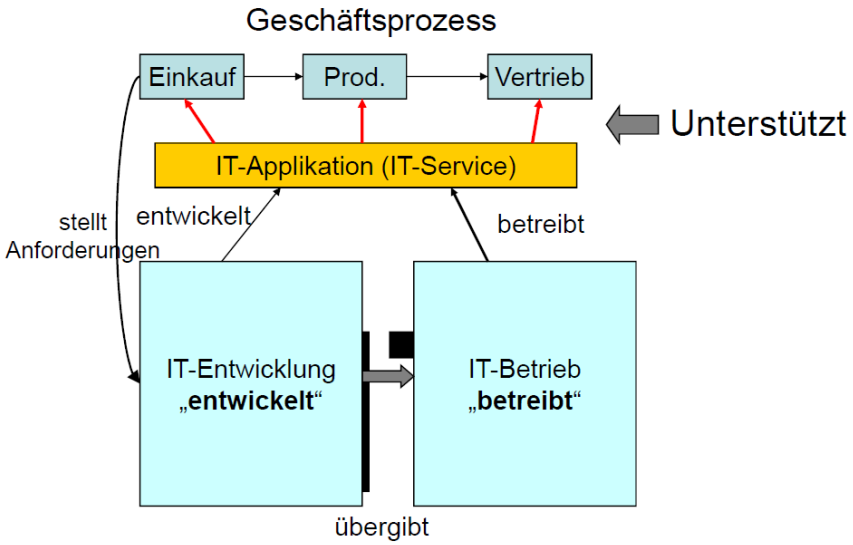
- Modelle sind Abbilder oder Vorbilder eines vorhandenen oder zu schaffenden Originals
- Zu jedem Modell gehört eine Abbildung, welche die Details des Originals auf diejenigen des Modells abbildet
- Das Original kann selbst wieder ein Modell sein
- Es kann verschiedene Modelle des selben Originals geben

### 5.1.2 Abstraktion / Vereinfachung

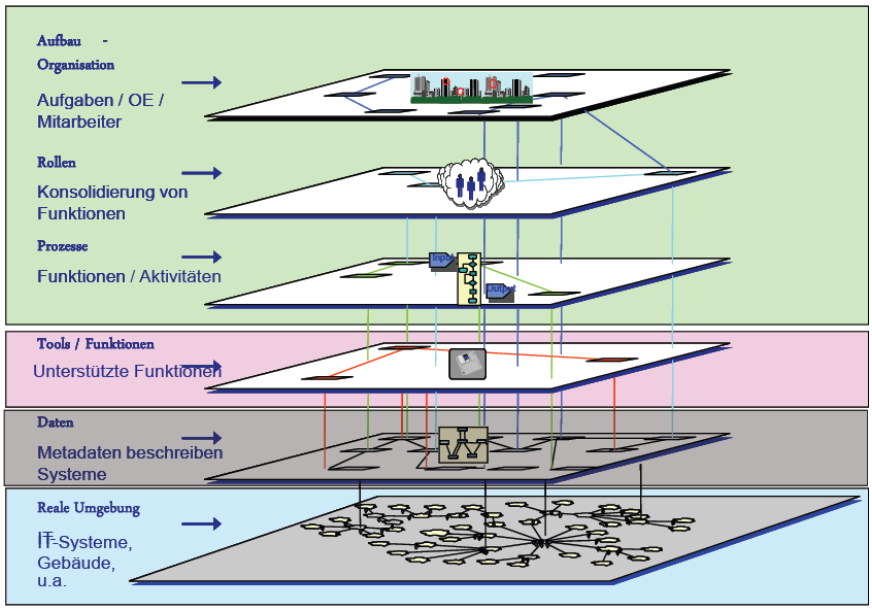
- Modelle erfassen meistens nicht alle Elemente und Details des Original
- Es wird nur das modelliert, was den Modellschaffenden wichtig/nützlich/notwendig erscheint → Abstraktion
- Das Modell kann Eigenschaften enthalten, die keine Entsprechung im Original haben: SOLL-Modell

## 5.2 Zweck der IT

Die IT unterstützt.



### 5.2.1 Sichten in der Unternehmung




### 5.2.2 Betrachtung aus verschiedenen Sichten

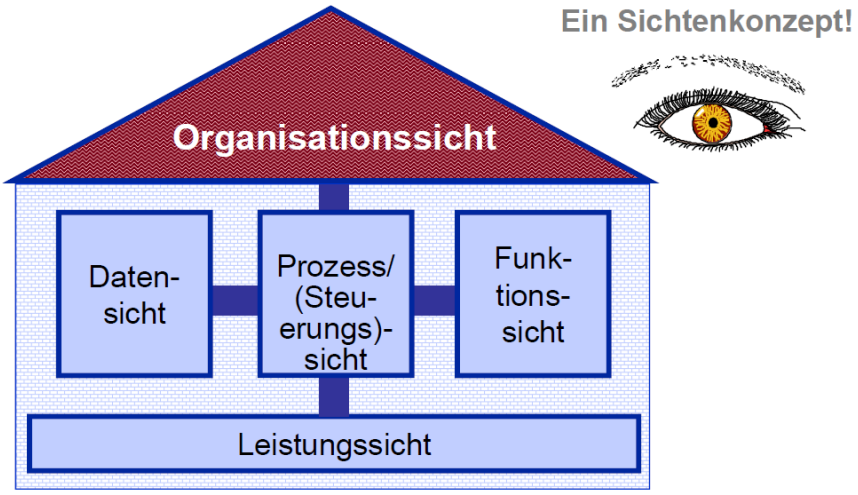
| | Organisation | Prozesse | Funktionen | Daten |
|---|---|---|---|---|
| Strategie | Sichtenübergreifende Betrachtung | | | |
| Fachliches Konzept | Aufbauorganisation | Geschäftsprozesse | Anwendungen Fachliche Komponenten | Objekte des Anwendungsbereichs |
| Technisches Konzept | Unterstützung der Zusammenarbeit | Koordination und Verteilung der Informatikmittel | Klassen Komponenten Schnittstellen | Datenbankschema(ta) |
| Realisierung | Kommunikations- und Suchdienste | Plattform(en) | Programme Bibliotheken Rahmenwerke | Datenbanksystem(e) |

## 5.3 Detailmodelle

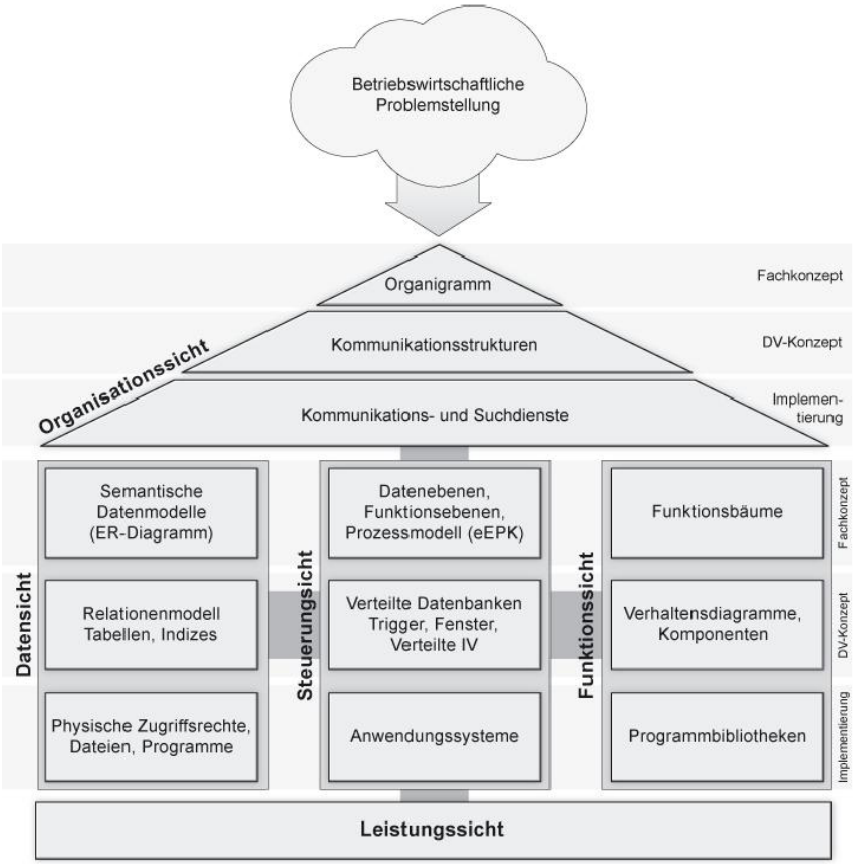
Fachkonzeptmodelle der verschiedenen Sichten:

- Die Schlüsselidee für die Kommunikation zwischen
  - Auftraggeber / Anwenderseite
  - Informatikseite
- (Organisationsmodelle)
- Prozessmodelle
- Funktionen- & Interaktionsmodelle
- Datenmodelle

## 5.4 ARIS – Architektur integrierter Informations-Systeme (von A.W. Scheer)




### 5.4.1 Sichten und Beschreibungsebenen von ARIS



### 5.4.2 Sichten von ARIS

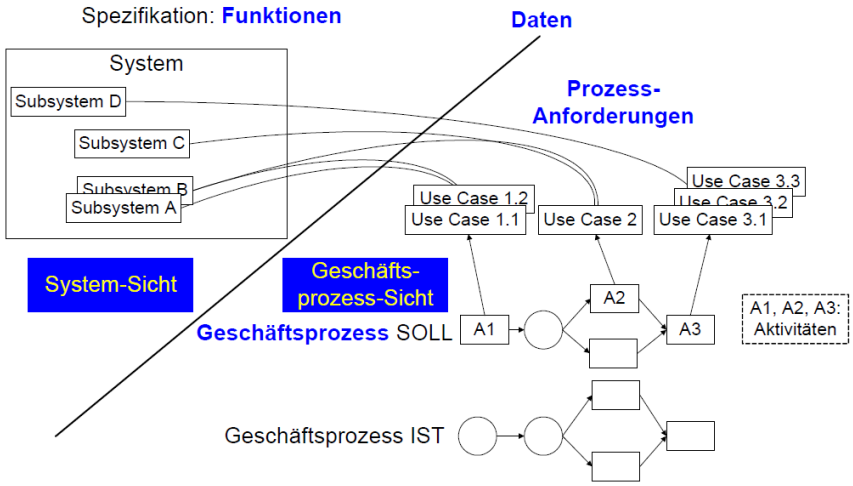
Organisationssicht (Wo? Wer?)
- Zuständige Stellen, Personen, ihre Kompetenz und Verantwortung (Organigramme)

Prozess-(Steuerungs)sicht (Wann?)
- Verbindungen zwischen verschiedenen Sichten (erweiterte ereignisgesteuerte Prozessketten)

Funktionssicht (Wie? Warum?)
- Notwendige Funktionen der Geschäftsprozesse und ihre Verbindung (Funktionshierarchiebäume)

Datensicht (Was?)
- Sachverhalte, Ereignisse und Bedingungen des Betriebs und seiner Prozesse (ER-Modelle)

### 5.4.3 Zusammenspiel der Sichten




## 5.5 Sichten (Interessen / Stakeholder)

| 1. Sicht | (Geschäfts-) Prozess |
|---|---|
| Inhalt | Beschreibt die Geschäfts- und Kernprozesse einer Unternehmung |
| Stakeholder | Manager der Führungsebene |

| 2. Sicht | Daten |
|---|---|
| Inhalt | Unternehmensweites Datenmodell der wichtigsten Geschäftsdaten |
| Stakeholder | Systemnutzer, Systemkonstrukteure |

| 3. Sicht | System |
|---|---|
| Teilsicht:<br>Inhalt:<br>Stakeholder: | Funktion<br>Das „Was“ (Spezifikation) der Lösung<br>Systemnutzer, Fachbereiche |
| Teilsicht: | Konstruktion |

| | |
|---|---|
| Inhalt: | Das „Wie“ der Lösung |
| Stakeholder: | Systemkonstrukteure |
| Teilsicht: | Verteilung / Betrieb |
| Inhalt: | Das „Wo“ und „Womit“ der Lösung |
| Stakeholder: | IT-Betrieb |

| 4. Sicht | Prozess-/Funktions-Integration |
|---|---|
| Inhalt | Beschreibt das Zusammenspiel von Prozessen und Funktionen |
| Stakeholder | Manager der operativen Ebene, Systemnutzer |

Die Prozessunterstützung durch eine IT-Lösung kann unvollständig sein; Use Cases sind dann nur „Durchstiche“ auf die durch IT-Funktionen unterstützten Funktionsschritte des Prozesses. Zu testen ist seitens der Prozessnutzer der gesamte Prozess, d. h. die Integration dieser Funktionsschritte in dem Gesamtprozess. Dies ist eine Sicht, die oft vernachlässigt wird, aber sehr wichtig ist.

# 6 UML-Übersicht

Die Unified Modelling Language UML bietet eine einheitliche Notation und Semantik der Modellierungselemente, die auf einem Metamodell begründet ist. Die Beschreibung einer Entwicklungsmethode ist nicht Teil von UML.
Die UML wird von der Object Management Group OMG als Standard gepflegt. Ausserdem gibt es einen entsprechenden ISO/IEC Standard 19501:
http://www.iso.org/iso/catalogue_detail.htm?csnumber=32620

Die Notationsübersicht der verschiedenen Diagramme findet man im PDF uml-2-Notationsuebersicht-oose.de.pdf

## 6.1 Anwendungsfalldiagramm (Use case diagram)

Ein Anwendungsfalldiagramm zeigt Akteure, Anwendungsfälle und die Beziehungen zwischen diesen Elementen.

### Beschreibung

Es beschreibt die Zusammenhänge zwischen einer Menge von Anwendungsfällen und den daran beteiligten Akteuren. Es bildet somit den Kontext und eine Gliederung für die Beschreibung, wie mit einem Geschäftsvorfall umgegangen wird.

Fast wichtiger als das Diagramm selbst ist die Beschreibung dazu.

→ unterstützen die Kommunikation

### Praxis

In der Praxis unterscheidet man drei verschiedene Arten von Anwendungsfällen:

1. Geschäftsanwendungsfälle
2. Systemanwendungsfälle
3. Sekundäre Anwendungsfälle

### Notation

| Darstellung | Bezeichnung | Beschreibung |
|---|---|---|
| | Akteur | Benutzer, System, der/das an Interaktion beteiligt ist |
| | Systemkontext-diagramm | ein Anwendungsfalldiagramm,. das alle Akteure dieses Systems zeigt |
| | Realisierung | Beziehung zwischen einem Element, das eine Anforderung beschreibt und einem Element, das diese Anforderungen umsetzt |
| | Spezialisierung | Beziehung allgemein → speziell |
| | Enthältbeziehung (Include) | bindet einen anderen Anwendungsfall als logischen Teil von diesem ein |
| | Erweiterungs-beziehung (Extend) | drückt aus, dass ein Anwendungsfall unter bestimmten Umständen an einer bestimmten Stelle (Extension point) durch einen anderen erweitert wird |
| | Assoziation | Relation zwischen einem Akteur und einem Anwendungsfall in einem Anwendungsfallmodell |
| | Anwendungsfall | beschreibt anhand eines zusammenhängenden Arbeitsablaufes die Interaktionen mit einem System. Wird stets von einem Akteur initiiert. |

| | | |
|---|---|---|
| | Geschäfts-anwendungsfall (business use case) | beschreibt einen geschäftlichen Ablauf, wird von einem geschäftlichen Ereignis ausgelöst und endet mit einem Ergebnis, das für den Unternehmensweck oder die Gewinnerzielungsabsicht direkt oder indirekt einen geschäftlichen Wert darstellt |
| | System-anwendungsfall (System use case) | beschreibt das für aussen stehende Akteure wahrnehmbare Verhalten eines Systems |
| | Sekundärer Anwendungsfall | |
| | Abstrakter Anwendungsfall | |
| | Anforderung, Feature & Co | |
| | Anwendungs-fallszenario | mögliche Ausprägung eines Anwendungsfalls |

### 6.1.1 Beschreibung eines Anwendungsfalls

Tabelle und Checkliste in „Die UML-Kurzreferenz 2.3 für die Praxis", S. 34f

## 6.2 Klassendiagramm

Ein Klassenmodell beschreibt, welche Klassen existieren und in welchen Beziehungen sie zueinander stehen.

### Praxis

Klassenmodelle werden für verschiedene Zwecke verwendet, bspw. mit Geschäfts- oder Fachklassen, um fachliche Begriffsmodelle darzustellen oder mit Designklassen, um die Struktur eines Lösungskonzepts abzubilden.

### Notationselemente

- Klasse
- parametrisierbare (generische) Klasse
- abstrakte Klasse
- Objekt
- Attribut
- Operation
- Verantwortlichkeit
- Enumeration
- Schnittstellen
- Stereotyp
- Notiz

## 6.3 Aktivitätsdiagramm

Ein Aktivitätsdiagramm beschreibt einen Ablauf und wird definiert durch verschiedene Arten von Knoten, die durch Objekt- und Kontrollflüsse miteinander verbunden sind.

### Praxis

→ Abläufe beschreiben

### Notationselemente

- Kontrollknoten
- Objektknoten
- Partitionen
- Signale und unterbrechbare Bereiche
- Mengenverarbeitungsbereiche

## 6.4 Zustandsdiagramm

Ein Zustandsdiagramm zeigt eine Folge von Zuständen, die ein Objekt im Laufe seines Lebens einnehmen kann und aufgrund welcher Stimuli Zustandsänderungen stattfinden.

### Notationselemente

- Zustand
- Ereignis oder Zustandsübergang (Trigger)
- Unterzustand
- Protokollautomat

## 6.5 Objektdiagramm

Ein Objektdiagramm ist ein Schnappschuss der Objekte im System zu einem bestimmten Zeitpunkt mit den augenblicklichen Werten.

### Beschreibung

Wird eher selten verwendet und eignet sich dazu bespielhaft, ein zur Laufzeit existierendes Objektnetz zu visualisieren.

## 6.6 Paketdiagramm

Pakete sind Ansammlungen von Modellelementen beliebigen Typs, mit denen das Gesamtmodell in überschaubare Einheiten gegliedert wird.

### Beschreibung

Pakete können verschiedene Modellelemente enthalten, z.B. Klassen und Anwendungsfälle. Sie können hierarchisch gegliedert werden, können also selbst wieder Pakete enthalten.

## 6.7 Komponentendiagramm

Eine Komponente (Component) ist eine spezielle Klasse, die eine austauschbare Einheit in einem System repräsentiert, deren Bestandteile gekapselt sind.
Die Komponente stellt ihre öffentliche Funktionalität über Schnittstellen zur Verfügung. Funktionalität, die von aussen benötigt wird, wird ebenfalls über Schnittstellen definiert.

### Beschreibung

Eine Komponente ist ähnlich wie eine Klasse instanzierbar und kapselt komplexes Verhalten. Mit ihr werden Einheiten gebildet, die eine hohe fachliche Kohärenz haben.

## 6.8 Sequenzdiagramm

Eine Sequenz zeigt eine Reihe von Nachrichten, die eine ausgewählte Menge von Beteiligten (Rollen und Akteuren) in einer zeitlich begrenzten Situation austauscht, wobei der zeitliche Ablauf betont wird.

## 6.9 Beziehungen

Übersicht in „Die UML-Kurzreferenz 2.3 für die Praxis“, S. 75

# 7 Prozessmodellierung

## 7.1 Geschäftsprozess

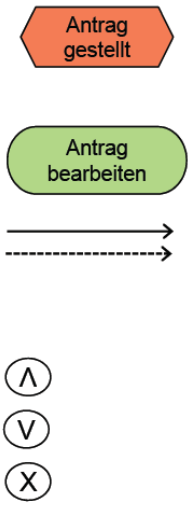
Ein Geschäftsprozess ist ein Ablauf von Aktivitäten, die der Erzeugung eines Produktes/einer Dienstleistung dienen. Er wird durch ein oder mehrere Ereignisse gestartet und durch ein oder mehrere Ereignisse abgeschlossen. Es liegt eine Organisationsstruktur zu Grunde.


### 7.1.1 Prozess (Definition)

Ein Prozess ist ein allgemeiner Ablauf mehrerer Schritte,bei denen es sich um Aufgaben, Ausführungen, Arbeitsschritte oder Ähnliches handeln kann. Zwischen diesen Prozessabschnitten bestehen bestimmte Abhängigkeiten.

Ein typischer Geschäftsprozess umfasst:
(1) Startereignis (Auslöser) à auch mehrere
(2) Aktivität(en)
(3) Zerlegung
(4) Sequenz
(5) Parallelität
(6) Abschlussereignis(se)

## 7.2 EPK




Ereignis: Anfang oder Schlussereignis, Zwischenereignis → Zustand des Prozess verändert sich

Funktion: Aktivität oder Prozessschritt

Steuerfluss / Datenfluss: definiert Abhängigkeit zwischen Ereignissen und Funktionen

Konnektoren: Verzweigung oder Zusammenführung von steuerflüssen

UND

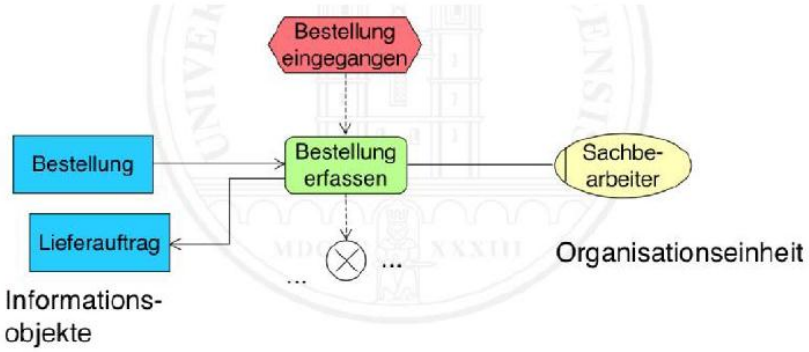
ODER

Exlusives ODER (XOR)

- Grundprinzip: auf ein Ereignis folgt immer eine Funktion und umgekehrt
- Im deutschsprachigen Raum weit verbreitet
- Wird insbesondere beim Einsatz von SAP zur Modellierung der Geschäftsprozesse eines Unternehmens eingesetzt
- Erweiterung mit Informationsobjekten und Organisationseinheiten möglich (EEPK)

### 7.2.1 eEPK

- Zu jeder Funktion wird zusätzlich erfasst
  - die beteiligten Informationsobjekte
  - die ausführende Einheit in der Organisation




## 7.3 Modellbildung

*Wie beginne ich?*

- *Immer mit einem Ereignis: Feststellen, welche(s) Ereignis(se) den Prozess anstösst und dieses Ereignis modellieren*

Ich habe ein Ereignis modelliert – wie weiter? Nach Eintreten dieses Ereignisses …

- … ist genau eine Transformation auszuführen: Diese Transformation als Funktion modellieren und direkt an das Ereignis anschließen
- … sind mehrere Transformationen auszuführen, die voneinander unabhängig ausgeführt werden können: Mehrere Funktionen modellieren und mit UND-Konnektor an auslösendes Ereignis anschließen

Ich habe eine Funktion modelliert – wie weiter?

- Genau ein Ereignis markiert den Abschluss genau dieser einen Funktion: dieses Ereignis modellieren und an die Funktion anschließen
- Genau ein Ereignis markiert den gemeinsamen Abschluss mehrerer Funktionen: diese Funktionen mit einem UND Konnektor an das Ereignis anschließen
- Genau ein Ereignis markiert den Abschluss einer beliebigen Funktion aus einer Gruppe von Funktionen: diese Funktionen mit einem ODER-Konnektor an das Ereignis anschließen
- Die Funktion wird durch genau eines von mehreren möglichen Ereignissen abgeschlossen: Ereignisse alle modellieren und die Funktion mit einem Exklusiv-ODER-Konnektor an diese Ereignisse anschliessen.
- Die Funktion wird durch mehrere Ereignisse, die einzeln oder gemeinsam auftreten können, abgeschlossen: alle Ereignisse modellieren und die Funktion mit einem ODER-Konnektor an diese Ereignisse anschließen

Auf mehrere Ereignisse folgt eine gemeinsame Funktion – wie weiter?

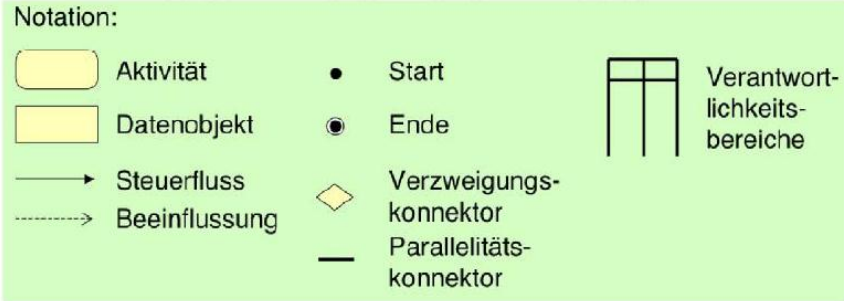
→ Die Ereignisse über einen Konnektor an die Funktion anschließen

- Die Ereignisse schließen sich gegenseitig aus: Exklusiv-ODER
- Die Ereignisse müssen alle eingetreten sein: UND
- Mindestens eines der Ereignisse muss eingetreten sein: ODER
- Es liegt eine Mischform vor: Konnektoren passend kaskadieren
- Mein Prozess wird zu gross – was tun?
    - Subprozesse bilden
    - In mehrere Prozesse aufteilen und über Wegweiser verbinden

## 7.4 Weitere Modellierungssprachen

### 7.4.1 UML

- Funktionen werden als Aktivitäten modelliert
- Ereignisse werden nur bei Fallunterscheidungen explizit modelliert
- Organisationseinheiten und Informationsobjekte sind modellierbar
- Parallelverarbeitung entspricht den UND-Konnektoren in EPKs




### 7.4.2 BPMN – Business Process Modeling Notation

- Die BPMN wurde 2002 durch den IBM-Mitarbeiter Stephen A. White erarbeitet und später von der Business Process Management Initiative (BPMI), einer Organisation, die Standards im Bereich der Geschäftsprozessmodellierung definiert hatte, veröffentlicht.
- BPMN ist heute eine gängige Modellierungssprache für den Entwurf von Workflows in Unternehmungen, welche in der Praxis durch sogenannte Workflow-Engines ausgeführt werden können.

| Symbol | Benennung | Bedeutung |
|---|---|---|
| | Aktivität (atomar) | Eine Aktivität (Activity) beschreibt einen Vorgang, der durch das Unternehmen ausgeführt wird. Sie kann atomar (task) oder zusammengesetzt sein, also Unterprozesse (subprocesses) enthalten. |
| | Aktivität (mit Unter-prozessen) | |
| | Start-Ereignis<br>Zwischenereignis<br>End-Ereignisse | Ereignisse (Events) sind Geschehnisse, die während eines Prozesses auftreten. Sie können auslösend sein oder das Ergebnis einer Aktivität. Es gibt drei grund-legende Typen (start, intermediate und end) und weitere Spezialfälle. |
| | Entscheidung (Gateway) | Gateways sind Synchronisationspunkte im Prozessverlauf. Sie entscheiden über den weiteren Verlauf des Prozesses. Es gibt mehrere Gateway-Typen: XOR, OR, AND und Eventbasierte Entscheidung. |
| | Kontrollfluss (Sequence flow) | Der Kontrollfluss beschreibt den zeitlichen Ablauf der Aktivitäten im Prozess |
| | Nachrichtenfluss (Message flow) | Der Nachrichtenfluss beschreibt den Austausch von Nachrichten zwischen zwei Objekten (Aktivitäten, Ereignisse oder Entscheidungen). |
| | Verbindung (Association) | Die Verbindung zeigt an, dass Daten, Texte oder andere Objekte dem Kontrollfluss verbunden sind, z.B. Input oder Output einer Aktivität. |
| Name | Datenobjekt (Data Object) | Das Datenobjekt zeigt an, welche Informationen/Daten als Input benötigt bzw. Output einer Aktivität sind |

# 8 Funktionsmodellierung

## 8.1 CRC-Karten

| Class Name RegistrationManager | |
|---|---|
| Responsibilities | Collaborations |
| Teilnehmer verwalten | Kurse |
| Kurse verwalten | Teilnehmer |

Ist eine Programmieraufgabe
- übersichtlichen und klein
- hat sie ein geschlossene Problemstellung
- und die Anwendung ist nicht verteilt

»So kann eine Subsystem- und Komponentenbildung entfallen und
»es lässt sich direkt aus der Spezifikation ein fachliches Klassenmodell erstellen.

Eine einfache Technik dafür ist es
1. mit Hilfe der Verb/Substantiv-Methode Klassen zu finden
2. mittels CRC-Karten deren Zuständigkeiten und Beziehungen festzuhalten
3. damit ein fachliches Klassenmodell zu erstellen
4. und dessen Vollständigkeit und Korrektheit mit dem Durchspielen von Szenarien zu verifizieren

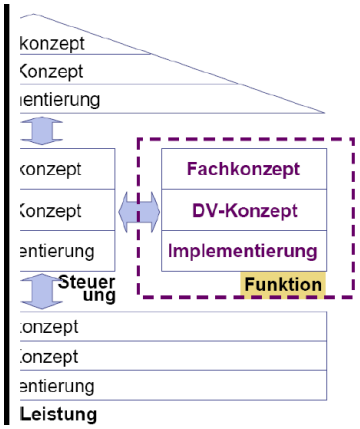
## 8.2 Dekomposition grösserer Systeme

Ist eine Programmieraufgabe
- nicht übersichtlichen und klein
- hat sie keine geschlossene Problemstellung
- oder die Anwendung ist verteilt
  (d.h. eigentlich fast immer im richtigen Leben)

» lässt sich ein fachliches Klassenmodell nicht direkt aus der Spezifikation erstellen,
» muss vorgängig eine Subsystem- und Komponentenbildung gemacht werden.

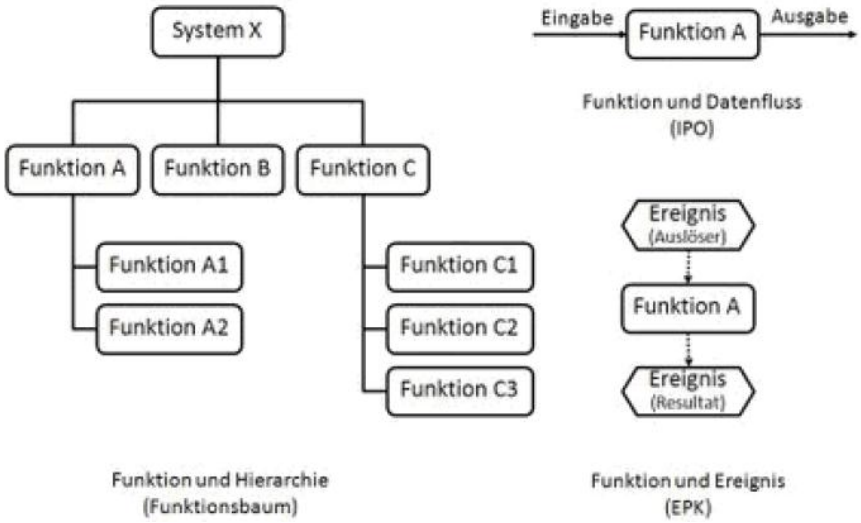
### 8.2.1 Grundlagen der Dekomposition

Grundidee der Funktionsmodellierung ist die Komplexitätsreduktion bei der Beschreibung von Informationssystemen durch die Zerlegung in einzelne, abgegrenzte Teileinheiten.




Dieses Bestreben betrifft
- die fachliche Beschreibung in der Systemanalyse
- die programmtechnische Realisierung eines Systems In diesem Sinne sind Funktionen ein wichtiges Element sowohl des Fachkonzepts (Funktions-hierarchie) als auch des DV-Konzeptes (Komponentenbildung) von Informationssystemen.

Ein grundsätzliches Anliegen der Funktionsmodellierung ist es, ein Informationssystem in verschiedene sachlich abgeschlossene Teileinheiten zu zerlegen.




Das Konzept der Funktion ist in etlichen Methoden der Systementwicklung von Bedeutung, die verschiedene Aspekte von Funktionen modellieren.

Die funktionale Zerlegung (Funktionsmodell) entspricht dem Informationsmodell auf Ebene der Funktionalität:

Sie stellt das System dar als aus Bausteinen zusammengesetzt, die eine klar eingrenzbare und beschriebene Funktionalität abdecken.

Auf Ebene der Systemarchitektur sind das die Software-Systeme, darunter kommen die

- Subsysteme
  - Komponenten
    - Subkomponenten
      - Klassen etc.

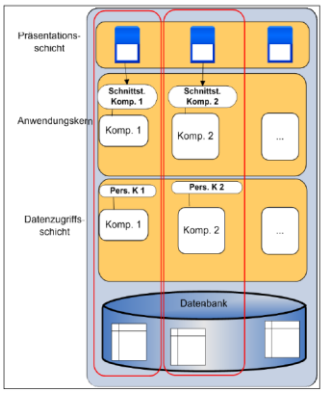
Wir unterscheiden im Wesentlichen zwei Architektur-Grundprinzipien:

- Trennung der Verantwortlichkeiten Separation of Concerns
- Geheimnisprinzip Information Hiding

Zusammengenommen beschreiben sie das Ideal der Delegierung einer Aufgabe: *„Dafür ist x zuständig, und ich brauche mich um keine Details zu kümmern.*

- Die Trennung der Verantwortlichkeiten „Separation of Concerns“ führt im Idealfall zu einer doppelten Modularisierung:
- In ***fachliche Komponenten*** durch die Trennung der fachlichen Verantwortlichkeiten (senkrechte Gliederung)
- In ***Schichten*** oder durch die Trennung der technischen Verantwortlichkeiten (waagerechte Schichtung)

**Separation of Concerns**



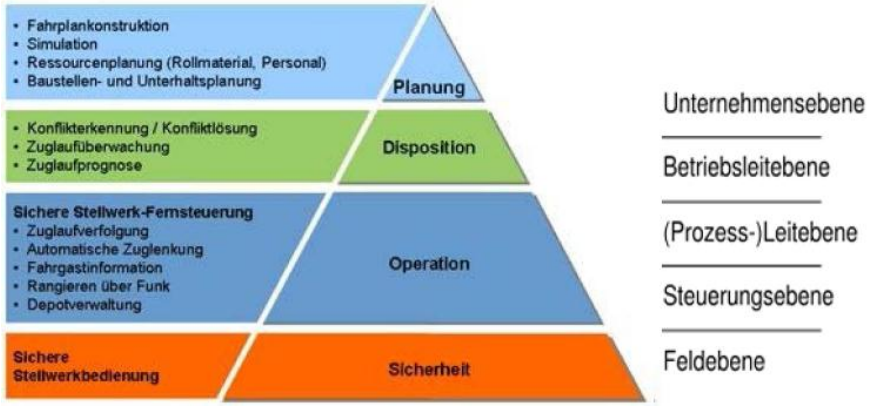

Beispiel: 3-Schichten-Architektur

Geheimnisprinzip („Information Hiding“):

Man zeigt einem Nutzer nur diejenigen Informationen, die er zur Erfüllung seiner Aufgabe braucht. Beispiele:

- OOP: Alle Daten sind „private“, Zugriff auf Daten von aussen nur über dedizierte Methoden
- Fassade, die den Zugriff auf ein ganzes Subsystem regelt und das Subsystem vor direktem Zugriff schützt (z.B. ein Interpreter)
- Schichten-Architektur: Schicht *n* sieht nur Schicht *n-1*, die anderen bleiben verborgen.

Separation of Concerns & Information Hiding werden in Referenzarchitekturen exemplarisch berücksichtigt, z.B:
Leittechnik-Pyramide

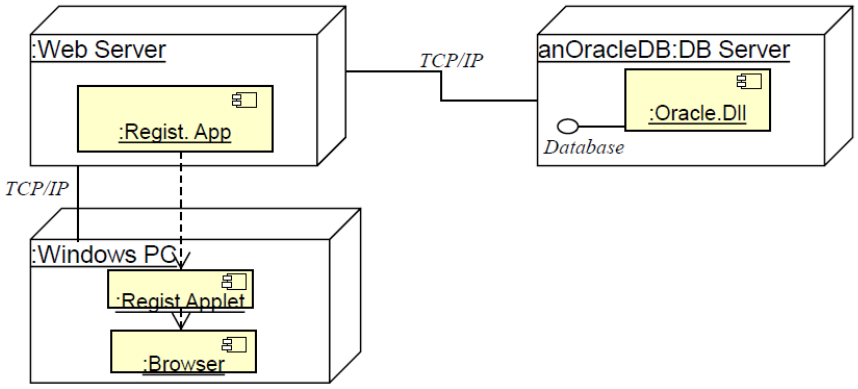



## 8.2.2 Techniken & Tools

- Systemdekomposition ist kein deterministischer Prozess, d.h. es gibt nicht die richtige Lösung.
- Wichtig ist deshalb das Denken in Varianten und deren Vor- und Nachteile im gegebenen Kontext abzuwägen.
- Die Varianten-Diskussion wird durch eine allgemein verständliche und standardisierte Darstellung erst sinnvoll möglich.
- UML Aktivitäts-, Deployment- und Kompo-nenten-Diagramme bieten sich hier an.

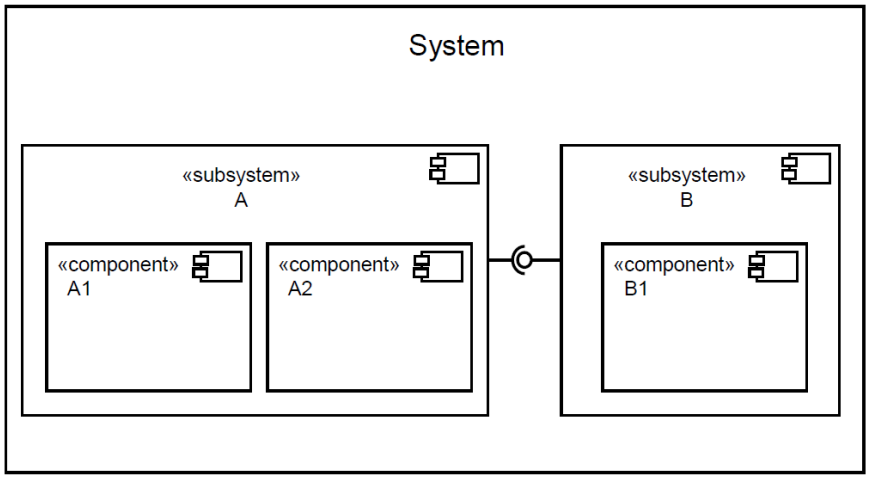
## 8.2.3 Deployment Diagramm

Deployment-Diagramme unterstützen die Modellierung verteilter Systeme. Subsysteme und Komponenten können Knoten zugeordnet werden. Die Kommunikationsverbindungen zwischen den Knoten kann dargestellt werden und bei Bedarf lassen sich Abhängigkeiten zwischen den Elementen aufzeigen.




### 8.2.4 Komponenten Diagramm

Komponentendiagramme unterstützen die Strukturierung von Systemen in Subsysteme und Komponenten.




### 8.2.5 Software-Kategorien

Eine Software-Kategorie ist alle Software, die das Wissen (im Sinne von „Know-How") eines bestimmten Gebietes enthält. Beispiel: Das Wissen um die Besonderheiten einer bestimmten Datenbank.

„Was tut / kann / weiss das Ding??"

Software-Kategorien helfen erheblich bei der Komponentisierung eines Systems – sie sind sozusagen die Komponenten-Behälter.
Software-Kategorien können verfeinert werden.

Der erste Schritt bei der funktionalen Zerlegung eines Systems ist die Aufstellung der Software-Kategorien und die Analyse der Abhängigkeiten dazwischen.

- In der Praxis werden häufig erst die Komponenten erdacht und dann die Abhängigkeiten geprüft (wenn überhaupt). Das zieht meist eine schlechte Entwurfsqualität nach sich.

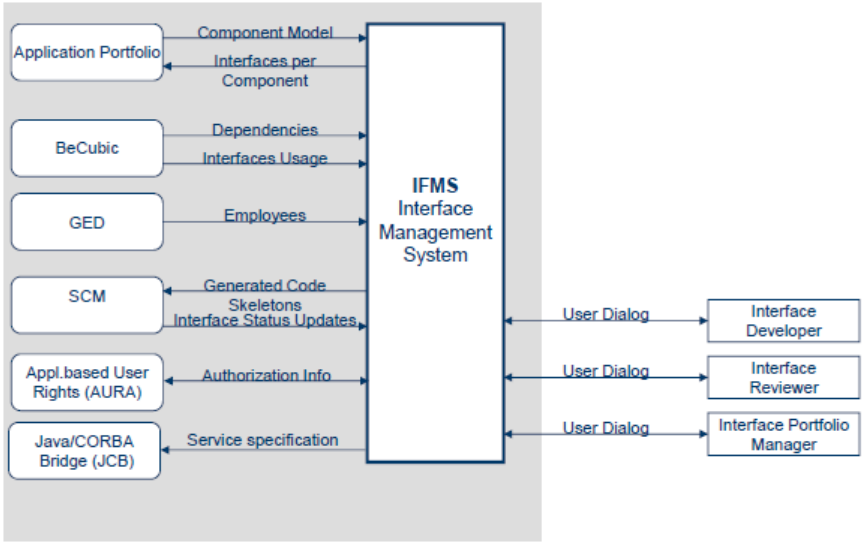
Die Kategorien liefern bereits eine erste grobe Aufteilung des Systems in funktionale Bausteine.

# 9 Kontextdiagramm

Das Kontextdiagramm veranschaulicht in der Konzeptionsphase das Umfeld des zu realisierenden IT-Systems. Insbesondere wird klar abgegrenzt, was zum Aufgabenbereich des IT-Systems gehört und was nicht.

Weiterhin veranschaulicht es grob, welche Informationen in das System hinein fließen und welche Informationen das System liefern kann. Das Kontextdiagram wird zu einer sehr frühen Phase der Systemkonzeption eingesetzt.

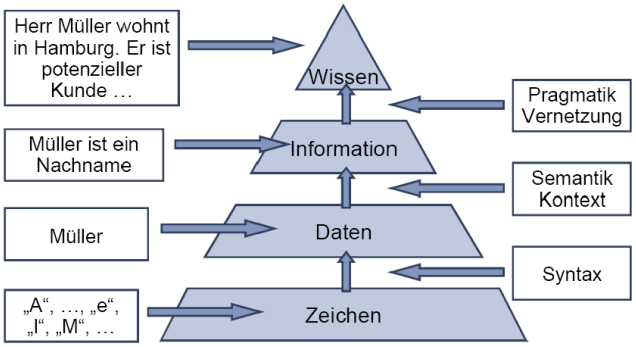
Beispiel IFMS



# 10 Datenmodellierung

siehe auch Leitfaden Informations- und Datenmodell.pdf der HSLU-T&A

## 10.1 Begriffshierarchien




### 10.1.1 Begriffe

**Daten**
Angaben über Sachverhalte und Vorgänge aufgrund bekannter oder unterstellter Abmachungen in einer maschinell verarbeitbaren Form.

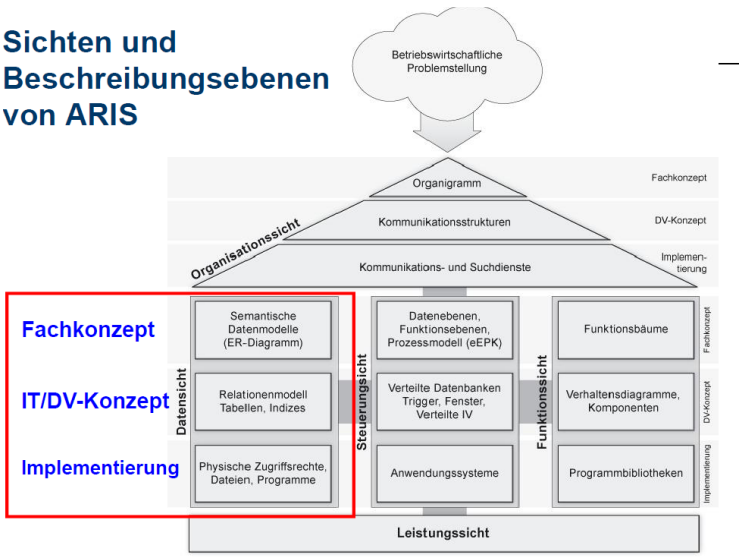
**Datenbank**
selbständige, auf Dauer und für den flexiblen und sicheren Gebrauch ausgelegte Datenorganisation

**Datenmodell**
Formale Beschreibung des Schemas, z. B. in Form eines Diagramms oder einer Datenstruktur.

- Strukturierte Darstellung der Daten der Diskurswelt mit einem formalen Beschreibungsmittel

**Datenmodellierung**
Prozess, der sicherstellen soll, dass eine Datenbasis zu jedem Zeitpunkt ein korrektes Abbild der Diskurswelt wiedergibt.

## 10.2 Sichten und Beschreibungsebenen von ARIS



## 10.3 Semantische Datenmodellierung

Klassifikation der Datenmodelle anhand ihrer Nähe zur Realwelt

### Semantisches Datenmodell

- Brücke zwischen Realwelt und dem logischen Datenmodell
- losgelöst vom einzusetzenden Datenbanksystem
- Realitätsausschnitt wird abstrahierend in einem Modell abgebildet

### Logisches Datenmodell

- ebenfalls unabhängig von der physischen Repräsentation
- Ausrichtung an der für die Speicherung einzusetzenden Datenbanktechnologie

### Physisches Datenmodell

- Aspekte der physischen Speicherung und Speicheroptimierung

In der Praxis: Semantisches = Logisches Datenmodell

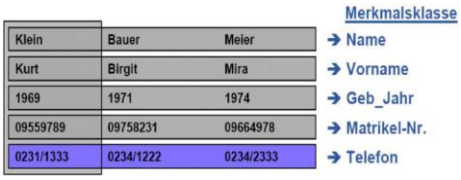
### 10.3.1 Informationsstrukturmodell erstellen

Schritt 1

Informationsobjekte (IO) = Objekte der Wahrnehmung oder Vorstellung,

- die vom Menschen beschrieben und unterschieden werden können
- die in Bezug auf die gegebene Problemstellung relevant sind


- Merkmale charakterisieren die zugehörigen Informationsobjekte
- Gleichartige Merkmale werden als Ausprägungen einer gemeinsamen Merkmalsklasse verstanden
- Alle Merkmale, die einem Informationsobjekt zuzuordnen sind, werden als charakterisierende Merkmalskombination bezeichnet



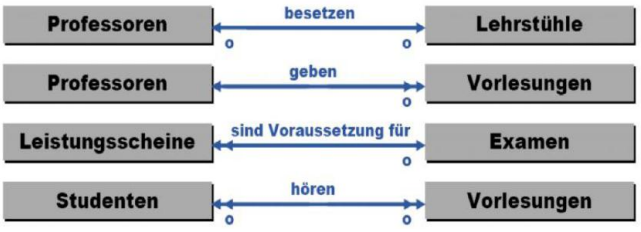

Schritt 2

- Gleichartige Informationsobjekte bilden Informationsobjektklassen (IOK) (Generalisierung)
- Informationsobjekte sind gleichartig, wenn sie durch Merkmalskombinationen derselben Merkmalsklassenkombination charakterisiert werden


Schritt 3

Verknüpfungstypen

- Nach der Kardinalität 1:1, 1:N, N:M
- Nach der Operationalität
  feste Verknüpfungen oder optionale Verknüpfungen




- Komplexe Verknüpfungen: Rollenkonzept
  - Person kann bspw. Student und Assistent an einer Hochschule sein

Vorgehen bei der Informationsstrukturierung

- Definition der Informationsobjekte und ihrer Merkmale (Eigenschaften)
- Definition der Informations-KLASSEN unter Berücksichtigung der
  - relevanten Informationsobjekte
  - der charakterisierenden und identifizierenden Merkmalsklassen und -kombinationen
  - vollständige Auflistung der relevanten Merkmale
- Definition der Verknüpfungen zwischen den ausgewählten Informationsklassen unter Berücksichtigung der
  - Qualifizierung der Verknüpfungen
  - Auflösung komplexer Verknüpfungsstrukturen

## 10.4 Entity Relationship Modell (ER Modell)

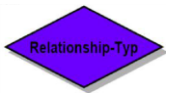
### 10.4.1 Strukturelemente

Eine Entity ist ein unterscheidbares Objekte der Realwert, das ein reales Objekt oder eine gedankliche Abstraktion darstellen kann.
→ konkrete Informationsobjekte, z.B. Student Hans


Ein Entity-Typ ist eine Zusammenfassung von gleichartigen Entities, welche die gleichen Eigenschaften besitzen. Entity-Typen stehen für eine Menge von Objekten-
→ Informationsobjektklassen, z.B. Studenten


Die Eigenschaften eines Entity-Typs werden Attribute genannt. Jedes Attribut hat einen bestimmten Wertebereich. Ein Attribut oder eine Attributsmenge, das jedes Entity eines Entity-Typs eindeutig identifiziert, heisst Schlüssel bzw. Schlüsselkandidat.
→ Merkmalsklasse, z.B. MatrikelNr von Student

Eine Beziehung (relationship) ist eine Verknüpfung von zwei oder mehreren Entities.
*Bsp: Student Hans entleiht ein oder mehrere Bücher*
→ Verknüpfungen

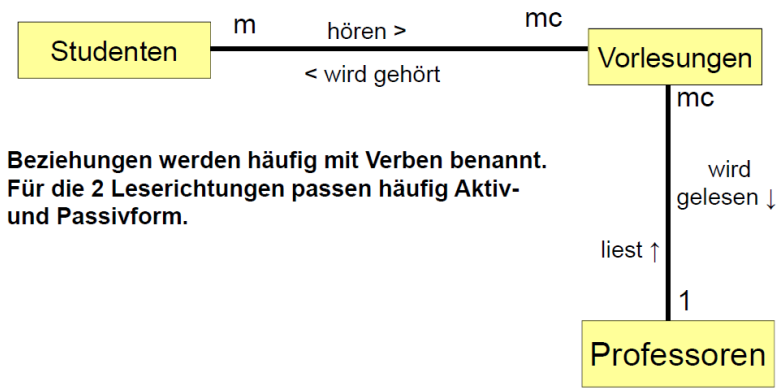



Ein Beziehungstyp (relationship-typ) ist eine Zusammenfassung von gleichartigen Beziehungen, welche zwei Entity-Typen miteinander verknüpfen.
Bsp: Studenten entleihen Bücher
→ Verknüpfungsklassen

## 10.5 Vereinfachtes ER-Modell

- Kardinalität wird mit „1-m-c" dargestellt
- Die Relationship-Typ werden weggelassen, an ihre Stelle tritt die Bezeichnung der Beziehungen der beiden Entitäten
- Die Definition der Attribute wird in einem separaten Dokument erstellt
- Pro Entitätsmenge und pro Attribut gibt es eine kurze Beschreibung
- Das Schlüsselattribut wird ausgezeichnet

Beispiel:




## 10.6 Beziehungen zwischen Entitätsmengen

Eine Beziehung resp. Assoziation (EM1, EM2) legt fest, wieviele Entitäten aus EM2 einer Entität aus EM1 zugeordnet werden können.

| Verknüpfungsstyp ( EM1, EM2) | | Entitäten aus EM2, die jeder Entität aus der Menge EM1 zugeordnet sind: |
|---|---|---|
| 1: | einfache Verknüpfung | genaue eine |
| c: | konditionelle Verknüpfung | keine oder eine (c = 0/1) |
| m: | multiple Verknüpfung | mehrere (m >= 1) |
| mc: | multipel-konditionelle Verknüpfung | keine, eine oder mehrere (mc >= 0) |

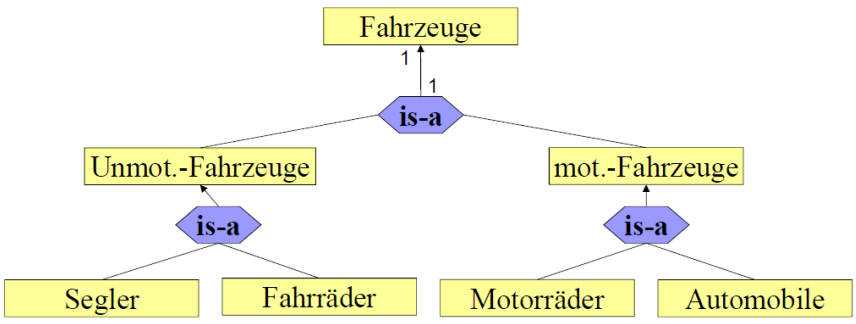
### 10.6.1 Spezialisierung, Generalisierung

Spezialisierung

Ist die klassische IS-A-Beziehung, d.h. der Entity-Typ mot. Respektive unmot. Fahrzeug ist eine Spezialisierung von Fahrzeug. Kennzeichnend ist die Vererbung von Eigenschaften.

Generalisierung

Bei der Generalisierung wird in dem Sinne verallgemeinert, dass der Generalisierungs-Entity-Typdurch die Vereinigungsmenge der Instanzen mehrerer Entity-Typengebildet wird.



## 10.7 Informationsmodell

- Das Informationsmodell in einem Grobkonzept beschreibt alle wichtigen Informationsobjekte und deren Zusammenhänge, welche für das Projekt, das zu bauende SW-System und für das Projektverständnis eine zentrale Rolle spielen.
- Das Informationsmodell ist eine abstrakte, vereinfachte Sicht der Realität. Es schafft eine gemeinsame Sprache und eine gemeinsame Sicht aller Beteiligten.

- Umfang:
    - 1 Schema; 8 – 14 Entitätsmengen inkl. Bezeichnung der Beziehungen
    - Kurzbeschreibung der Entitätsmengen in Prosa
    - Benennung und Kurzbeschreibung der 2-4 wichtigsten Attributen

# 11 Systemdesign

## 11.1 Übersicht

### 11.1.1 Einordnung des Systemdesigns

**Sichten/Ebenen: Matrixdarstellung**

| Sicht → / Ebene ↓ | (Geschäfts-) Prozess | Daten | System | | | Prozess-Funktions-Integration |
|---|---|---|---|---|---|---|
| | | | Funktion | Konstruktion | Verteilung / Betrieb | |
| Überblicks-Ebene | | | | | | |
| Grob-Ebene | | | | | | |
| Detail-Ebene | | | | | | |

3. Sicht: System

- Teilsicht Funktion: Das „Was" (Spezifikation) der Lösung
- Teilsicht Konstruktion: Das „Wie" der Lösung →Stakeholder: Systemkonstrukteure
- Teilsicht Verteilung/Betrieb: Das „Wo" und „Womit" der Lösung

- Das Ziel bei der Modellierung von IT-Lösungen ist letztlich die konkrete Umsetzbarkeit.
- Es werden nicht immer alle Ebenen modelliert, in manchen Fällen fehlt eine Ebene oder es werden zwei Ebenen in der Modellierung verschmolzen.
- Für die „Konstruktions"-Teilsicht wird das Ziel der konkreten Umsetzbarkeit je nach Kontext auf unterschiedlichen Ebenen erreicht.

### 11.1.2 Themenfelder im Systemdesign

- Einflussfaktoren und Vorgehen beim Systementwurf.
- Leitplanken, Muster und Kriterien für das Systemdesign
- Rund ein Dutzend Architekturaspekte wie z.B. Sicherheit, Persistenz, Usability etc. beeinflussen je nach zu realisierendem System die Entwurfsentscheidungen.
- Die Konstruktionsteilsicht kann aus verschiedene Perspektiven betrachtet werden, man spricht in diesem Zusammenhang von Architektursichten
  [Starke] unterscheidet:
  - Kontextsicht (bereits im Kapitel 10 behandelt)
  - Bausteinsicht
  - Laufzeitsicht
  - Verteilungssicht
- Zur Dokumentation und Kommunikation von Systementwürfen sind insbesondere UML Diagramme hilfreich.

## 11.2 Grundlagen des Systementwurfs

### 11.2.1 Einflussfaktoren und Vorgehen

Wesentliche Schritte, die zu einem guten Systemdesign führen sind:

1. Informationen sammeln
2. Systemidee entwickeln
3. Einflussfaktoren und Randbedingungen würdigen
4. Risiken identifizieren
5. Qualität explizit beschreiben
6. Lösungsstrategien entwickeln

Details siehe auch Kp13-1.pdf, Seiten 8 bis 15

*"Everything should be as simple as possible, but not simpler."*
*Albert Einstein*

*"Wenn du es in fünf Minuten nicht erklären kannst, hast du es entweder selbst nicht verstanden, oder es funktioniert nicht."*
*Eberhardt Rechtin, Mark Maier The Art of Systems Architecting*

### 11.2.2 Von der Idee zur Struktur

Komplexität

- natürliche Komplexität ist die Komplexität des zu Grunde liegenden Problems, das die Software lösen soll. → beherrschen
- künstliche Komplexität kommt bei der Entwicklung dazu und trägt nichts zur Lösung des Problems bei → vermeiden

Heuristiken

Systementwurf ist keine exakte Wissenschaft, die Kombination von Grundsätzen der Zerlegung, Architekturmuster, Entwurfsprinzipien und Heuristiken kommt Ihren Entwürfen zugute.

Heuristiken sind Regeln zum Umgang mit komplexen Aufgaben, entstanden aus verallgemeinerten und abstrahierten Erfahrungen. Darum können Heuristiken lediglich Hinweise geben auf dem Weg zum guten Systementwurf, aber ohne Erfolgsgarantie, denn Entwurf bleibt (zum Glück) eine kreative Tätigkeit.

- So einfach wie möglich
- Separation of concerns
- Auf Schnittstellen hin entwerfen
- Fehler berücksichtigen
- Lose Kopplung und hohe Kohäsion
- Das Offen-Geschlossen-Prinzip
- Keine zyklischen Abjängigkeiten
- Das Liskov-Substitutionsprinzip
- Dependency Injection

Details siehe Heuristiken.pdf

## 11.3 Dokumentation des Systementwurfs

### 11.3.1 Systemdesign kommunizieren

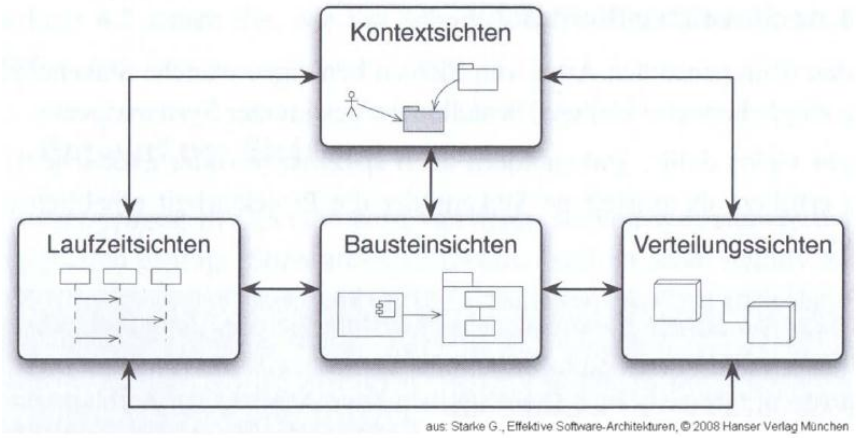
Beim Systementwurf werden wichtige Entscheidungen gefällt, die Einfluss auf Entwicklung, Betrieb und Wartung haben.

Entwurfsentscheidungen müssen deshalb den entsprechenden Stakeholders kommuniziert und erklärt werden:

- Auftraggeber / Nutzer
- Entwicklungsteam
- Betrieb und Wartung

Weil die Lebensdauer von IT-Systemen oft Jahrzehnte beträgt, muss deren Architektur so dokumentiert werden, dass Generationen von Entwicklern das System verstehen und an neue Bedürfnisse anpassen und erweitern können.

In einer einzelnen Darstellung lässt sich die Vielschichtigkeit und Komplexität eines Systemdesigns nicht ausdrücken.




Sichten erlauben die Konzentration auf bestimmte Aspekte, indem sie von Details abstrahieren, die für die gewählte Perspektive nicht von Bedeutung sind.

## Die Sichten im Detail

### Kontextsichten - Wie ist das System in seine Umgebung eingebettet?

- zeigt System als Blackbox in seinem Kontext aus der Vogelperspektive
  - Schnittstellen zu Nachbarsystemen
  - Interaktionen mit wichtigen Stakeholdern
  - wesentliche Teil der umgebenden Infrastruktur

### Bausteinsichten - Wie ist das System intern aufgebaut?

- zeigen die statischen Strukturen der
  - Architetkurbausteine des Systems
  - Subsysteme,
  - Komponenten und deren Schnittstellen.
- Top-Down entwickeln, ausgehend von der Kontextsicht
- letzte mögliche Verfeinerungsstufe der Zerlegung bildet der Quellcode
- Bausteinsichten
  - unterstützen Projektleiter und Auftraggeber bei der Projeküberwachung
  - dienen der Zuteilung von Arbeitspaketen
  - fungieren als Referenz für die Softwareentwickler

### Laufzeitsichten - Wie läuft das System ab?

Die Laufzeitsichten beschreiben, wie die Bausteine des Systems zur Laufzeit zusammenwirken. → dynamische Strukturen werden hier beschrieben

Verteilungssichten - In welcher Umgebung läuft das System ab?
Diese Sichten beschreiben die Hardwarekomponenten, auf denen das System abläuft. Dabei dokumentiert man-

- Rechner,
- Prozessoren,
- Netztopologien und -protokolle sowie
- sonstige Bestandteile der physischen Systemumgebung.

Die Infrastruktursicht zeigt das System aus Betreibersicht.

### 11.3.2 Entwurf der Sichten

Da sich die Sichten gegenseitig beeinflussen, gibt es keine zwingende Reihenfolge für deren Entwurf.

- Wenn ein bestehendes System erweitert oder umgebaut wird, ist die Bausteinsicht ein guter Ausgangspunkt.
- Wenn Verantwortlichkeiten und Zusammenspiel der Komponenten unklar ist, beginnen Sie mit der Laufzeitsicht.
- Wenn die Infrastruktur durch Randbedingungen eingeschränkt oder vorgegeben ist, entwirft man zuerst die Verteilungssicht.
- Allgemein ist die Kontextsicht ein sinnvoller Einstieg.

Details siehe Vorlesungsfolien IM-Kp13-2_Doku_Systementwurf.pdf

## 11.4 Typische Probleme beim Systementwurf

Unter Architekturaspekten verstehen wir Dinge, die Software-Architekten beim Systementwurf meistens beschäftigen.

Für viele dieser Aspekte gibt es gibt es daher typische Lösungsansätze auf die man für die konkrete Lösung aufbauen kann.

### 11.4.1 Architekturaspekte

1. Persistenz
2. Geschäftsregeln
3. Legacy
4. Verteilung
5. Kommunikation
6. GUI-Ablaufsteuerung
7. GUI-Ergonomie
8. Internationalisierung
9. Workflow-Management
10. Sicherheit
11. Logging
12. Fehlerbehandlung

1.Persistenz

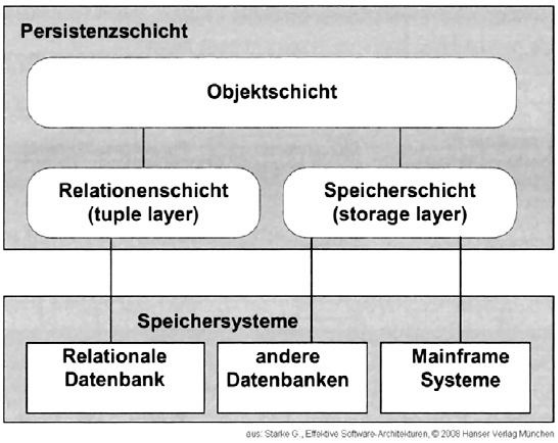
- effizienter Zugriff auf persistente Daten wird benötigt

Typische Probleme beim Entwurf sind

- Datenmodellierung (Performance vs. Normalisierung),
- Wahl eines DBMS,
- Trigger (Seiteneffekte),

- (modifizierender) Zugriff über Mainframe-Anwendungen,
- Strukturbrüche (impedance mismatch, OR-Mapping).

- Persistenzsicht
  - Persistenzschichten vermitteln zwischen Anwendung und unterschiedlichen Speichersystemen.
  - Fachklassen benutzen lediglich Methoden wie save(), read() oder delete().



aus: Starke G., Effektive Software-Architekturen, © 2008 Hanser Verlag München

## 2.Geschäftsregeln

Geschäftsregeln legen Abläufe in Abhängigkeit von bestimmten Bedingungen fest.

- if-then-else Konstrukte in den Fachklassen erschweren Verständlichkeit und Wartbarkeit eines Systems.
- Bessere Alternative: Geschäftsregeln in einer Regelsprache beschreiben. Eine RuleEngine führt dann die Fachlogik auf den Geschäftsobjekten aus.

## 3.Legacy

Legacy: alte bestehende Umgebung/Systeme eines Unternehmens

Der Eingriff in die Legacy soll mininmal invasiv gestaltet werden:

- Bestehende Schnittstellen unterstützen
- bestehende Daten beim Entwurf berücksichtigen
- Verantwortlichkeit alt/neu klar abgrenzen.

Lösungsansätze

- Dateitransfer
- Gemeinsame Datenbank
- Synchrone Applikationsintegration (RPC)
- Asynchrone App.Integration (Messaging, z.B. CORBA)

Minimal invasiven Ansatz durch Gateway, alternative durch Wrapper.

## 4.Verteilung

Entfernte Methodenaufrufe sind ressourcenintensiver, benötigen verteilte Objekte komplexere Infrastruktur und die Fehlerbehandlung ist über Systemgrenzen hinweg aufwändiger.

Tipps

- Verteilung nur wenn nötig (Kompexität)
- Verarbeitung möglichst nahe bei den betroffenen Daten
- Stabile Schnittstellen anstreben
- Lieber höhere Serverbelastung als höherer Datentransfer
- Zusatzaufwand für Test und Deployment einplanen

## 5.Kommunikation → grosse Bedeutung bei der Entwicklung verteilter Systeme

Fragen

- Synchron oder asynchron
- Stateful oder stateless

- Direkt oder via Middleware
- Stil (RPC, publish/subscribe, broadcast, poll, ...)
- nur TCP/IP oder zusätzlich IIOP, SOAP, HTTP, ...
- Datenkodierung, (little/big endian, Zeichensatz, ...)
- Semantik (aktuelle Werte oder nur Deltas)

## 6.GUI-Ablaufsteuerung

- Frage/Antwort-Systeme
- Objektorientierte Oberflächen
- MVC

## 7.GUI-Ergonomie

- Arbeitsmetaphern (Struktur der Arbeitsabläufe)
- Interaktionsstile (Art der Darstellung)
- Effektivität, Effizienz und Akzeptanz

- Strukturelle Wiederholung
  Darstellung und Navigation nach einheitlichem Muster
- Gruppe verwandter Dinge
  Auf einen Blick sind / +/-2 Dinge erkennbar
- Rückkehr an einen sicheren Ort
  letzter kosistenter Zustand immer erreichbar
- Statusanzeige
  immer an gleicher Stelle, symbolisch, nicht störend
- Fortschrittsanzeige
  wissen, ob das System noch arbeitet
- Mehrfach-Aktion
  gleiche Operation auf mehreren Objekten
- Verdichtete Information
  gleichartige Daten möglichst auf einer Ansicht
- Tabelle
  geordnete Daten möglichst Tabellarisch darstellen
- Hierarchie
  strukturierte Daten z.B. als Baum präsentieren
- Formular
  erwartete Eingabe durch formatierte Felder aufzeigen

## 8..Internationalisierung

Je nach Markt sind nebst Sprache, Währung, gängiger Formatierung etc. auch rechtliche (z.B. Datenschutz) und kulturelle Aspekte zu beachten.
Internationalisierung beeinflusst deshalb nicht nur die Architektur, sondern den gesamten Entwicklungsprozess.

## 9.Workflow-Management

Unter Workflow-Management versteht man die systemübergreifende Ablaufsteuerung und Koordination von Software-Systemen.

Grundsätzlich kann man drei Lösungsansätze unterscheiden:
1. Steuerung durch externes WMS bzw. BPMS
2. Manuelle Steuerung (mittels E-Mail oder Groupware)
3. Embedded Workflow mittels Zustandsautomat

## 10.Sicherheit

Insbesondere im Bereich eCommerce sind Sicherheitsaspekte von zentraler Bedeutung. Gesetzliche Vorgaben beschränken den Entscheidungs-Spielraum, z.B. Verschlüsselungsverbot vs. –Pflicht.

- Organisatorische Fragen
  - Firewalls / PKI / Hardware / ...
- Ziele

  - Vertraulichkeit / Integrität / Autentizität / Autorisierung, Verfügbarkeit (denial of service Angriffe)
- OSI-Schichten
  - Netzwerk (3) z.B. VPN, für Anwendungen transparent
  - Transport (4) z.B. SSL, Anwendung kennt Zertifikate
  - Anwendung (7) z.B. S/MIME E-Mail Anwendungen

## 11.Logging

Logging kann fachliche und/oder technische Protokollierung sein,
Tracing ist technische Protokollierung zu Debugging-Zwecken.
→ geeignetes Framework festlegen, z.B. log4j

- Loglevels und Kriterien definieren.
- Instrumentierter Code erleichtert die Fehlersuche

## 12.Fehlerbehandlung

Das Exception-Handling der Programmiersprache ersetzt nicht ein angemessenes Ausnahme- und Fehlerbehand-lungskonzept!

- Fragen
  - Welche Ausnahmen- und Fehlerkategorien gibt es?
    - Falsche / fehlende Eingaben
    - Fehler der Infrastruktur
    - Unzulässige interne Zustände
  - In welcher Form reagieren?
    - Recovery
    - Nachfragen (wo?)
    - Melden (wem?)
  - Testbarkeit der Fehlerbehandlung?

- Tipps
  - Fokus auf Schnittstellen, keine leeren Catch-Klauseln
  - Logging definieren