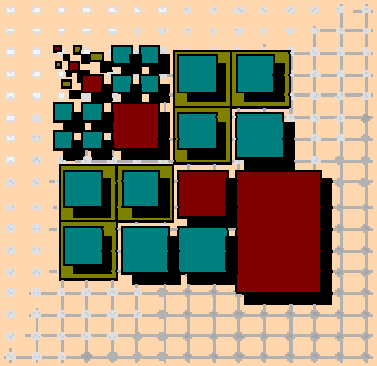
Softwaretechnik mit der UML

- Softwaresysteme & -technik
- Software-Entwicklungsprozesse
- Evolutionäre SW-Konstruktion
- Die Unified-Modeling-Language
- Zusammenfassung



Dipl.-Ing. Jörg Graf
FG „Verteilte Systeme“
Institut für Informatik
Universität Münster

Softwaresysteme

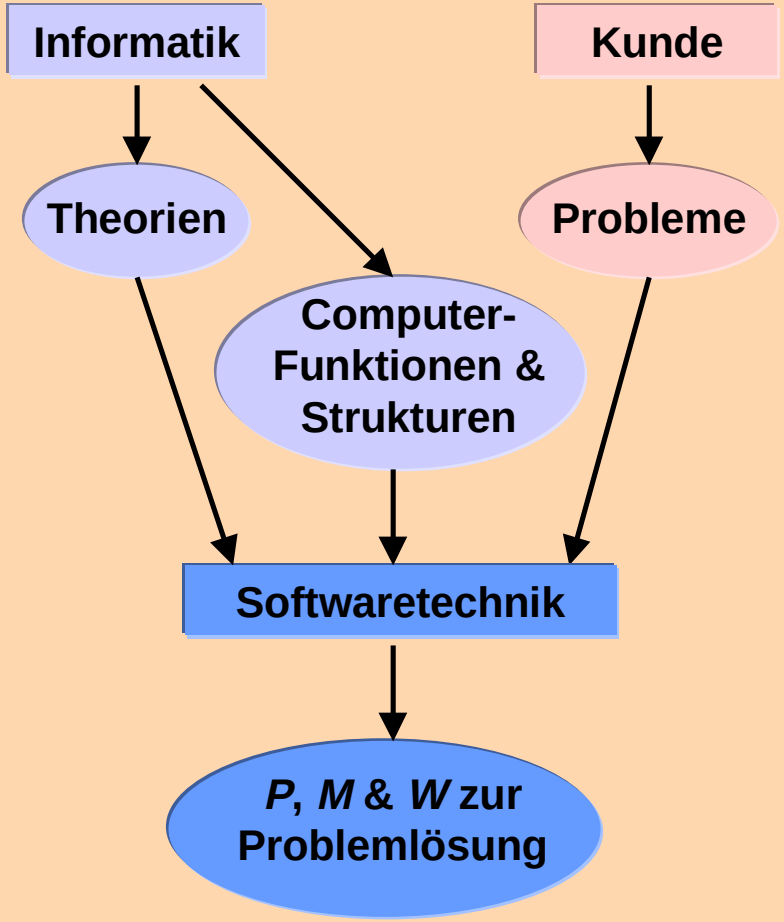
- Aus Softwarekomponenten zusammengesetzt

- Klassifikation
  - **allgemein ↔ individuell**
  - **zentralisiert ↔ verteilt**
  - **Echtzeit ↔ flexible Zeit**
  - **Computer ↔ eingebettet**
  - **kontroll- ↔ daten- ↔ berechungsintensiv**

- Angestrebte Qualtitätsmerkmale
  - **Benutzerfreundlichkeit**
  - **Korrektheit**
  - **Zuverlässigkeit**
  - **Effizienz**
  - **Wartbarkeit**

- Handlungsgegenstand der Softwaretechnik

Softwaretechnik

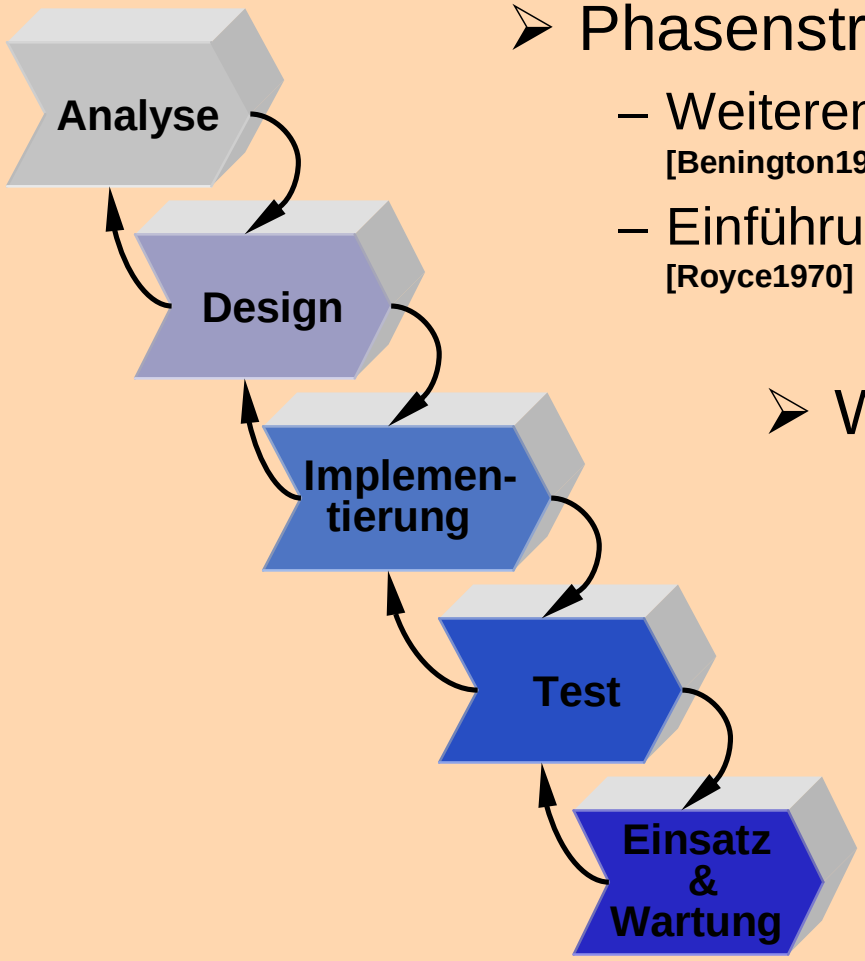


- Lehre der Konstruktion von Softwaresystemen
  - **Ingenieursdisziplin**
  - ***Prinzipien*, *Methoden* & *Werkzeuge* zur Lösung von Software-Problemen**
- SW-Probleme
  - **Komplexität**
  - **Immaterialität**
  - **Änderbarkeit**
  - **Nicht-Konformität**

Software-Entwicklungsprozesse

- Vorgehensweisen & Abläufe zur systematischen Softwarekonstruktion
- Unterteilt in Phasen & Aktivitäten
  - **Phasen:** Zeitabschnitte von Aktivitäten eines bestimmten Themas (Bsp. Analyse, Test, Wartung)
  - **Aktivität:** Zielgerichtete Handlung (Bsp. definieren, finden, testen)
- Klassifizierbar
  - **Phasenstrukturiert**
    - **Phasen linear an der Zeit orientiert**
    - **Betonung auf volländige Resultate an Phasenenden**
  - **Zyklenstrukturiert**
    - **Phasen zyklisch an der Zeit orientiert**
    - **Betonung auf unvollständigen Resultaten $\Rightarrow$ Iteration**

Das Wasserfall-Modell



- Phasenstrukturierter SE-Prozeß
  - Weiterentwicklung des **„stagewise model“** **[Benington1956]**
  - Einführung von **Rückkopplungschleifen** **[Royce1970]**

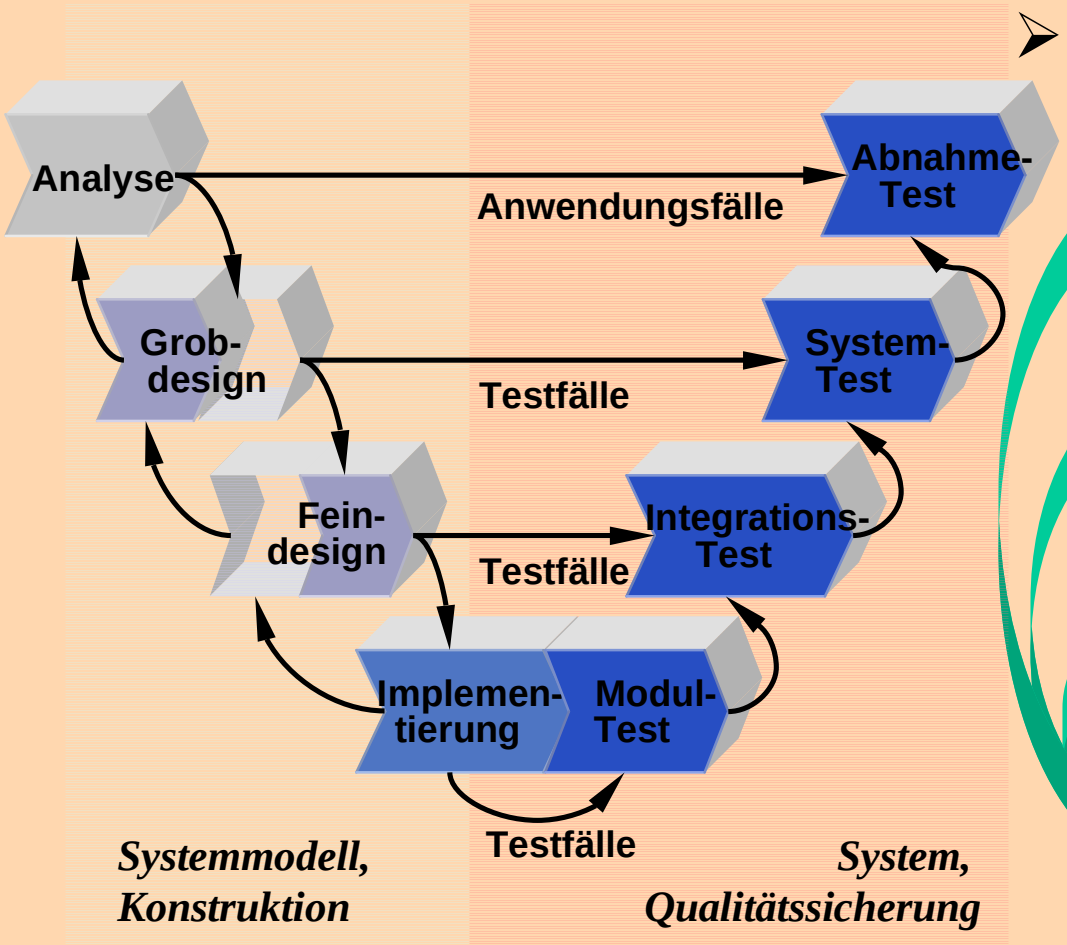
- Weitere Charakterisierung
  - **Sequentiell**
  - **Dokumentgetrieben**
  - **phasengesteuert**
  - **Phasen in vollständiger Breite**
  - **Top-Down-Strategien**

Das Wasserfall-Modell (II)

- Bewertung
  - ☺ Festdefinierte Phasen mit Dokumenten (Meilensteine)
  - ☺ Einfach & verständlich
  - ☺ Geringer Mangementaufwand (Konstruktionszentriert, seq. Deadlines)
  - ☹ Over-Engineering: Versuch, Phasen vollständig abzuschließen
  - ☹ Überorganisation: Dokumente wichtiger als Softwaresystem
  - ☹ Wenig Berücksichtigung von Risiken & Risikobeseitigungsstrategien
  - ☹ Wenig Berücksichtigung von Qualitätssicherung & SW-Management
  - ☹ Sehr hoher „Time-to-Market“-Faktor

Das V-Modell

[Bröhl1993]



- Phasenstrukturierter SE-Prozeß mit Instanzen

Systemerstellung (SE)

- S-Anforderungen definieren
- System entwickeln

Qualitätssicherung (QS)

- QS-Anforderungen vorgeben
- (Teil)Systeme prüfen

Konfig.-Management (KM)

- Systemarchitektur planen
- Produkte/Rechte verwalten

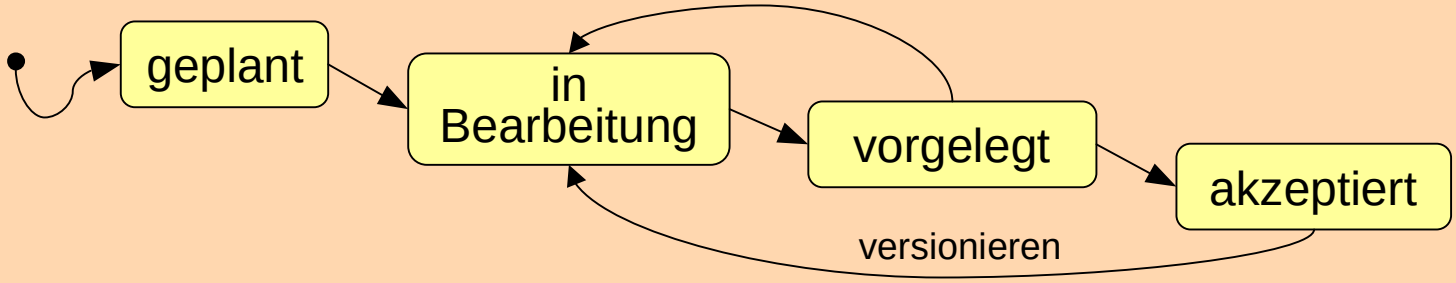
Projekt-Management (PM)

- Projekt planen, kontrollieren, überwachen

Das V-Modell (II)

- Feste Definition von Rollen & Aktivitäten
  - **Aktivität:** Handlungsablauf mit definiertem Resultat (Produkt)
  - **Rolle:** Handelnder mit bestimmten Aktivitäten
  - **AR-Matrix:** Zuordnung Aktivitäten ↔ Rolle
    - Zuordnung: **verantwortlich**, **beratend**, **mitwirkend**

- Feste Definition von Produkten
  - **Produkt:** Gegenstand oder Resultat einer Aktivität
  - **Produktzustände:**

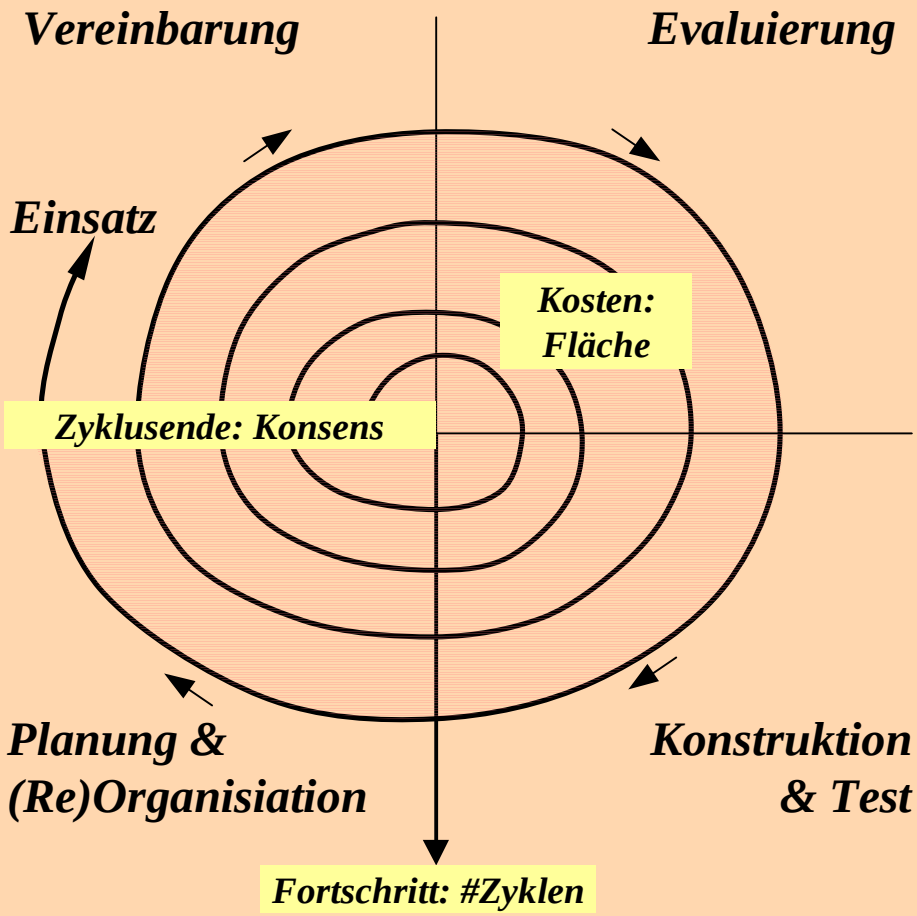


Das V-Modell (III)

- Weitere Charakterisierung
  - **an bestimmte IT-Problemdomänen anpaßbar („Tailoring“)**
  - **dokumentgetrieben**
  - **Bewertungsgesteuert (PM, QS)**
  - **Unterstützt eingebettete & Software-Systeme gleichermaßen**
- Bewertung
  - ☺ Berücksichtigt wesentliche Elemente industrieller SE (QS, SE, KM, PM)
  - ☺ Flexibilität: Anpaßbar an verschiedene IT-Problemdomänen
  - ☺ Gut geeignet für Großprojekte
  - ☹ Sehr hoher Mangementaufwand
  - ☹ Skalierbarkeit: Für kleine / mittlere Projekte ungeeignet
  - ☹ Hoher „Time-to-Market“-Faktor
  - ☹ Sehr komplex: Ohne CASE-Unterstützung nicht handhabbar

Das Spiral-Modell

[Boehm1988]



- Zyklenstrukturierter SE-Prozeß in 4 Schritten
  - **Vereinbarung**
    - Ziele, Alternativen, Strategien, Qualitäten
  - **Evaluierung**
    - Alternativen, „Make-or-Buy“, Risikominimierung
  - **Konstruktion & Test**
    - Beliebiger SE-Prozess für Teil-/Kernsystem !
  - **Planung**
    - Review, Planung neuer Zyklus

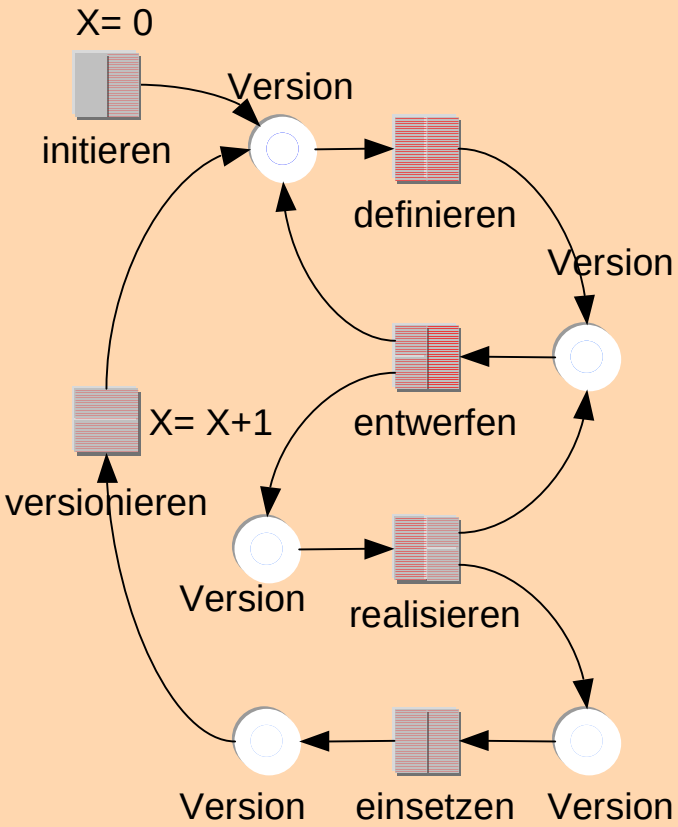
Das Spiral-Modell (II)

- Weitere Charakterisierung
  - **Risikogetrieben**
  - **Bewertungsgesteuert (Evaluierungsschritt, Review)**
  - **Eigentlich Meta-Entwicklungsprozess**

- Bewertung
  - ☺ Zyklische Bewertung & Planung in Abhängigkeit von Risiken
  - ☺ Flexibilität: Integration anderer Organisations- & SE-Prozesse
  - ☺ Frühzeitige Erkennung von Fehlern / ungeeigneten Alternativen
  - ☹ Hoher Mangementaufwand
  - ☹ Skalierbartkeit: Für kleine und mittlere Projekte ungeeignet
  - ☹ Wissen über Risikomangement bisher wenig verbreitet

Evolutionäre SW-Konstruktion

- Iterative, inkrementelle SE-Prozesse, die stabile Zwischenformen produzieren & weiterentwickeln



- Kernsystem (0-Version)
  - **voll einsatzfähiges System**
  - **Realsiert nur wichtige Kernanforderungen**
- Versionierung
  - **Einsatz liefert neue Erfahrungen & Anforderungen**
    ↳ neue Version

Evolutionäre SW-Konstruktion (II)

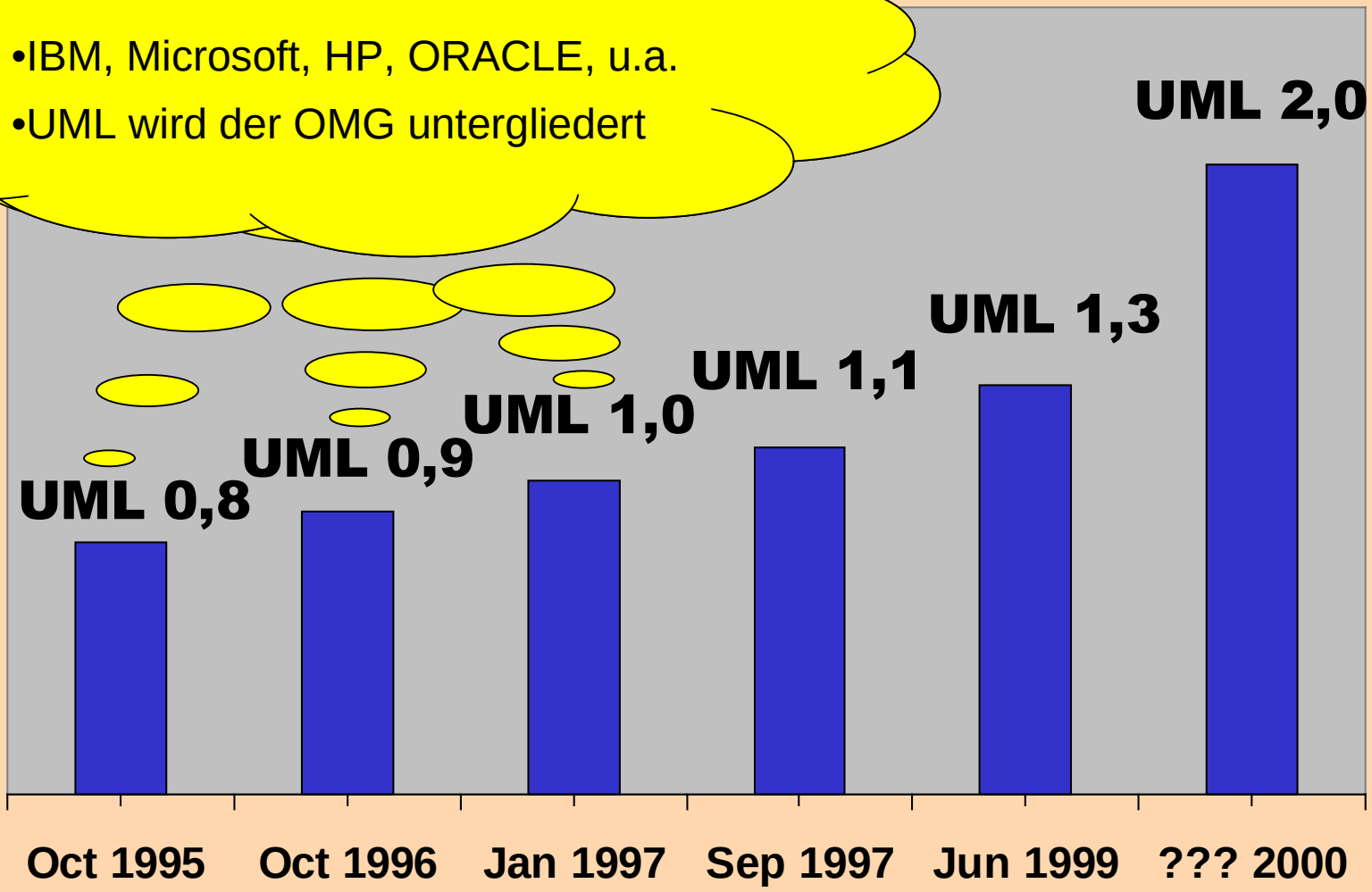
- Weitere Charakterisierung
  - **System/Code-getrieben (lauffähige Zwischenformen)**
  - **Erfahrungsgesteuert**
  - **Wartung ⇔ Weiterentwicklung**
  - **Software ≈ Prozeß mit Resultaten (statt Einzelsystem)**
- Bewertung
  - ☺ Relativ geringer „Time-To-Market“-Faktor
  - ☺ Versionen gut in Arbeitsabläufe integrierbar (gut studier- & planbar)
  - ☹ Gefahr teuerer Änderungswellen bei übersehenen Kernanforderungen
  - ☹ Gefahr von 0-Versionen, die bestimmte Evolutionspfade nicht tragen
- Gegenmaßnahmen
  - Größere Sensibilität bei Systemerweiterungen (Änderungen einplanen)
  - Strategien des stufenweisen Systemausbaus anwenden

Die Unified-Modeling-Language

- Objektorientierte Modellierungssprache
  - **visuell**
  - **Programmiersprachen-neutral**
  - **Problembereichs-neutral**
  - **Methoden- & Prozeß-neutral**
- **Ziel:** Industrieller Sprachstandard für objektorientierte Softwareentwicklungsmethoden
- **Strategie:** Fusion der 3 bek. Modellierungsnotationen
  - **OMT (Rumbaugh et al.)**
  - **BOOCH (Booch)**
  - **OBJECTORY (Jacobson et al.)**

Geschichtliches zur UML

- IBM, Microsoft, HP, ORACLE, u.a.
- UML wird der OMG untergliedert



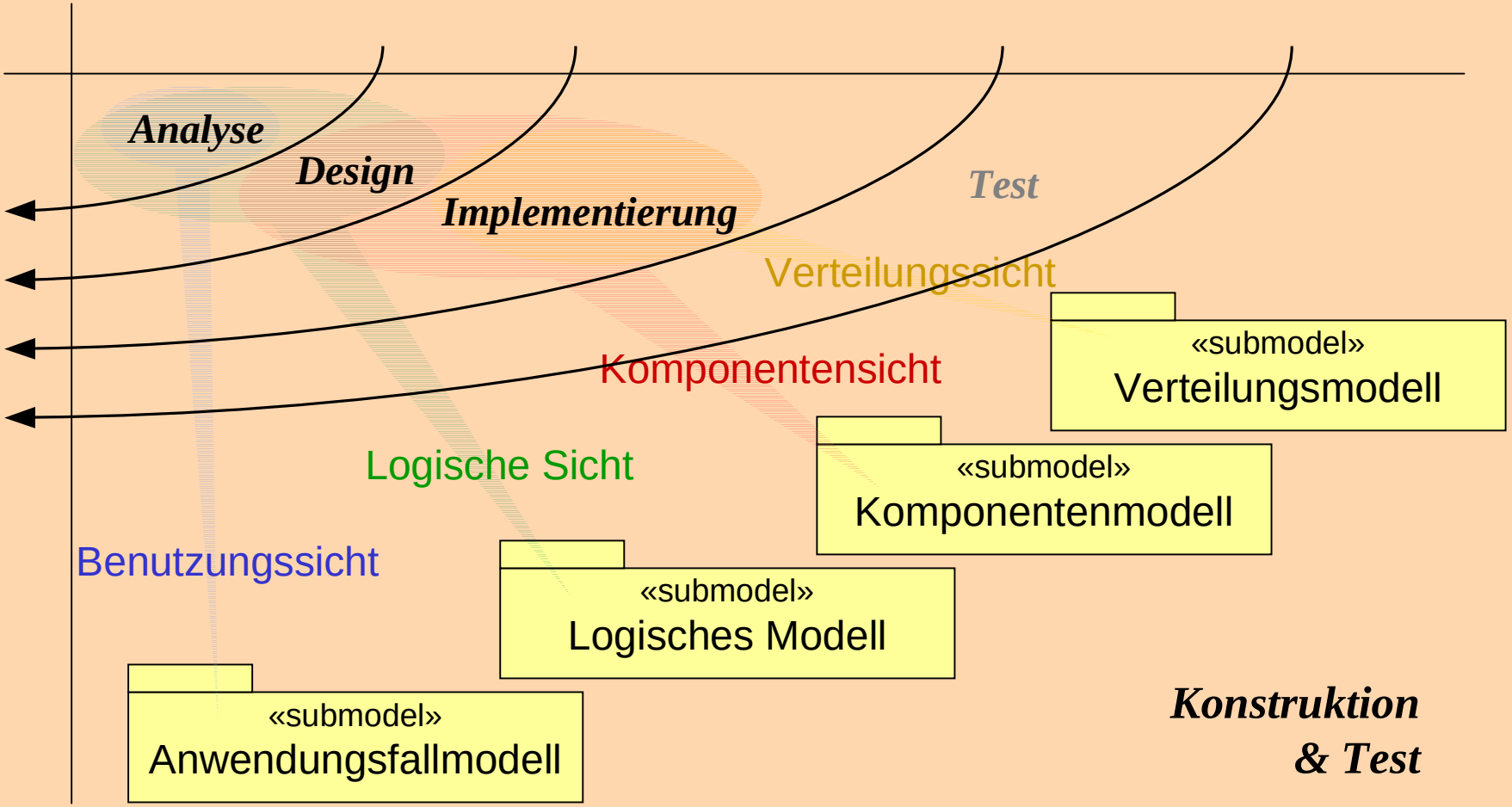
Grundkonzepte der UML

- Semantische Netzwerke
  - **Struktur- & Verhalteneigenschaften größtenteils graphisch**
  - **Graphen: Semantische Knoten & Kanten**
- Modulares Sprachdesign
  - **Syntax & statische Semantik im Meta-Modell organisiert**
  - **Sprachkern als „Back Bone“ anderer Sprachmodule**
- Erweiterbarkeit
  - **„Profiles“ zur Anpassung an Modellierungsdomänen**
  - **Bsp.:** UML/RT, UML for Business Modeling, *OCoN*/UML

Systemsichten & Teilmodelle der UML

- Systemsicht
  - **Partielles Projektions- oder Abstraktionsmittel**
  - **Erfassung von Systemeigenschaften unter vorgeg. Kriterien**
- Systemmodell
  - **Organisationsmittel**
  - **Ergebnis einer Systemabstraktion/-projektion**
- 4 Systemsichten
  - **Benutzungssicht**
  - **Logische Sicht**
  - **Komponentensicht**
  - **Verteilungssicht**
- 4 Teilmodelle
  - **Anwendungsfallmodell**
  - **Logisches Modell**
  - **Komponentenmodell**
  - **Verteilungsmodell**

Sichten & Teilmodelle der UML innerhalb eines SE-Prozeßes



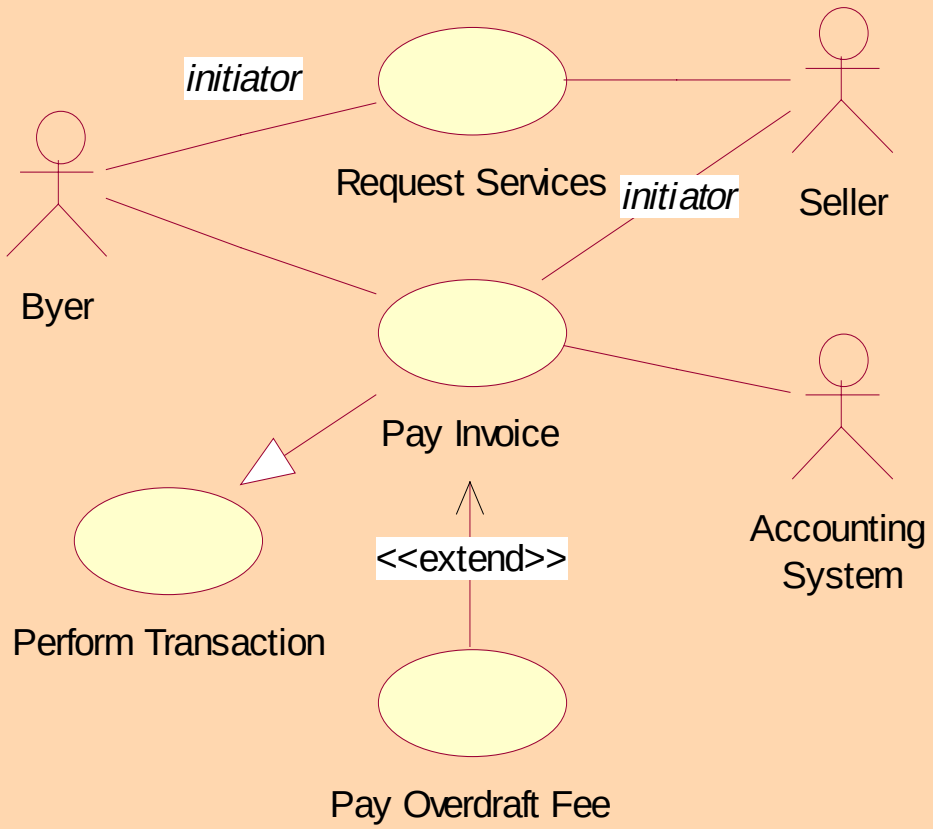
Diagrammformen der UML

- Diagrammformen repräsentieren
  - **(Teil)sprachen unter den 4 Systemsichten**
  - **Organisationsmittel in UML-Teilmodellen**

- Strukturdiagramme
  - **Typstrukturen**
    - *Systemdiagramme*
    - Klassendiagramme
    - Komponentendiagramme
  - **Instanziierungen**
    - *Objektdiagramme*
    - Verteilungsdiagramme

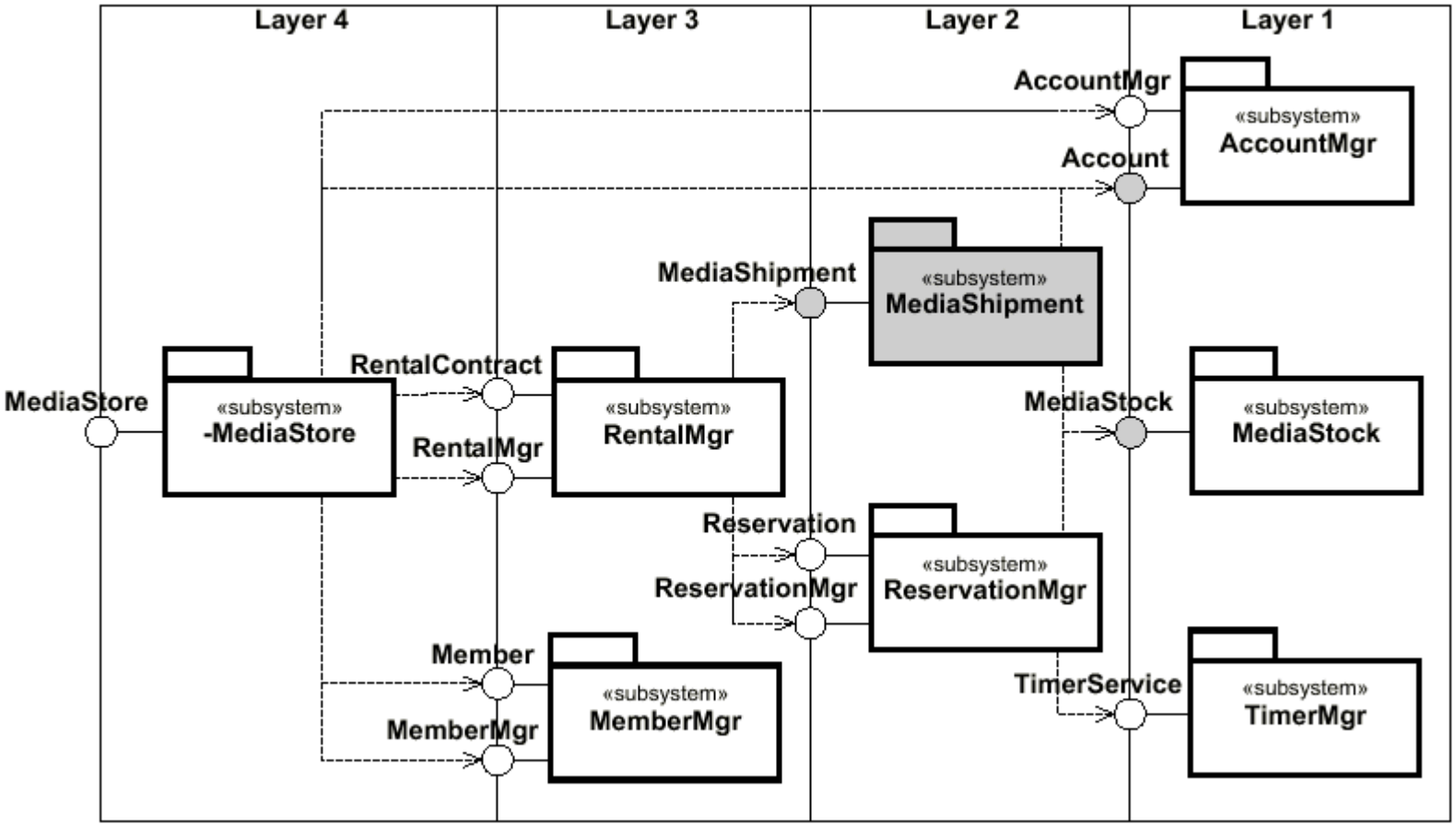
- Verhaltensdiagramme
    - Anwendungsfalldiagramme
  - **Kooperationen**
    - Kollaborationsdiagramme
    - Sequenzdiagramme
  - **Reaktionen**
    - Zustandsdiagramme
    - Aktivitätsdiagramme

Anwendungsfalldiagramm

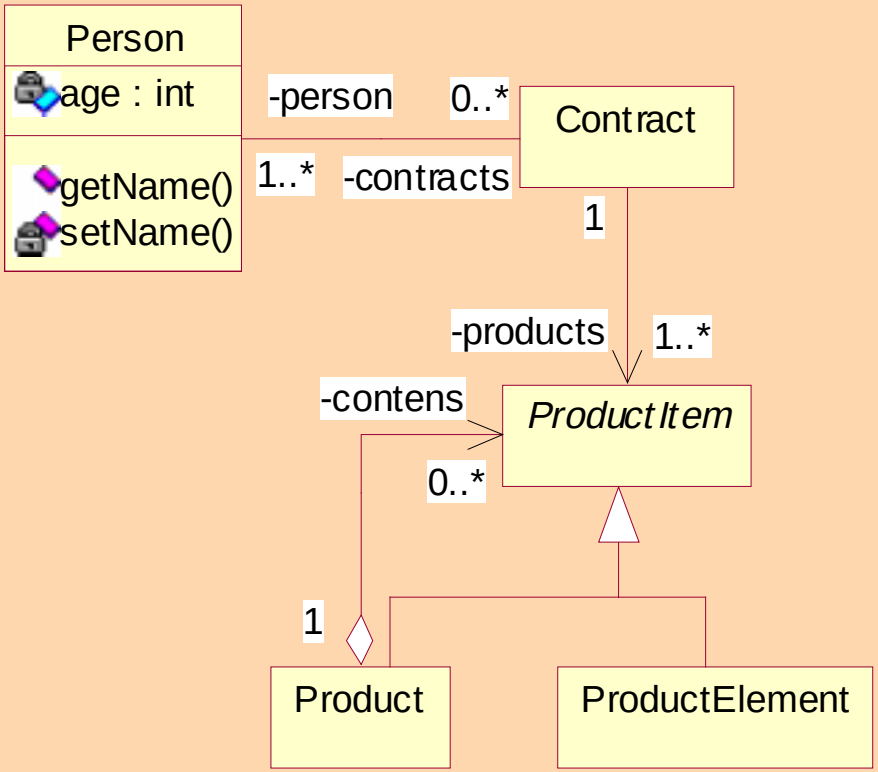


- **Beschreibt Benutzungsfälle & Benutzerrollen**
- **Verwendungszweck**
  - Geschäftsprozesse
  - Systemanforderungen
- **Charakteristika**
  - Akteure & Anwendungsfälle
  - Anwendungsfall-Taxonomien
  - Anwendungsfall-Erweiterungen
  - Kommunikationskanäle

Systemdiagramm

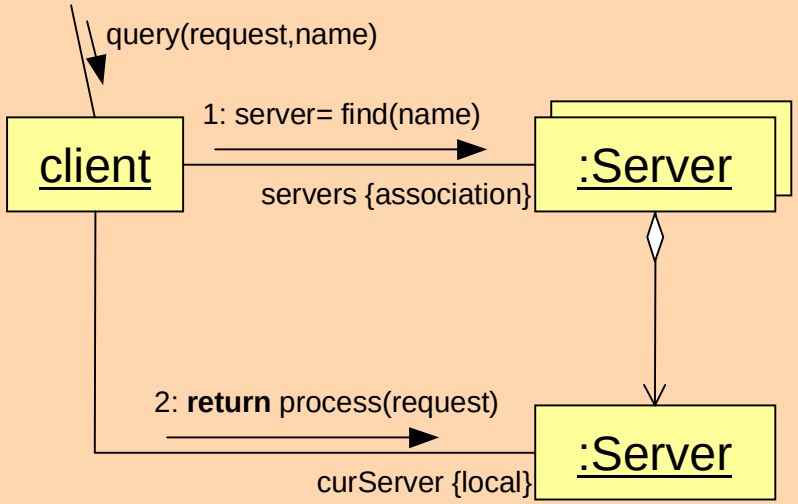


Klassendiagramm



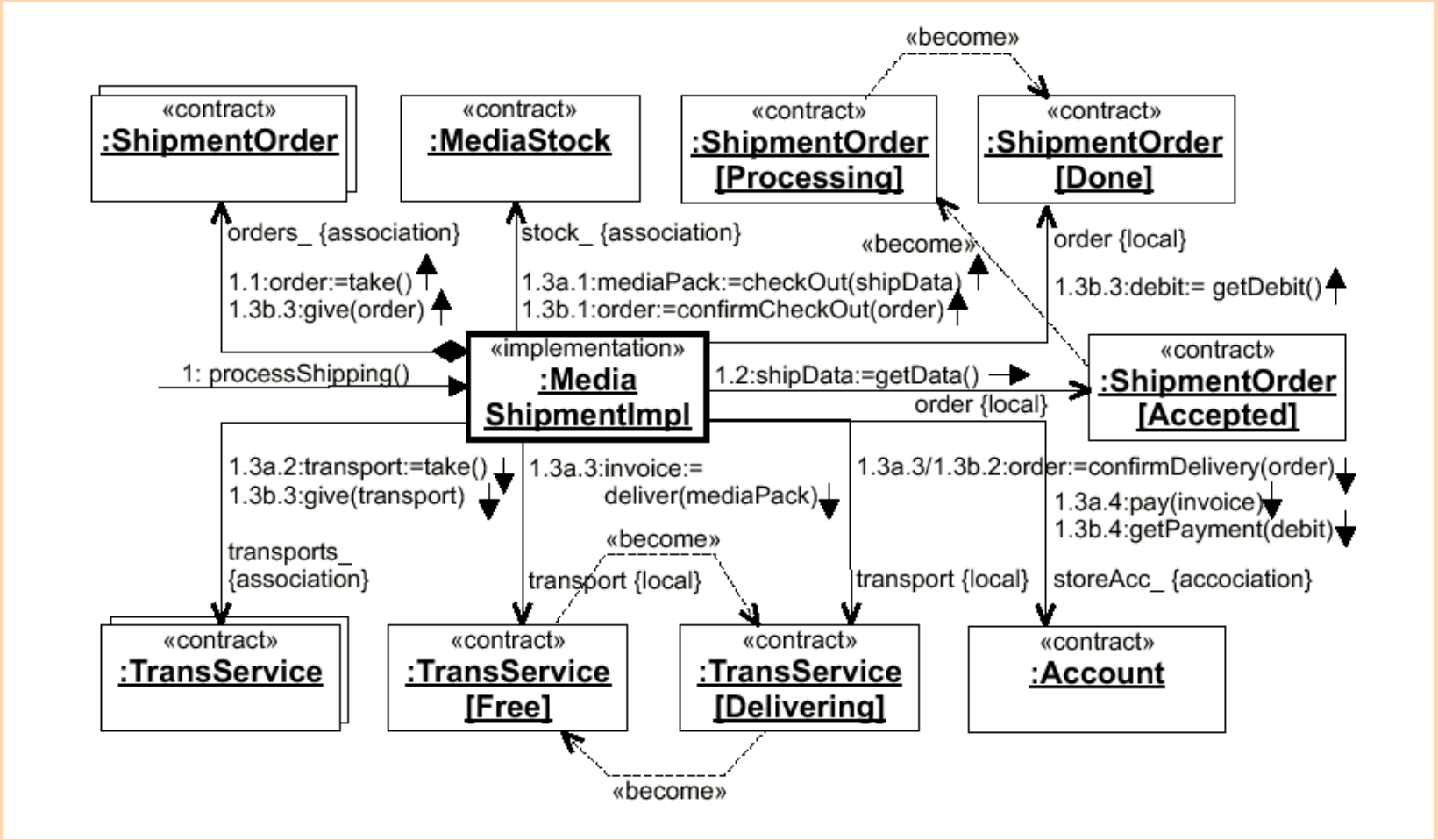
- **Beschreibt statische Feinstrukturen**
- **Verwendungszweck**
  - Fachkonzept-Klassen
  - System-Klassen
- **Charakteristika**
  - Klassen & Laufzeitbeziehungen
  - Klassentaxonomie
  - Strukturintegrität zur Laufzeit

Kollaborationsdiagramm

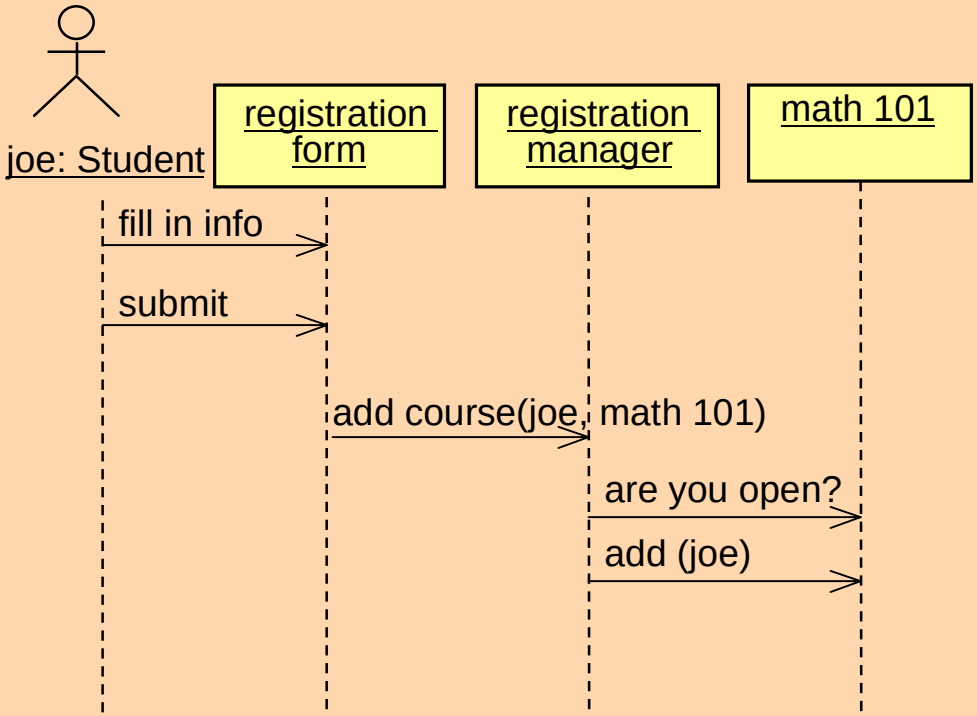


- **Beschreibt ein struktur-orientiertes Interaktionsverhalten**
- **Verwendungszweck**
  - Geschäftsprozessabläufe
  - Anwendungsfallabläufe
  - Operationsabläufe
  - Signalübermittlungsabläufe
- **Charakteristika**
  - Objektrollen
  - Objektverbindungen
  - Nachrichtenübermittlungen
  - Sequenznummerierung

Kollaborationsdiagramm (II)

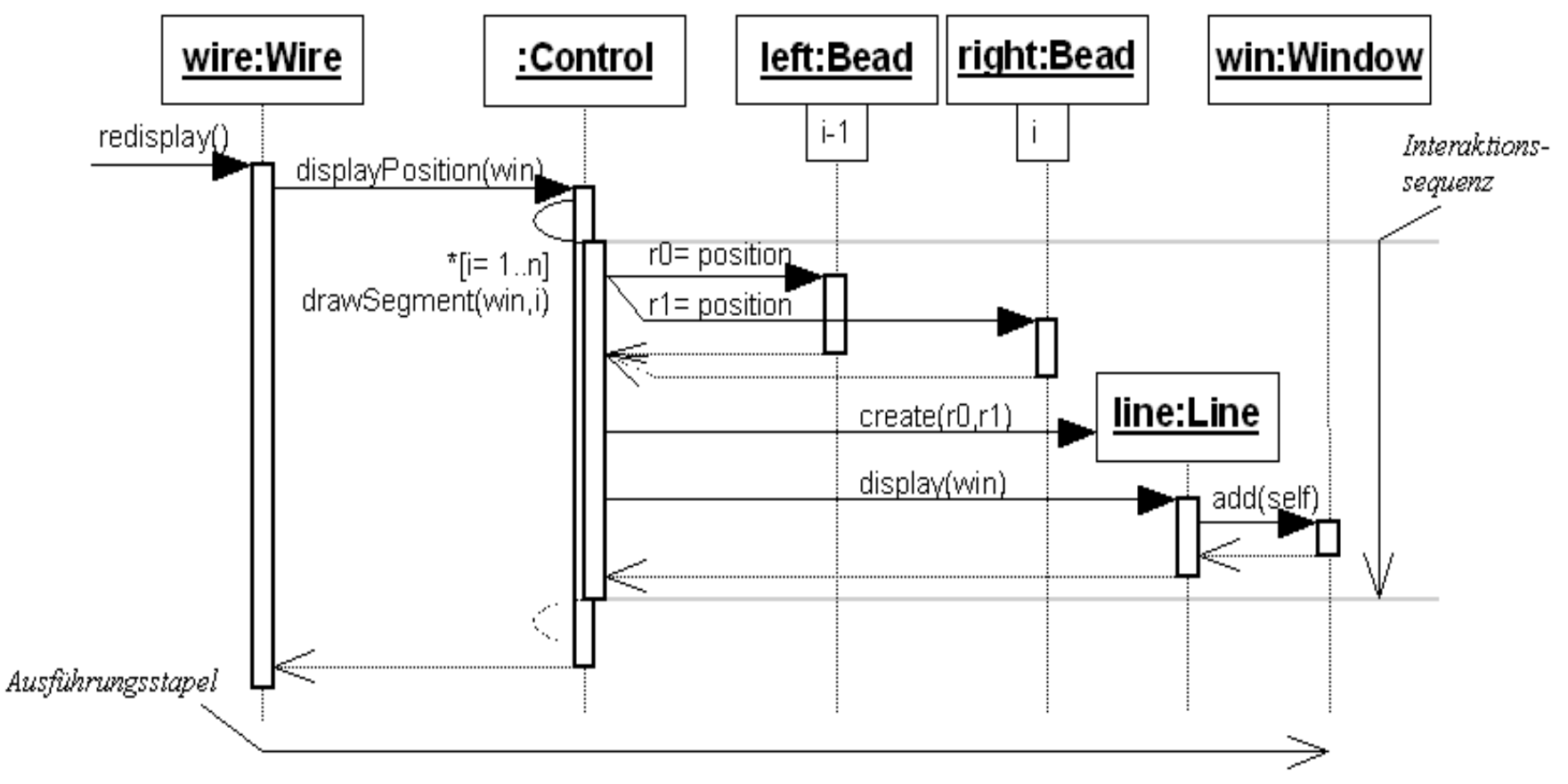


Sequenzdiagramm

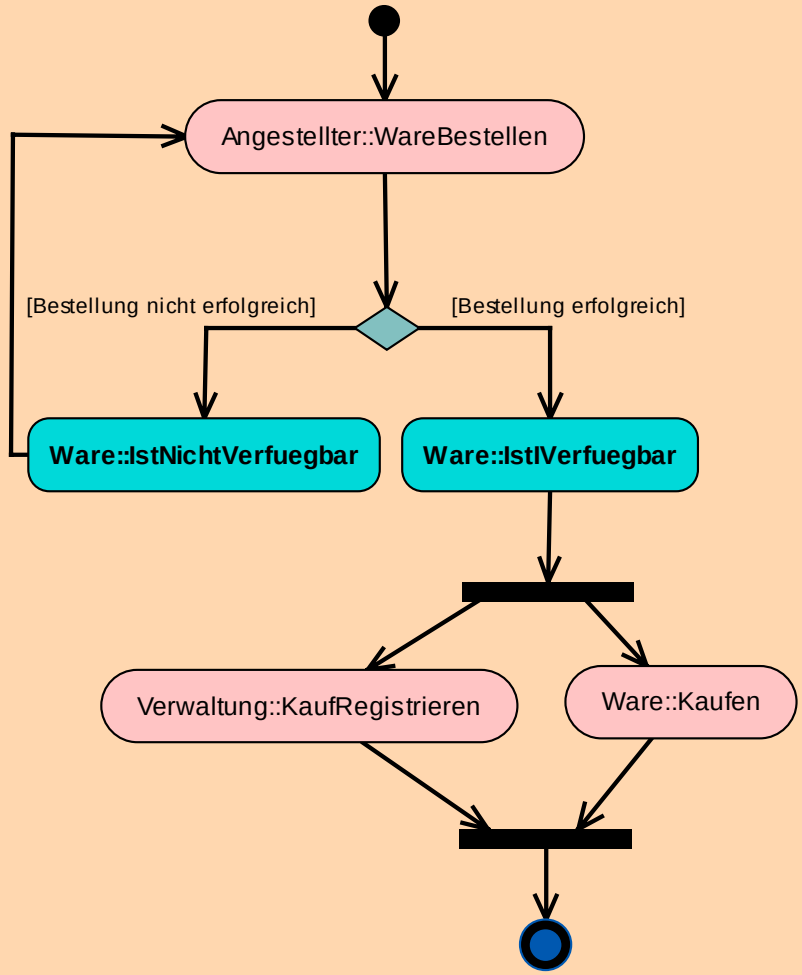


- **Beschreibt ein zeitorientiert. Interaktionsverhalten**
- **Verwendungszweck**
  - Geschäftsprozeßabläufe
  - Anwendungsfallabläufe
  - Operationsabläufe
  - Signalübermittlungsabläufe
- **Charakteristika**
  - Objektrollen
  - Zeitachsen
  - Nachrichtenübermittlungen

Sequenzdiagramm (II)

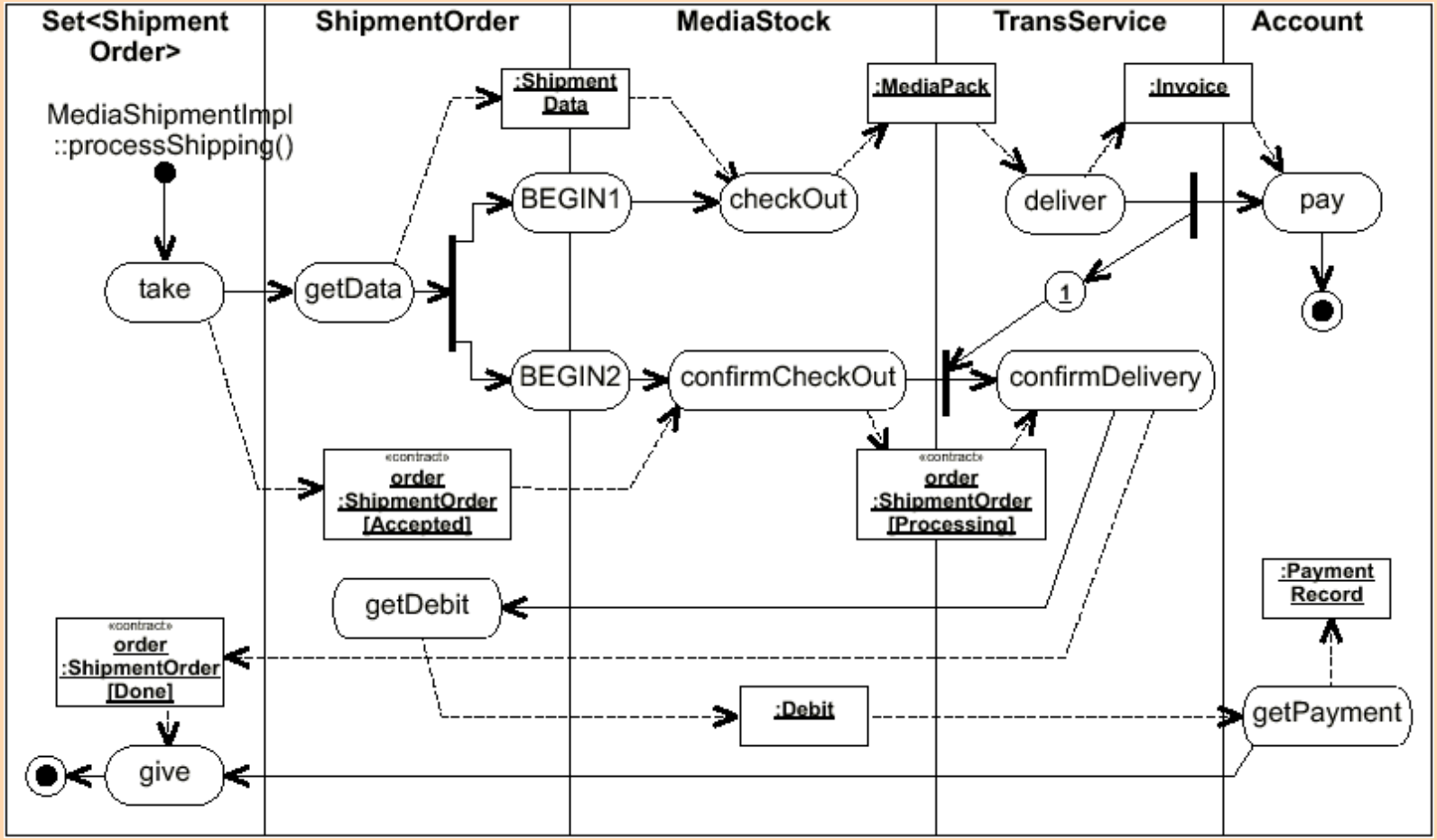


Aktivtätsdiagramm

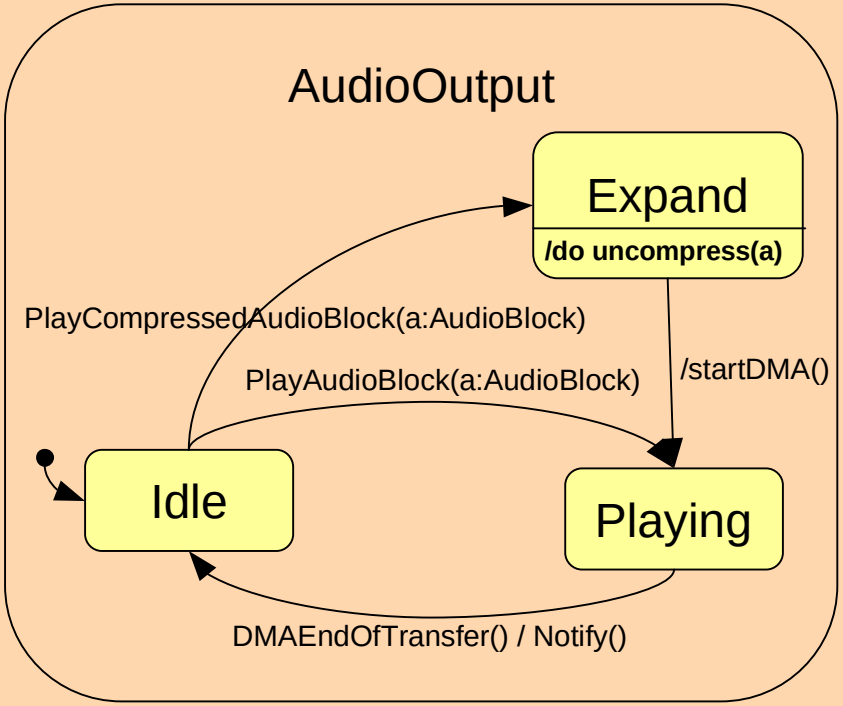


- **Beschreibt alle Aktionszustände einer Menge von Abläufen**
- **Verwendungszweck**
  - Anwendungsfallabläufe
  - Geschäftsprozeßabläufe
  - Methodenabläufe
- **Charakteristika**
  - Kausalordnung
  - 1 Startzustand, 1..* Endzustände
  - **Zustände:** Bedingungen, Aktionen, Aktivtäten
  - **Transitionen:** Ablauffortschritt (**keine** Ereignisreaktionen!)

Aktivtätsdiagramm (II)

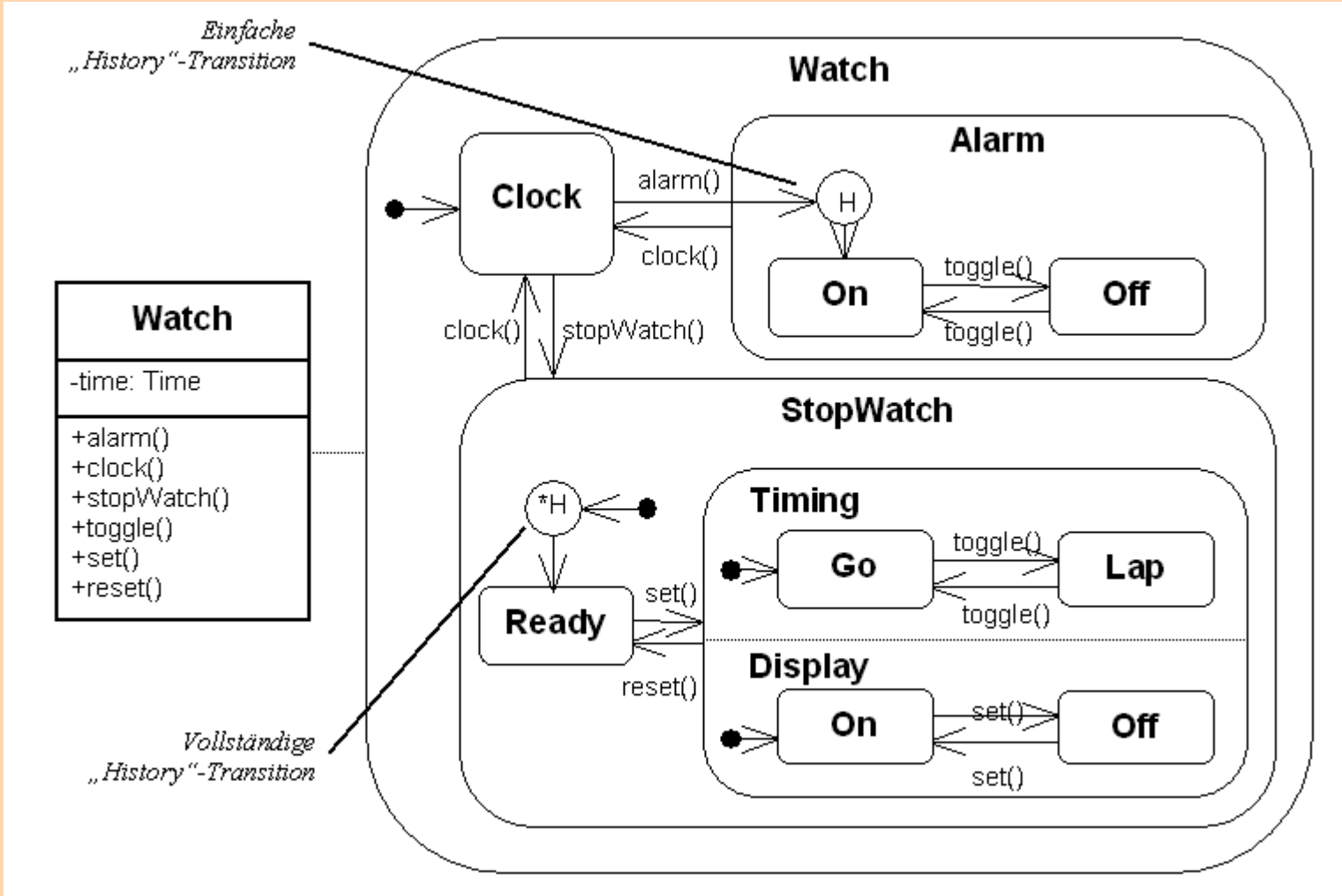


Zustandsdiagramm

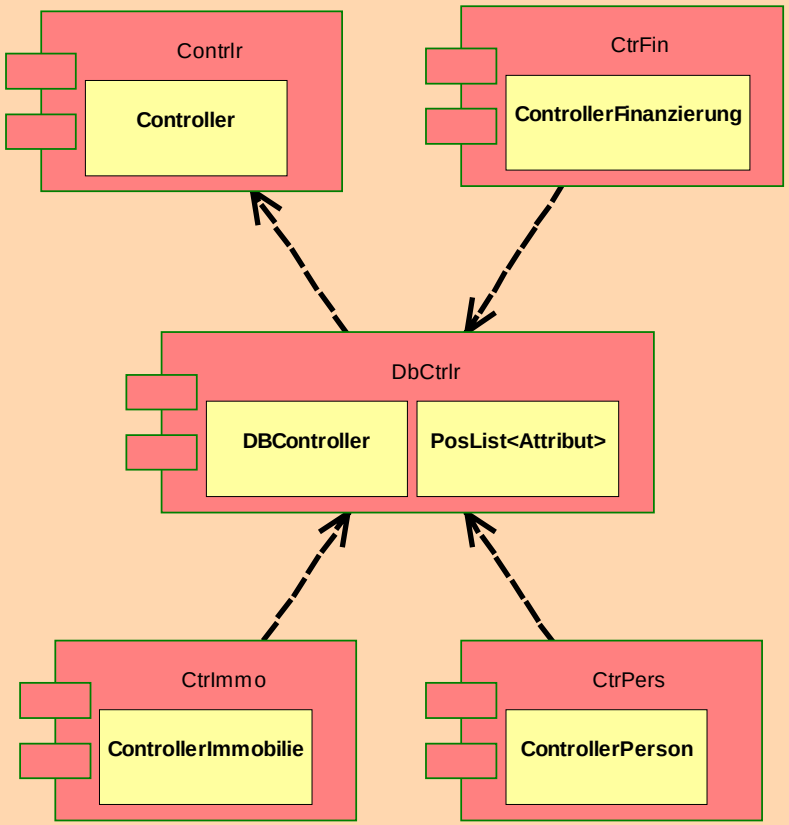


- **Beschreibt alle Reaktionszustände eines Klassifizierers**
- **Verwendungszweck**
  - (Eigen-)Kontrollverhalten
  - Protokollverhalten
- **Charakteristika**
  - 1 Startzustand
  - **Zustände:** Bedingungen,Aktivitäten
  - **Transitionen:** Ereignisreaktionen

Zustandsdiagramm (II)

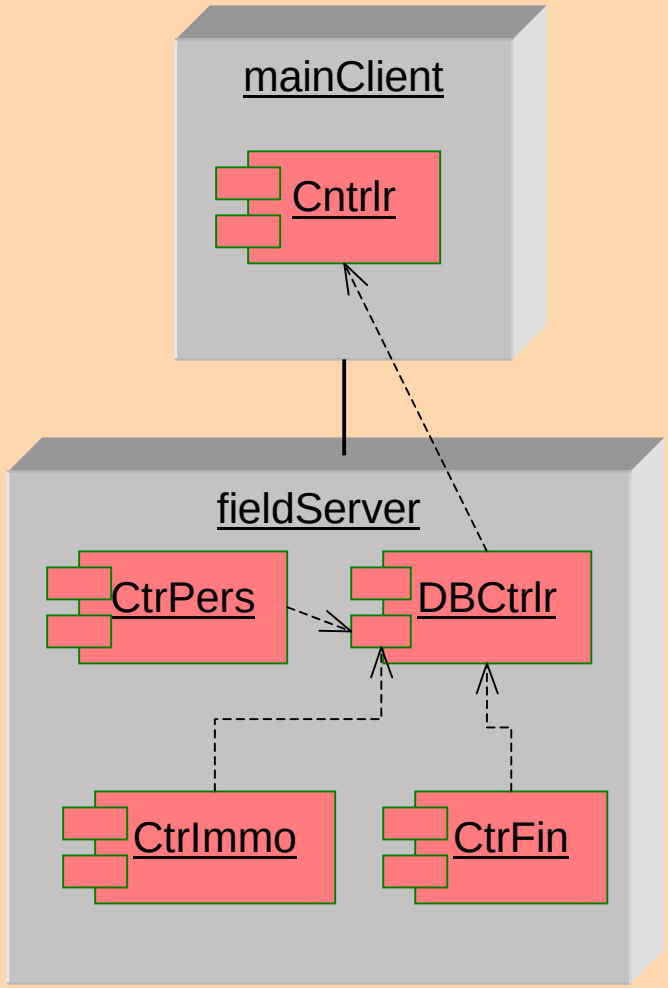


Komponentendiagramm



- **Beschreibt die Realisierungs-architektur**
- **Verwendungszweck**
  - Realisierungsarchitektur
  - Code-Organisation
- **Charakteristika**
  - Komponenten, Schnittstellen
  - Benutzungsbeziehungen
  - Elemente

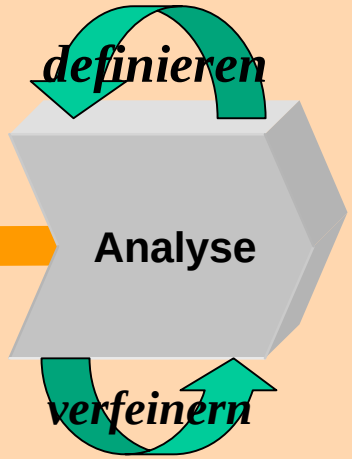
Verteilungsdiagramm



- **Beschreibt die Laufzeit-umgebung**
- **Verwendungszweck**
  - Prozess-/Prozessor-Organisation
- **Charakteristika**
  - Knoten
  - Kommunikationsverbindungen
  - Komponenten & Laufzeitobjekte

Analyse

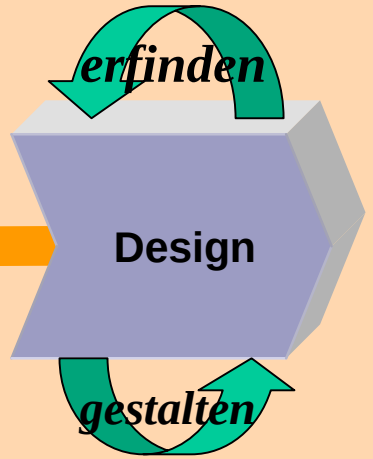
[Oesterreich1999, S.77-114]



- Erfassen **relevanter** Systemeigenschaften
  - **Wer** soll mit dem System interagieren ?
  - **Was** soll das System funktional leisten ?
  - **Wie** soll mit dem System interagiert werden ?
- Aktivitäten
  - **Systemkontext, Systemgrenzen & Schnittstellen finden**
  - **Akteure & Anwendungsfälle finden**
  - **Fachkonzept-Klassen & Beziehungen finden**
  - **Review: Kernanforderungen (sinnvoll, wiedersprüchl., benutzbar)**
- Ziele
  - **Produktdefinition**
  - Problemdomäne begreifen
  - Erste Systemvorstellung
- UML-Diagramme
  - **Anwendungsfalldiagramm**
  - **Aktivitätsdiagramm**
  - **Klassendiagramm**

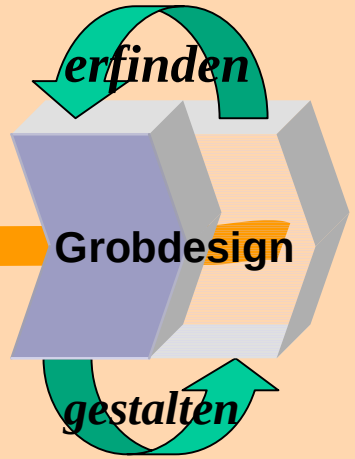
Design

[Oesterreich1999, S.115-140]



- Entwerfen eines sich verhaltenden Systems anhand analysierter Anforderungen
  - **Wie** sollen die Systemfunktionen ablaufen ?
  - **Wer** bilden die Verantwortlichkeiten im System ?
  - **Wann & Wie** sollen Verantwortlichkeiten interagieren ?
- Untergliederung: **Grobdesign & Feindesign**
- Ziele
  - **Systementwurf**
  - Implementierungsrahmen

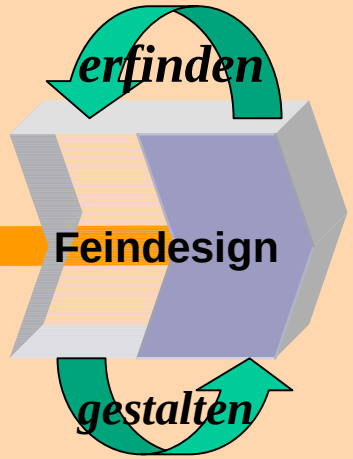
Grobdesign



- Entwerfen einer **Gesamtsystemarchitektur**

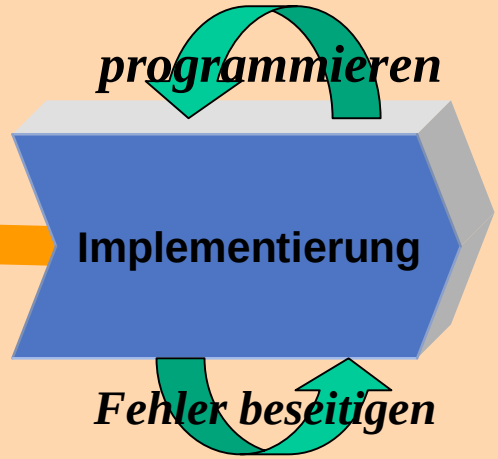
- Aktivitäten
  - **Bekannte GSA aus der Problemdomäne evaluieren**
  - **Teilsysteme & ihre Beziehungen entwerfen**
  - **Schnittstellen entwerfen**
  - **Review: Systemarchitektur (Effz, Wartb, Rob)**
- Ziele
  - Verteilung der Anforderungen
  - Zuordnung von Verantworlichkeiten
  - **Produktgrobstruktur**
  - Erste Realisierungsvorstellungen
- UML-Diagramme
  - Systemdiagramm
  - Zustandsdiagramm
  - Aktivitätsdiagramm
  - Kollaborationsdiagramm
  - Sequenzdiagramm

Feindesign



- Entwerfen von Klassen **& deren Verhalten**
- Aktivitäten
  - **Fachklassen & ihre Beziehungen einbetten**
  - **Sys-Klassen entwerfen (Kontroll-, Schnittstellen,- Datenklassen)**
  - **Objektinteraktionen entwerfen (Koordination, Operation)**
  - **Review: Feinstrukturen & ihr Verhalten (Effz, Wartb, Rob)**
- Ziele
  - Umsetzung einzelner Anforderungen
  - **Systemfeinstruktur**
  - Konktrete Realisierungsvorstellungen
- UML-Diagramme
  - Klassendiagramm
  - Zustandsdiagramm
  - Aktivitätsdiagramm
  - Kollaborationsdiagramm
  - Sequenzdiagramm

Implementierung



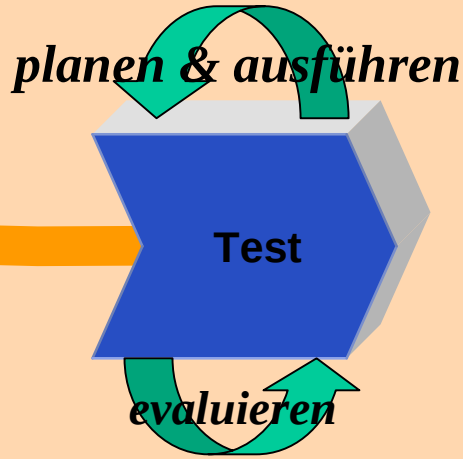
- Progammieren aller entworfenen Grob- & und Feinstrukturen & ihrem Verhalten

- Aktivitäten
  - **„Progammieren im Kleinen“**
  - **Teilsysteme, Klassen & ihre Beziehungen ausprogrammmieren**
  - **Operationen ausprogrammieren („Schrittweise Verfeinerung“)**
  - **Review: Komponenten, Schnittstellen, Klassen & Operationen**

- Ziele
  - Realisierung aller Anforderungen
  - **System & Systemdokumentation**
  - Bestimmung von Testfällen & Testdaten

- UML-Diagramme
  - Komponentendiagramm
  - Verteilungsdiagramm

Test



- Validieren von Anforderungs-, Grob-, Fein- & Realisierungsstrukturen & ihrem Verhalten

- Aktivitäten: (Automatisierten Tests, Reviews, Walktroughs)
  - **Planung & Entwurf: Prüflinge, Teststrategien, Maße, Ablaufplan**
  - **Definition von Testfällen / Testdaten: Testklassen, Randfälle**
  - **Durchführung von Tests**
  - **Evaluation der Testergebnisse: Korrekturplanung**

- Ziele
  - Qualitätssicherung (Ben, Korr, Eff, Zuv, Wartb)
  - Kostenminimierung

Einsatz & Wartung

- Systemabnahme & Weiterentwicklung mit Beseitigung von Betriebsfehlern

- Aktivitäten
  - **Übergabe: Belastungs- & Streßtests**
  - **Installation, Schulung, Inbetriebnahme**
  - **Fehlerkorrektur, Optimierung, Anpassungen**
  - **Review: Änderungen & Weiterentwicklung**

- Ziele
  - Einführung & Betrieb
  - Sammeln weiterer Anforderungen

Zusammenfassung

- Softwaretechnik & SE-Prozesse essentielle Mittel zur Konstruktion von Softwaresystemen
- Analyse, Design, Implementierung, Einsatz & Wartung invariante Phasen aller modernen SE-Prozesse
- UML = Modellierungssprachen-Standard zur objektorientierten SW-Konstruktion
- UML & Teilmodelle unterstützt in SE-Prozessen nur die Konstruktionsphasen
- UML besitzt visuelle Teilsprachen zur Anforderungs-, Struktur- & Dynamikmodellierung

Literatur

[Balzert1998a] Balzert, H. - *Lehrbuch der Software-Technik: Software-Entwicklung*. Spektrum Akademischer Verlag, Heidelberg - Berlin, 1998

[Balzert1998b] Balzert, H. - *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Spektrum Akademischer Verlag, Heidelberg - Berlin, 1998

[Benington1956] Benington, H.D. - *Production of Large Computer Programs.* In: Proc. ONR Symposium an Advanced Programming Methods for Digital Computers, Jun 1956

[Boehm1988] Boehm, W.B. - *Spiral Model of Software Development and Enhancement.* In: IEEE Computer, Nr.5 (21), p.61-21, 1988

[Bröhl1993] Bröhl, A.-P. - *Das V-Modell.* Oldenburg Verlag, 1993

[Oesterreich1997] Oesterreich, B. - *Objektorientierte Softwareentwicklung mit der Unified Modeling Language*. Oldenburg Verlag, 1997

[Royce1970] Royce, W.W. - *Managing the Development of Large Software Systems.* In: IEEE WESCON, p.1-9, Aug 1970