Proseminar

(Software-Spezifikation mit UML 2)

Systemspezifikation mit SysML
im SoSe 2010

Einführung

Prof. Dr. Joachim Fischer
Dipl.-Inf. Andreas Blunk

fischer@informatik.hu-berlin.de

Einführung

1. Organisatorisches
2. Trends in der Software-Entwicklung
3. Modelle in UML
4. Grundsätzliches zu Systemen und Modellen
5. Modelle in SysML

- Homepage
  http://www.informatik.hu-berlin.de/sam

- Seminar
  Di: 13.15 Uhr RUD 25 III.113

  - Teilnahme, 2 Vorträge, Diskussion
  - Bewertung der Präsentation, Mitarbeit im Seminar
  - Praktischer Teil: UML-Modellierungsprojekt

- Seminarschein

Bestimmung der Teilnehmer

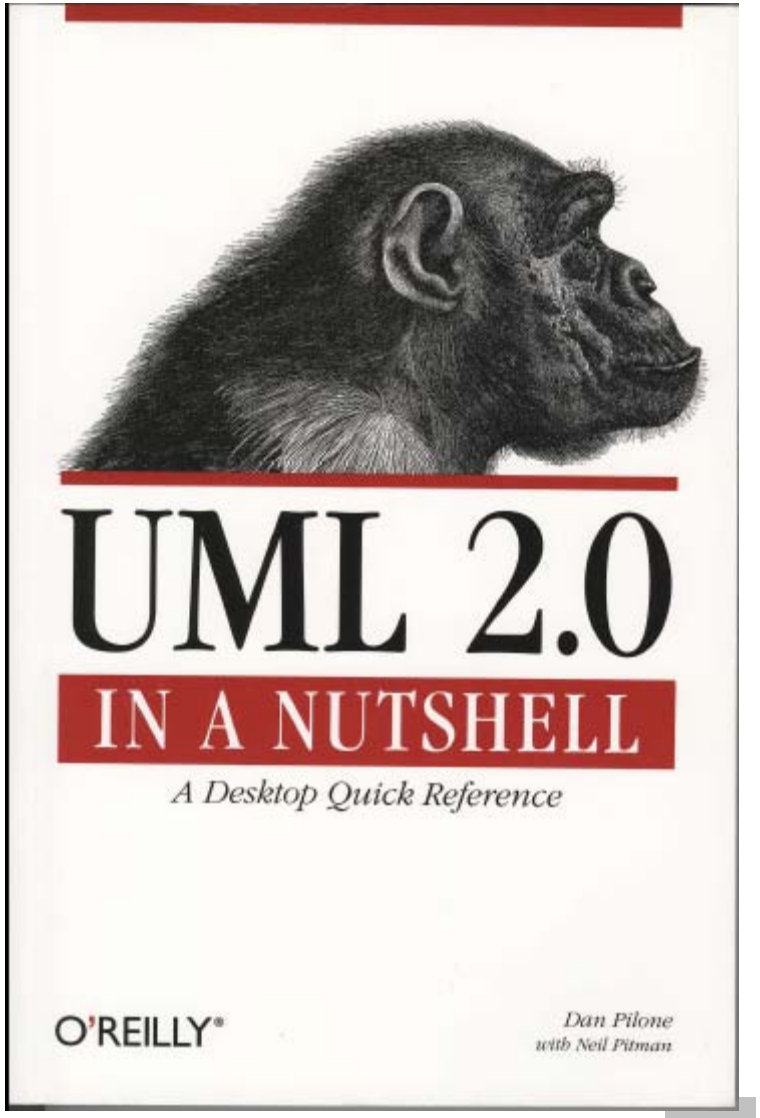
Pro-Seminar Systemmodellierung mit SysML

Goya-Meldungen

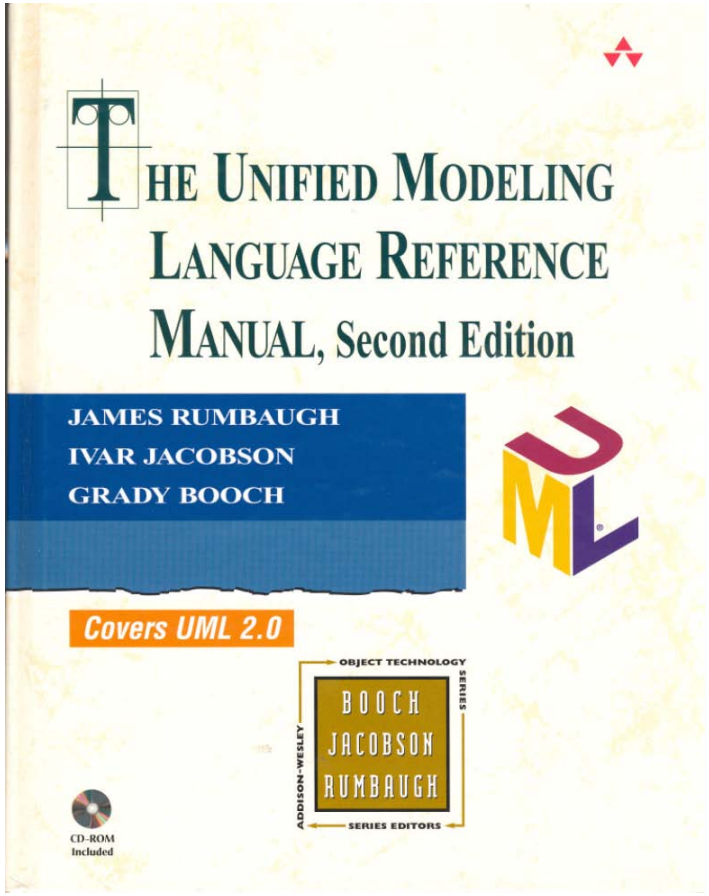
Nr.	Nachname	Vorname	Anwesenheit
1	Stadie	Oliver	
2	Grzebin	Felix	
3	Obst	Benjamin	
4	Useinov	Vitalij	
5	Manthey	Michel	
6	Krabi	Marianne	
7	Müller	Jens	
8	Iks	Taras	
9	Fobian	Martin	
10	Wächter	Georg	
11	Hönicke	Florian	
12	Grassmann	Tom	
13	Lunow	Daniel	

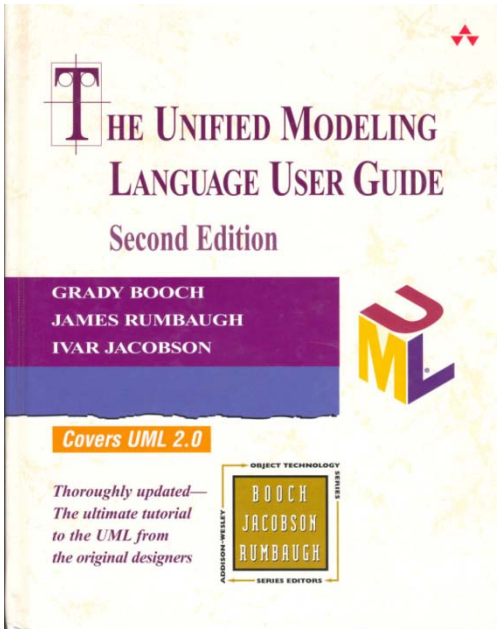
Unsere UML-Quellen: Bücher

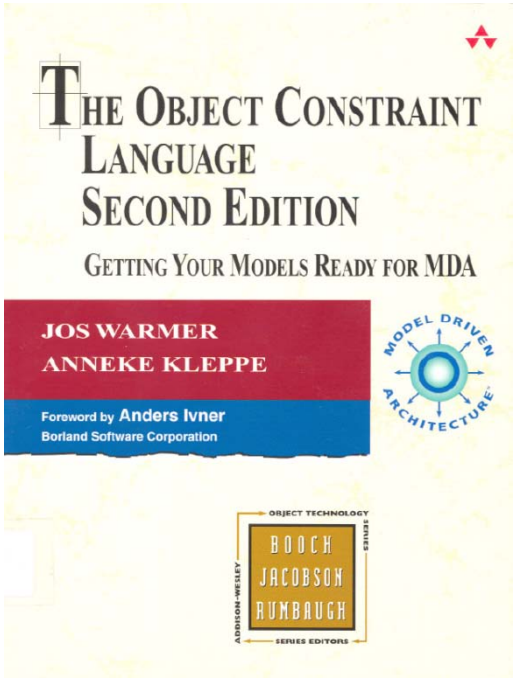
Weitere Empfehlungen



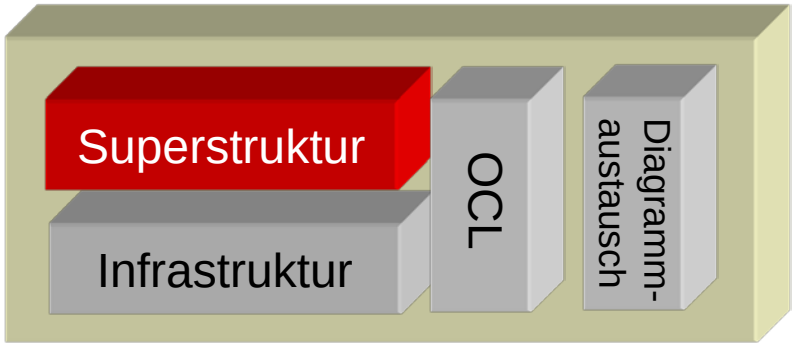
Die Originale !!!

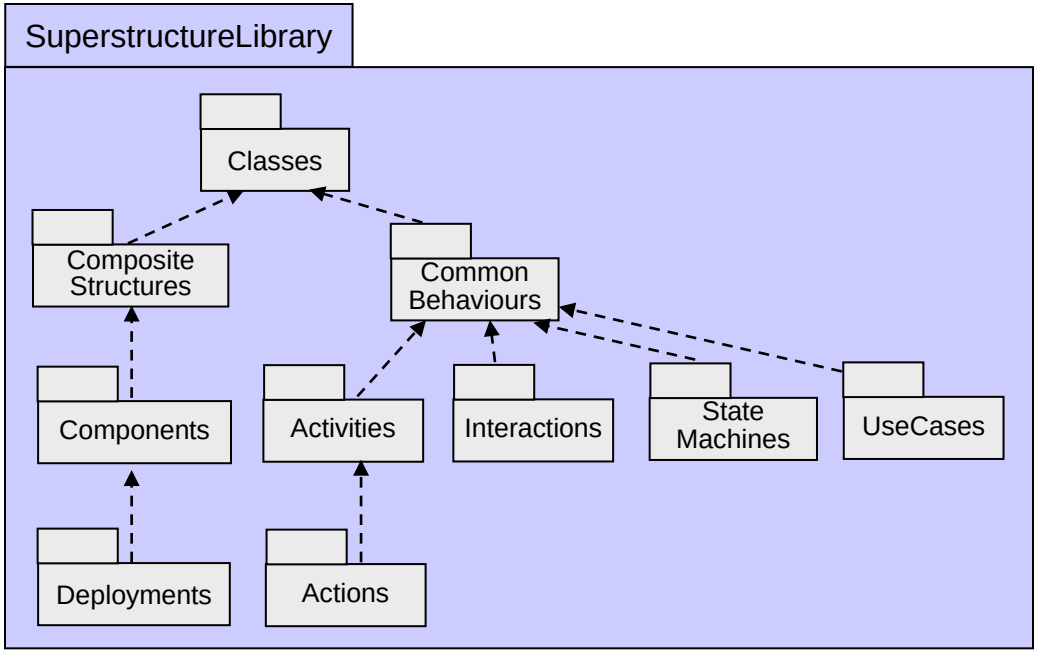






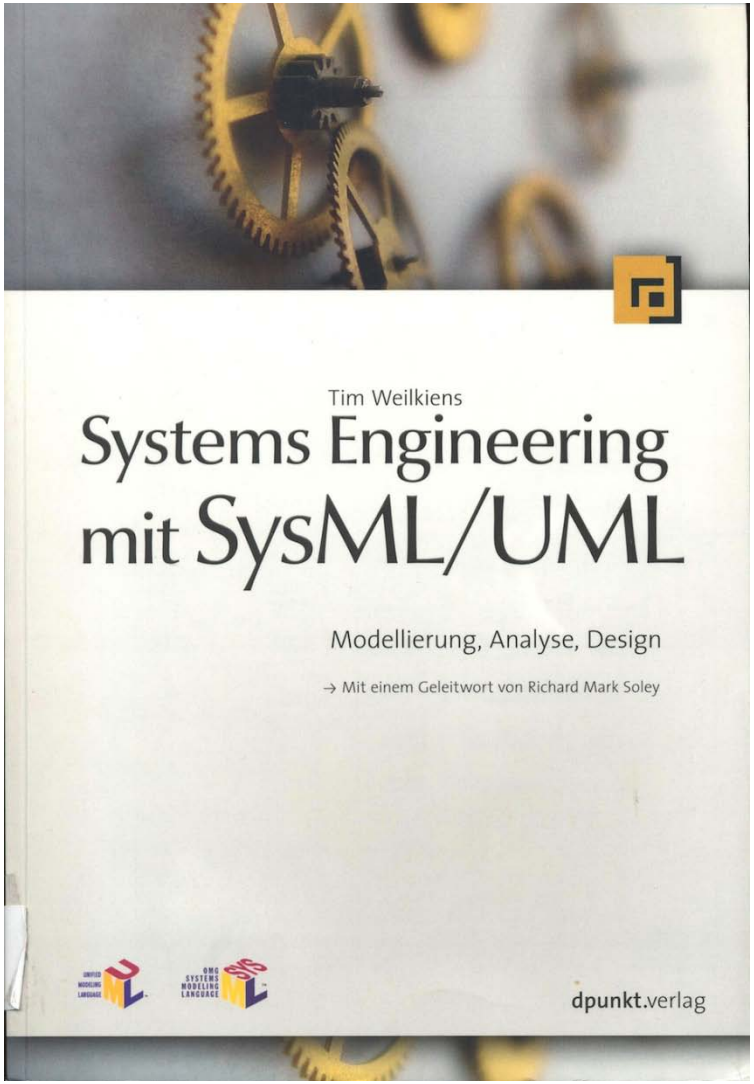
Unsere UML-Quellen: Der OMG-Standard





- Modellierungs-konzepte
- Sprachdefinitions-konzepte
- Transformations-konzepte

Unsere SysML-Quellen: Buch und OMG-Standard



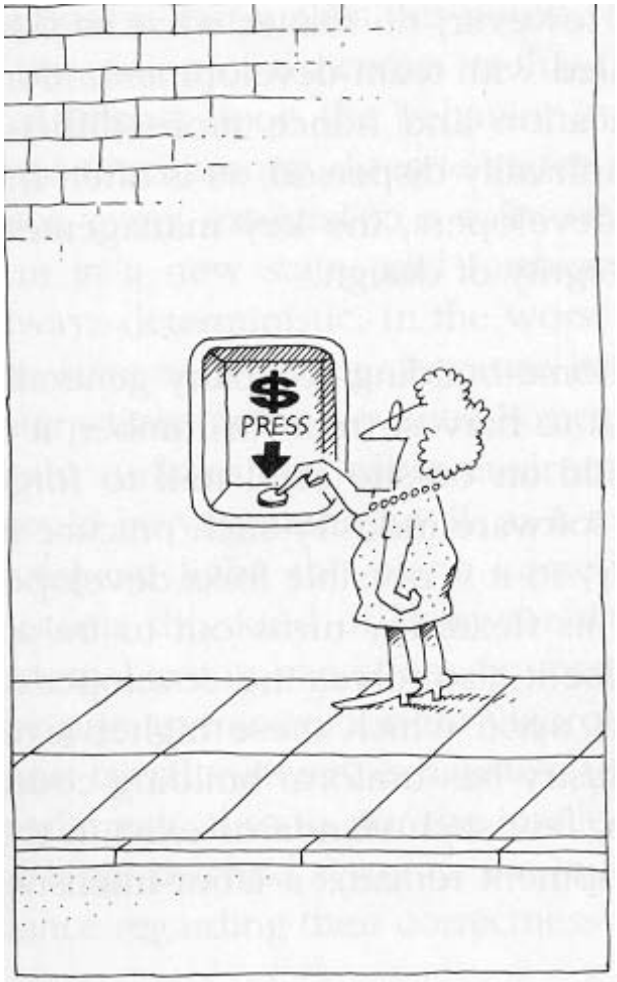
Standard: OMG-Dokument

OMG Systems Modeling Language (OMG SysML™)

Einführung

1. Organisatorisches
2. Trends in der Software-Entwicklung
3. Modelle in UML
4. Grundsätzliches zu Systemen und Modellen
5. Modelle in SysML

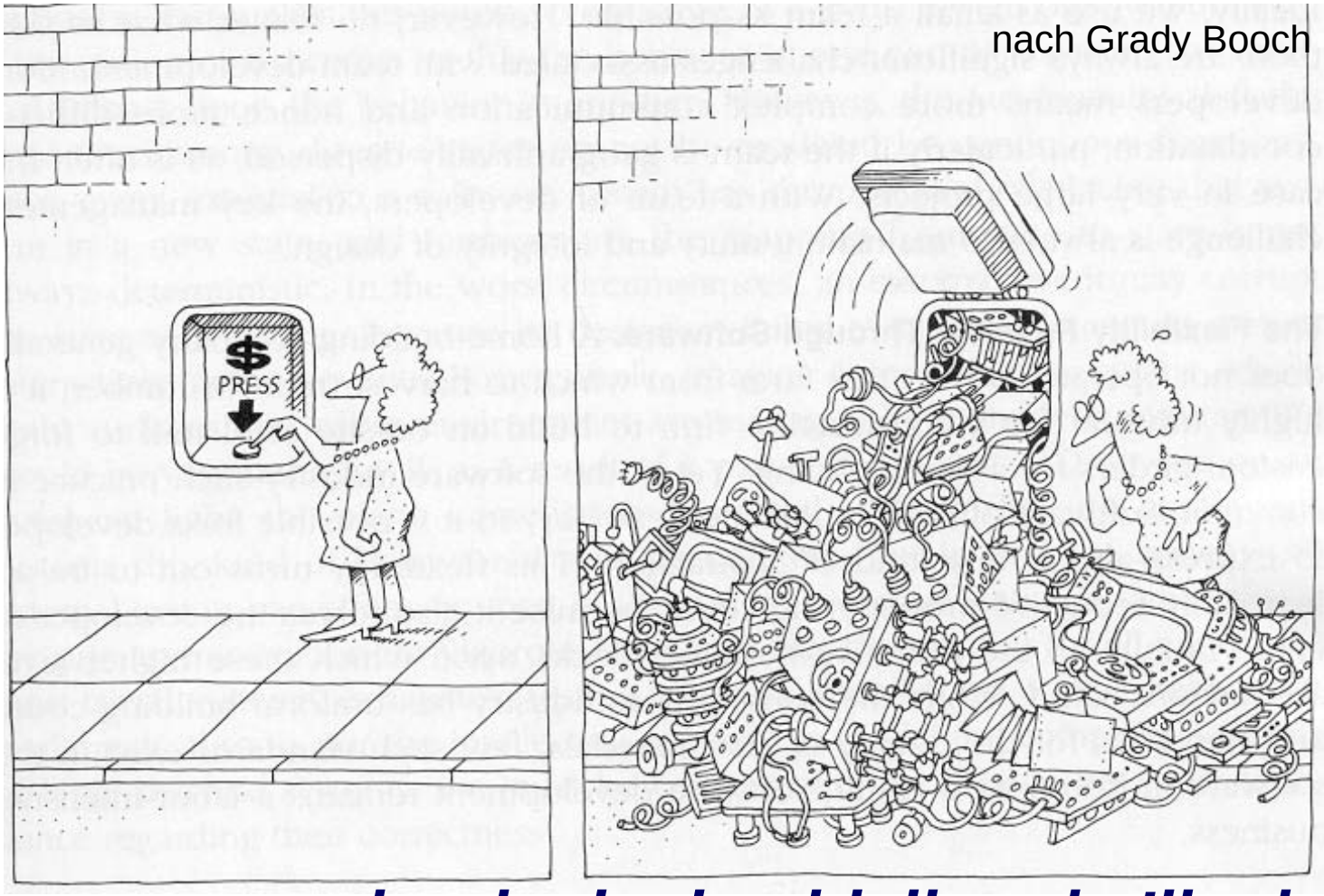
Vereinfachung der Softwareentwicklung



Herausbildung unterschiedlicher Paradigmen

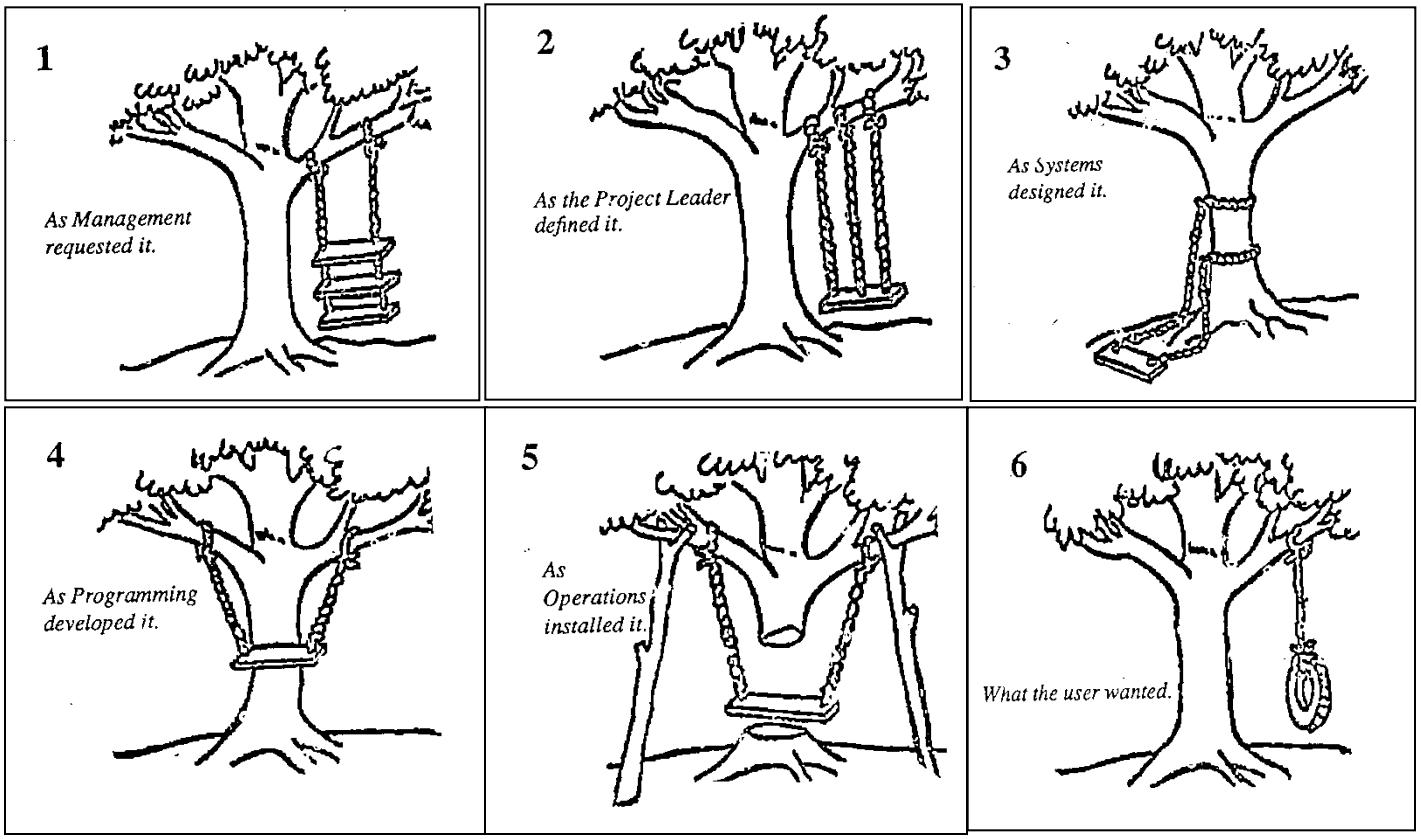
- …
- Objektkompositionsparadigma
- Modelltransformationsparadigma
- Komponentenparadigma

Vereinfachung der Softwareentwicklung

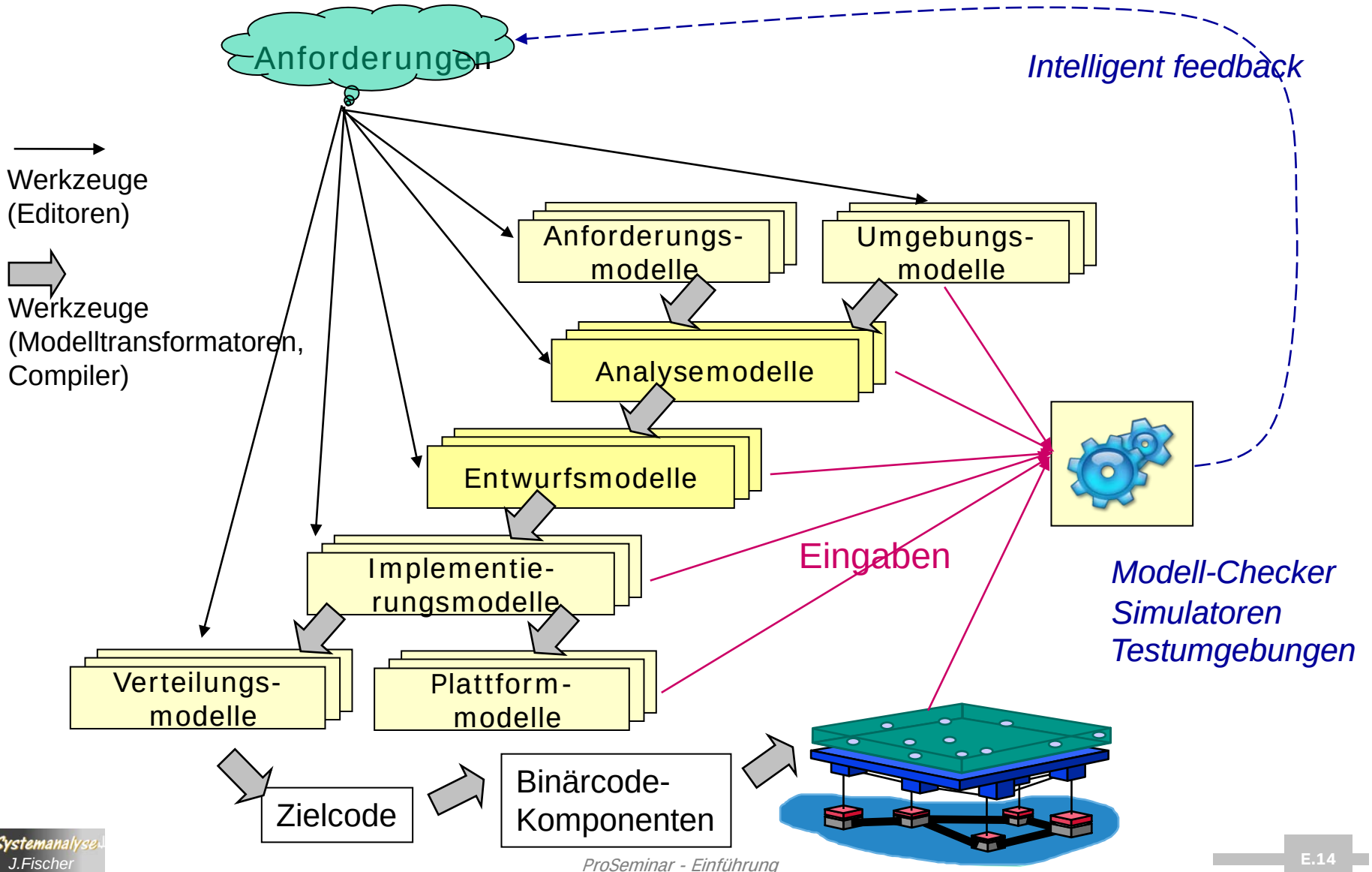


– oder doch eine bleibende Illusion ?

Wozu Modellierung und Computersimulation in der Softwareentwicklung ?



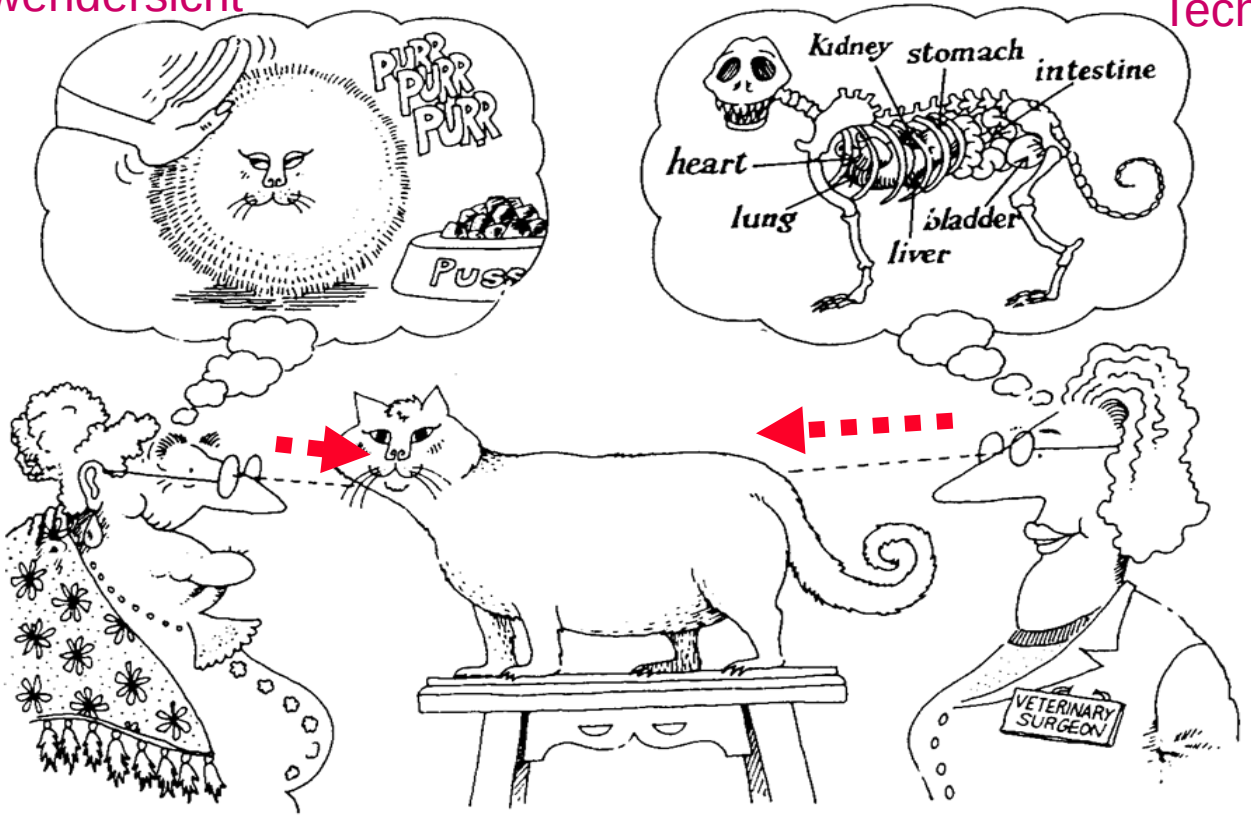
Modellbasierte Entwicklung (vereinfacht)



Modelle in unterschiedlichen Sichten auf einen Realitätsausschnitt

Anwendersicht

Technologiesicht



Das jeweilige Untersuchungsziel bestimmt Sicht und Abstraktionsgrad

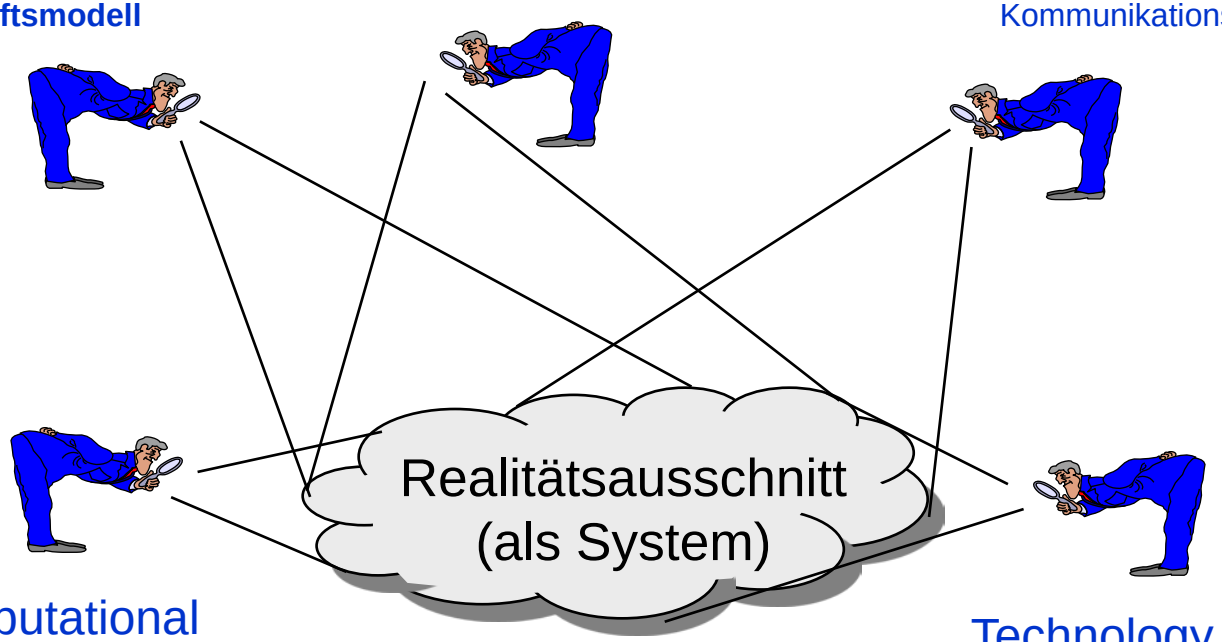
Modellierungskultur in der Softwareentwicklung: Sichtweisen

am Beispiel von ODP (Open Distributed Processing)
ISO- und ITU-Standard zur Modellierung verteilter Software-Systeme

Enterprise
Unternehmens
Anwendungsaspekte
Geschäftsmodell

Information
Informationsdarstellung,
-semantik,-verarbeitung

Engineering
Verteilungsaspekte
Kommunikationsinfrastruktur

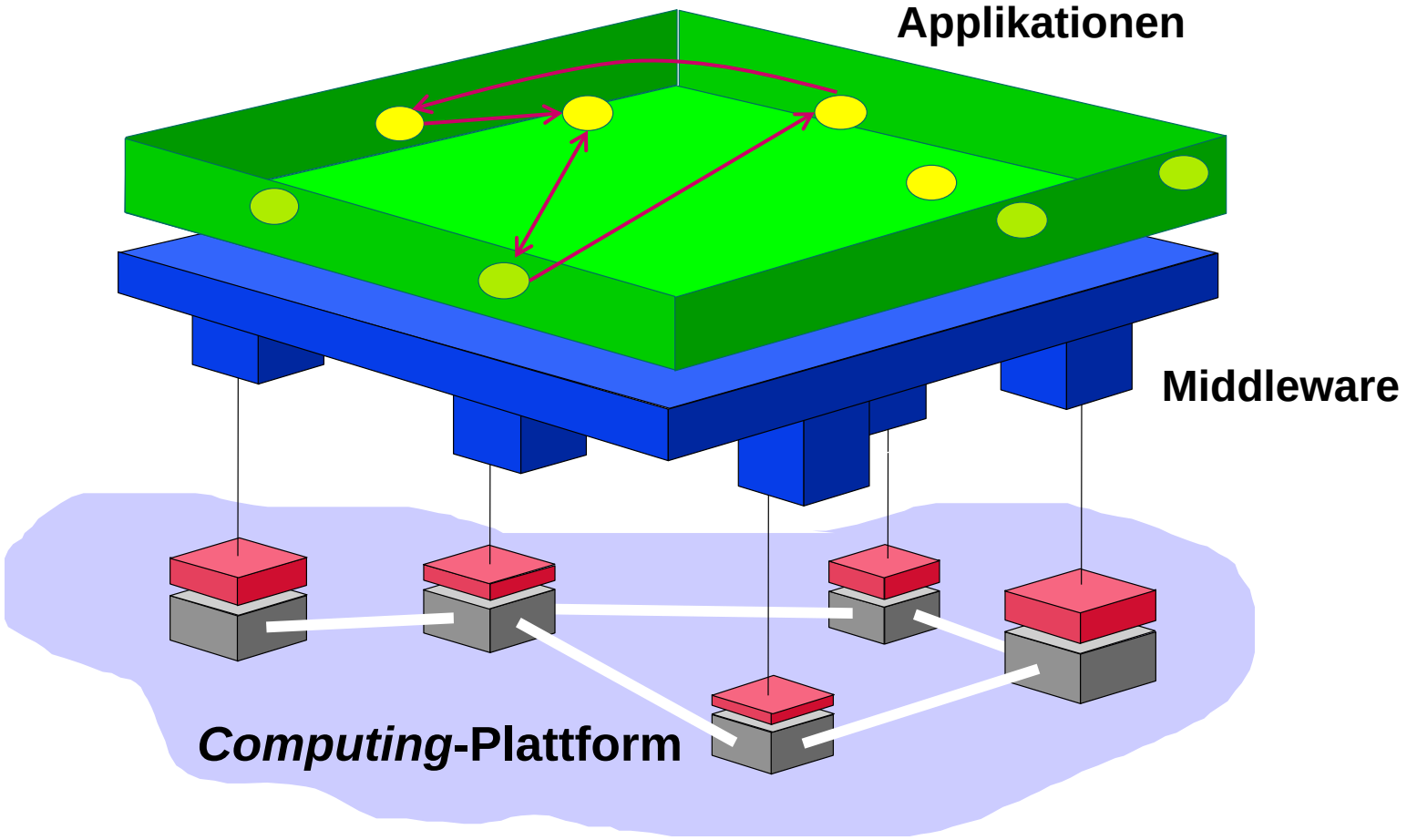


Computational
Dekomposition in verteilbare **Komponenten**
aus **funktioneller** Sicht
Definition von **Schnittstellen**

Technology
Implementationssprache
Werkzeuge
Plattformen

➔ 5 Modellklassen mit spezifischen Darstellungskonzepten

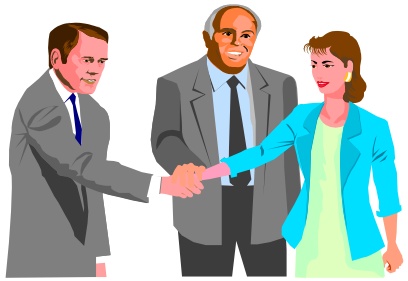
Verteilte Systeme



Herausforderung an die Middleware

- **keine** Vereinbarung über Hardware-Plattformen
- **keine** Vereinbarung über Betriebssysteme
- **keine** Vereinbarung über Netzwerkprotokolle
- **keine** Vereinbarung über anwendungsspezifische Datenformate

vereinbart werden **muss** aber die Interoperabilität



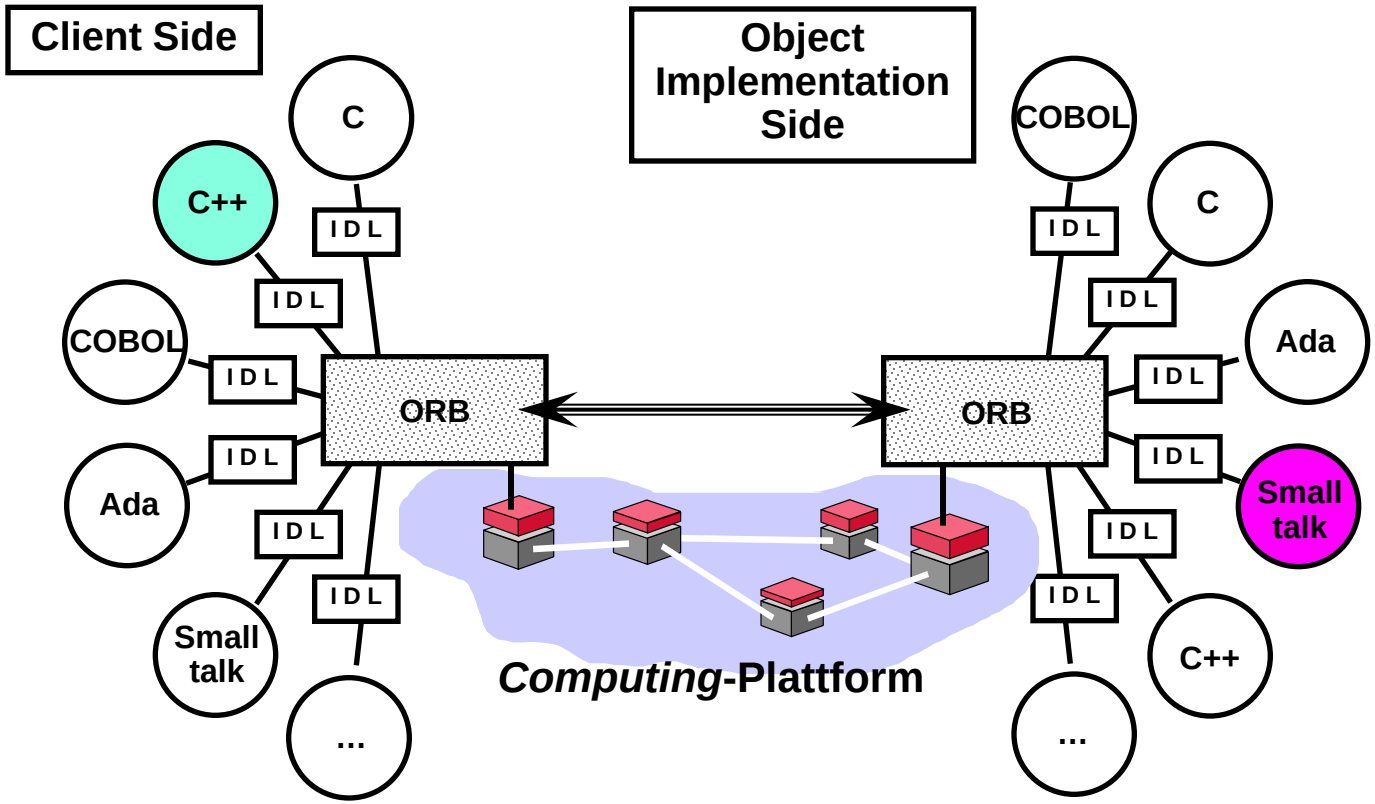
Aufgabe: Entwicklung einer (konsensfähigen) objektorientierten Architektur

- basierend auf Objektorientierung
- Integration verteilter Applikationen
- Garantie für
  - Wiederverwendbarkeit von Softwarekomponenten,
  - Interoperabilität und Portabilität
- basierend auf kommerziell verfügbarer Software

OMG

Sprachunabhängigkeit

Interface Definition Language (IDL)
~ Beispiel für die Bedeutung abstrakter Modelle in der SW-Entw.

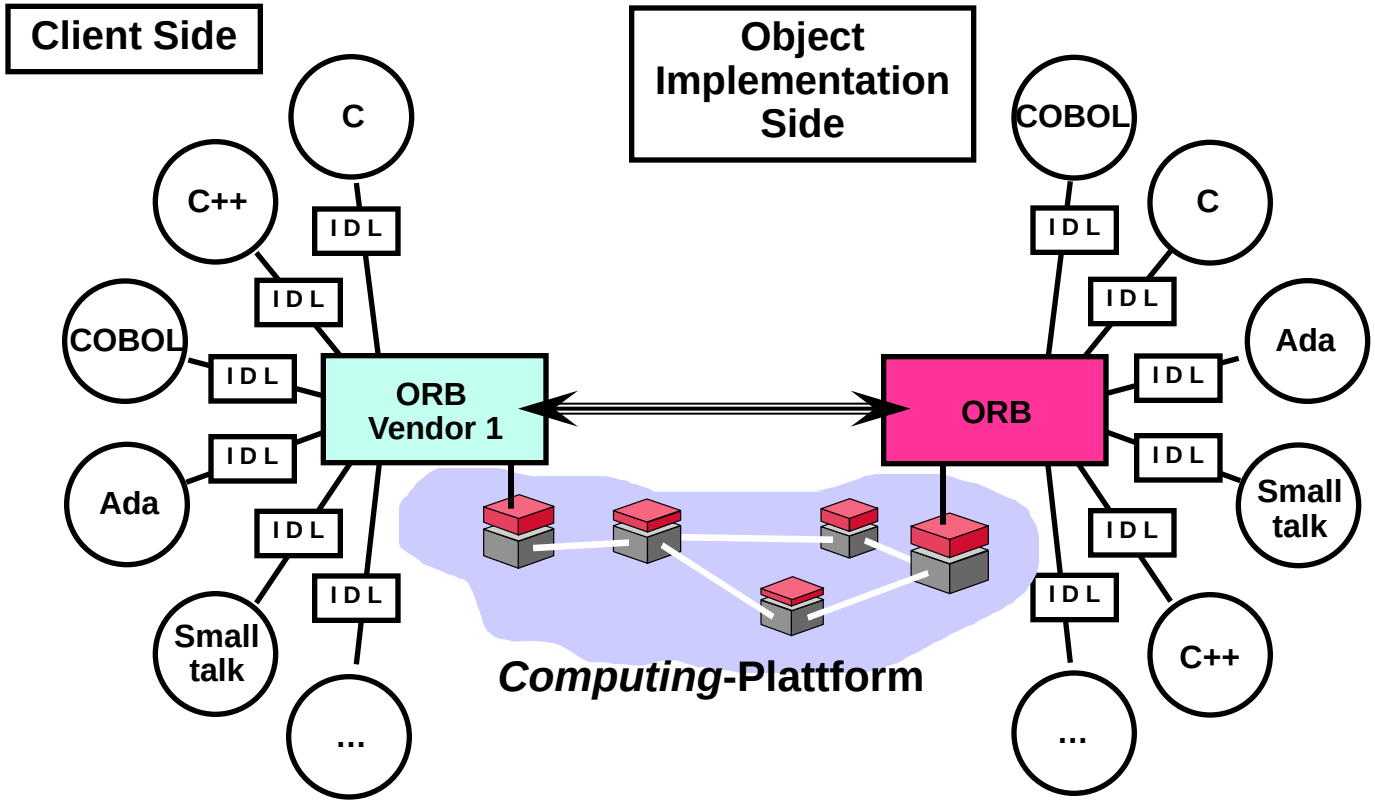


Grundbegriffe aus MDA-Dokumenten

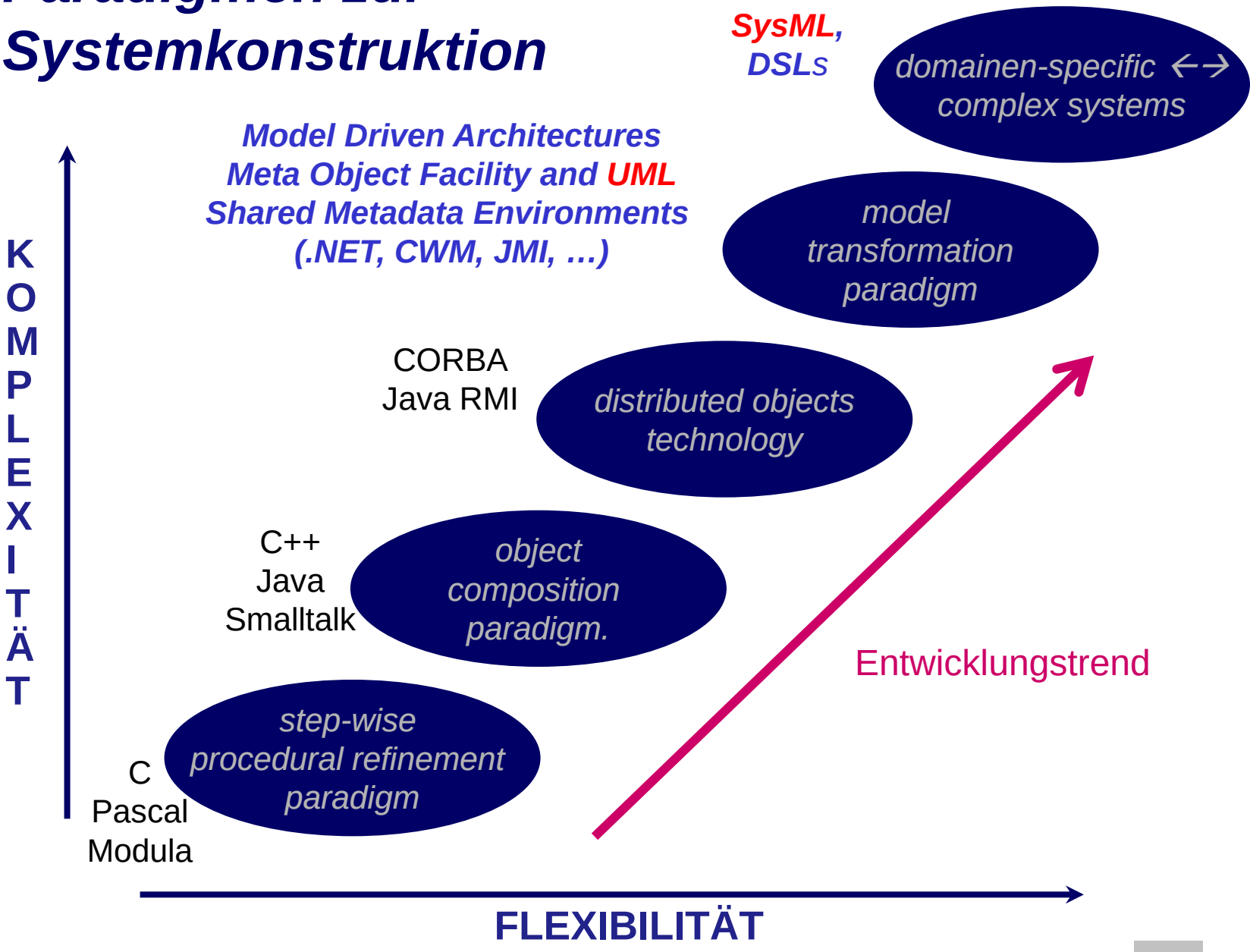
- *Model*: Representation of (part of) function, structure and/or behaviour of a system
- *Specification*: Presentation of a model based on a language (syntax + semantics)
- *Platform*: Technological and engineering details irrelevant to fundamental functionality of a software component
- (MDA = Model Driven Architecture)

Hersteller- und Übertragungstechnologie-Unabhängigkeit

Interface Definition Language (IDL)

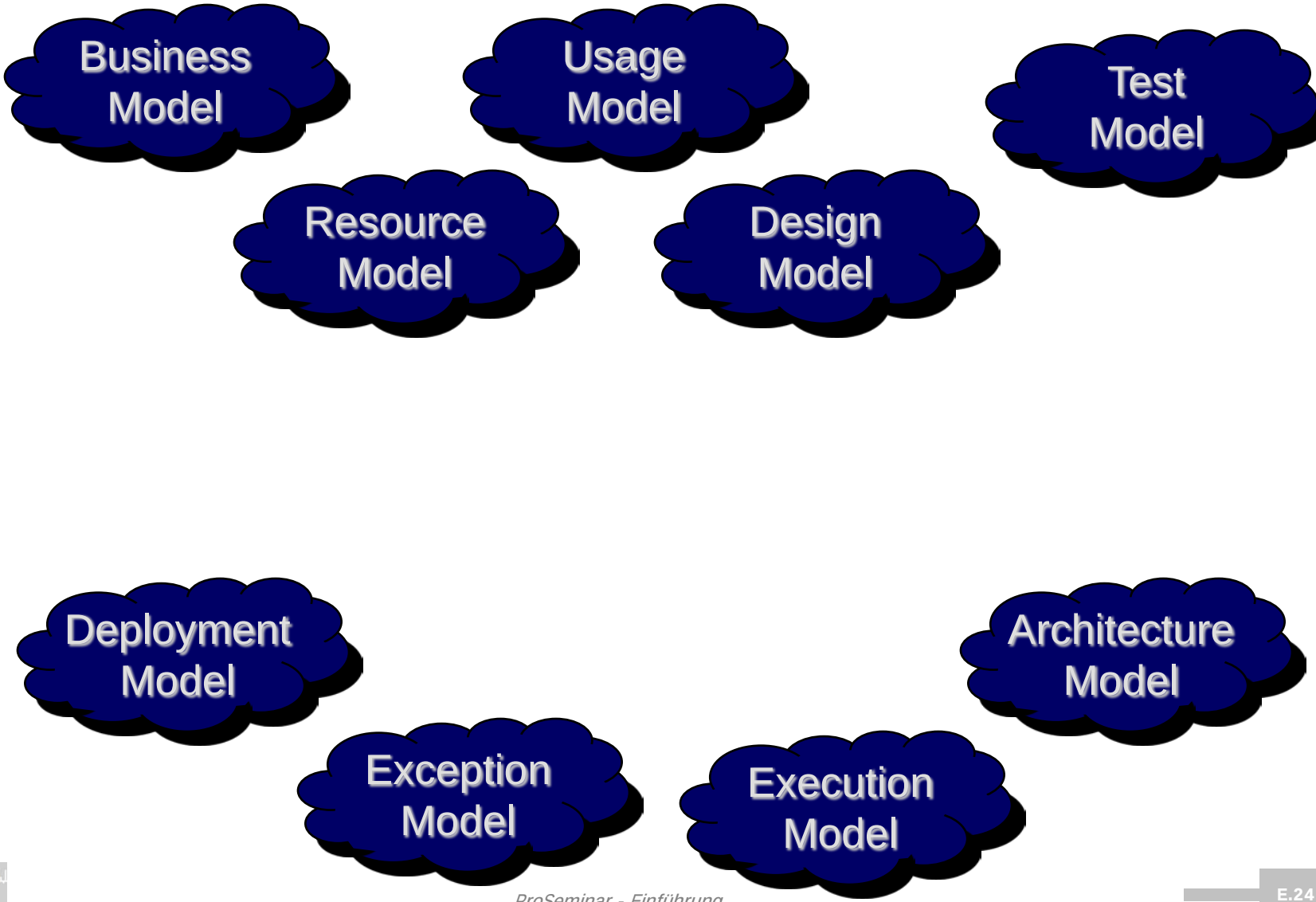


Paradigmen zur Systemkonstruktion



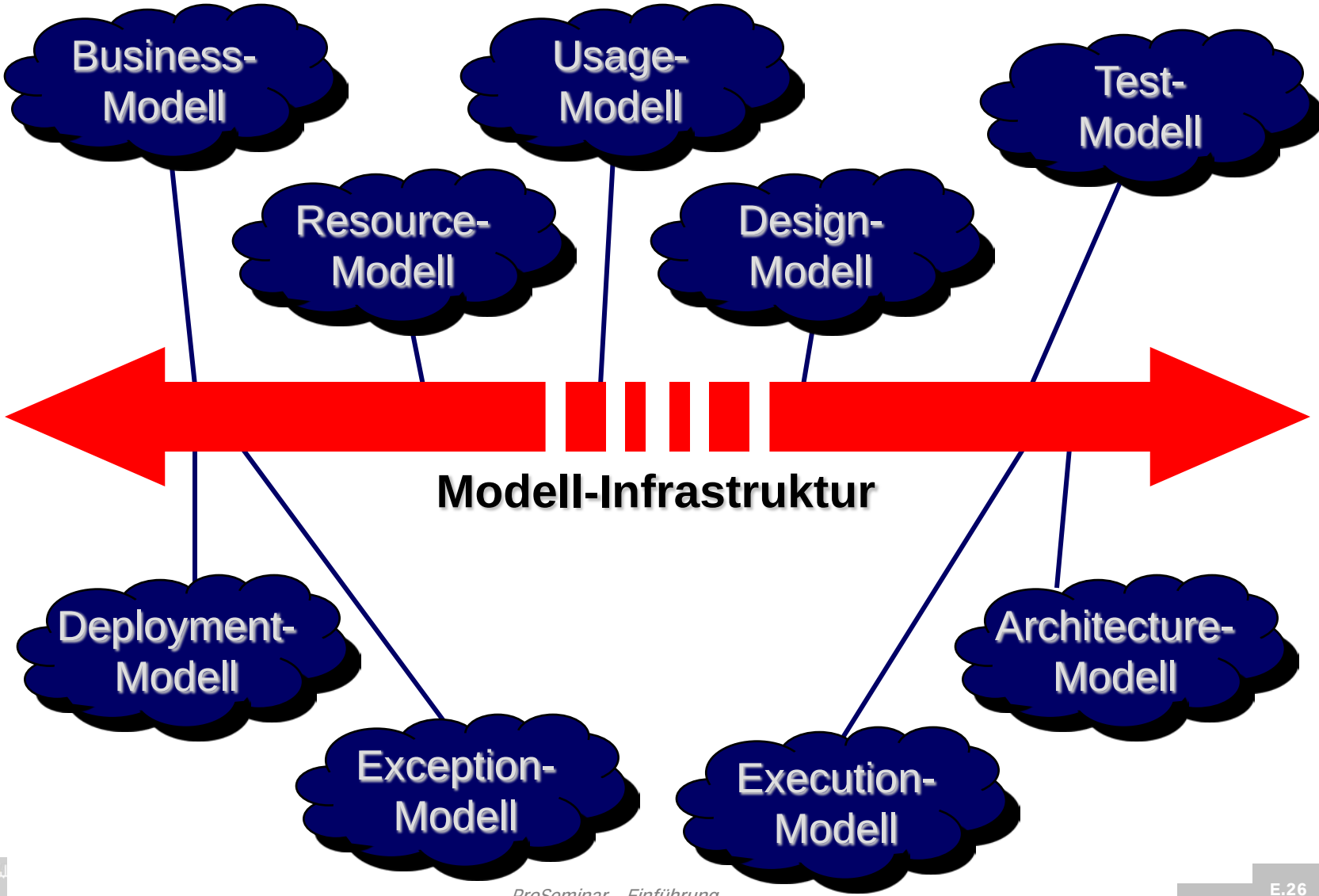
Bei der modell-getriebenen SW-Entwicklung werden in Modellen alle Aspekte eines Softwaresystems erfasst

Infrastruktur zur Modellverarbeitung



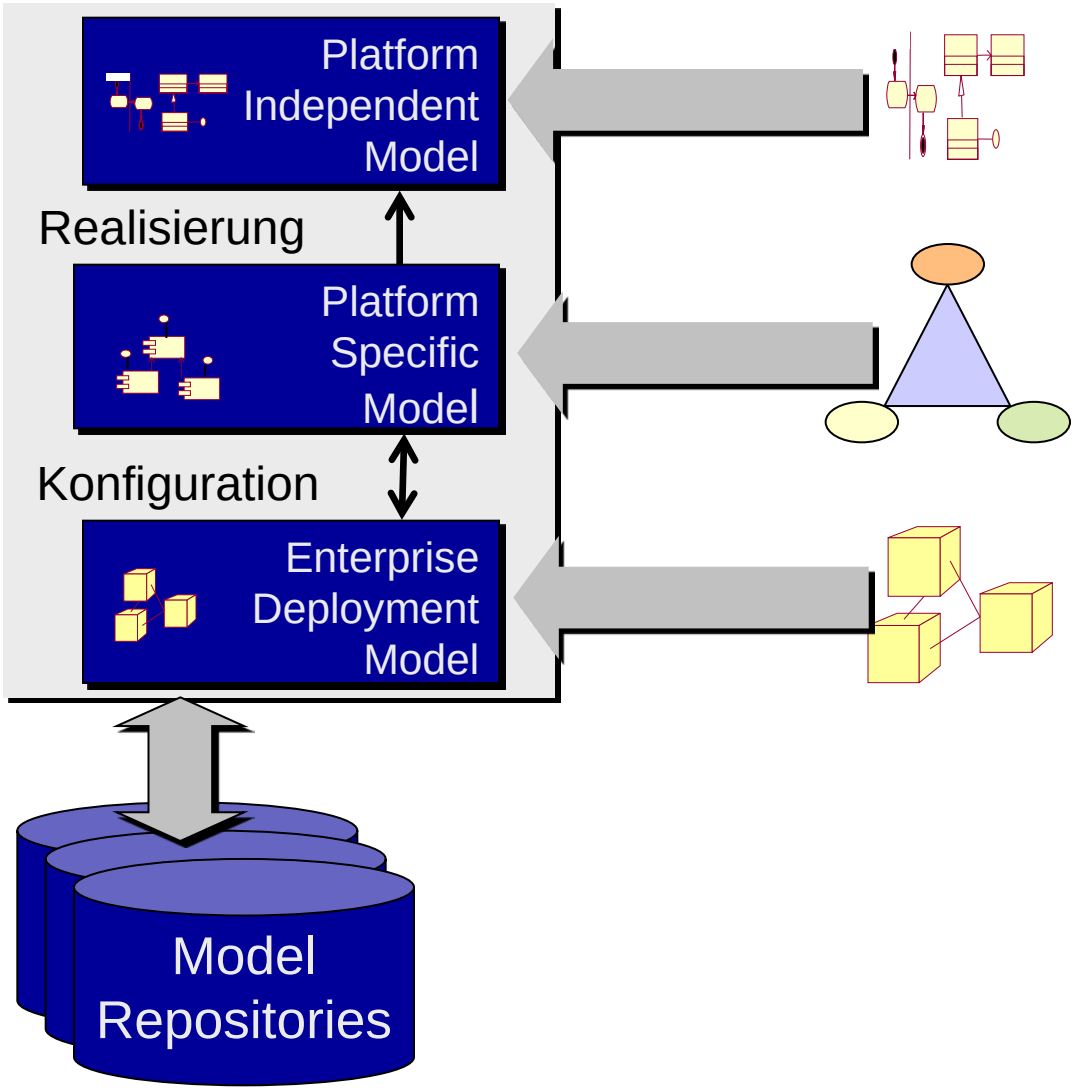
Modelle eines Systems und
die Modelle der Modelle werden in
Beziehung gesetzt, ausgetauscht und
transformiert.

Modell-Infrastruktur



Bei der Systemkonstruktion nach MDA werden Modelle weiter kategorisiert:
„platform independent“ und
„platform specific“.

Modelle, Repositories und Mapping



Domänenmodelle
Modelle von Fach-verfahren
Unternehmensmodelle

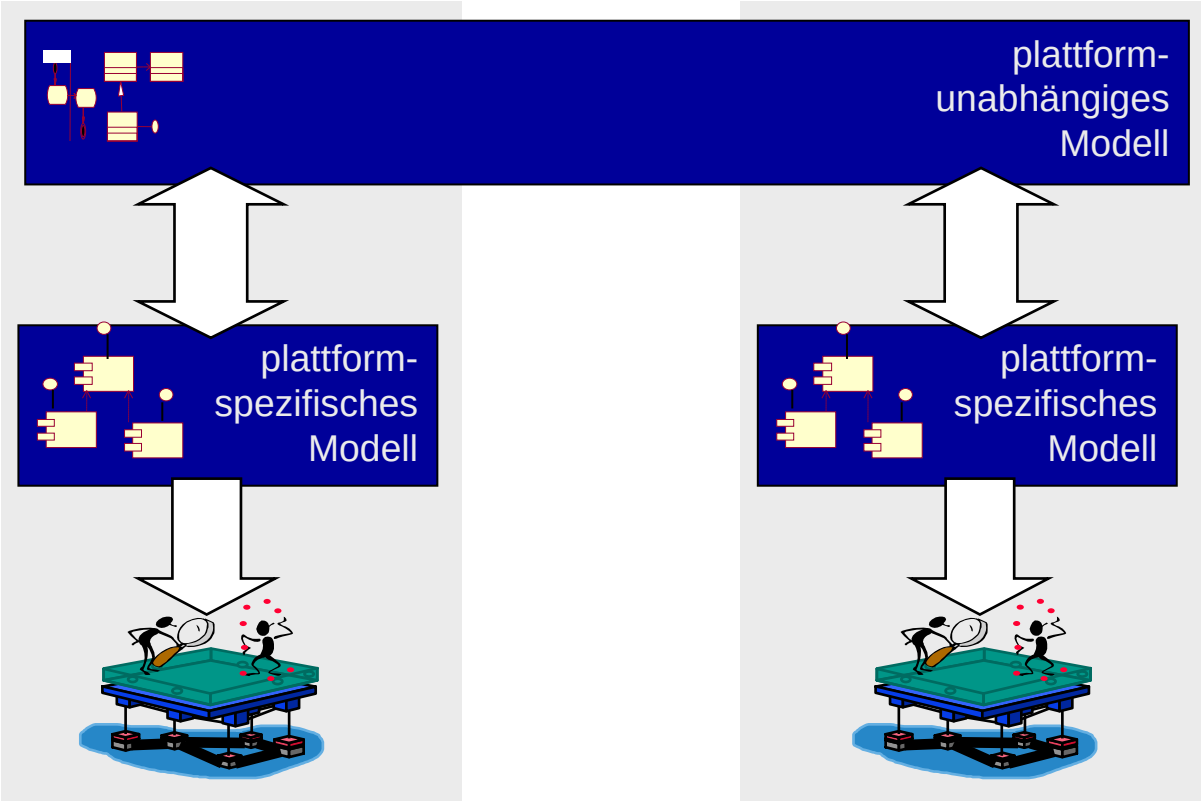
Plattformspezifische Artefakte

Unternehmens-Spezifische Konfigurations- und Betriebs-aspekte

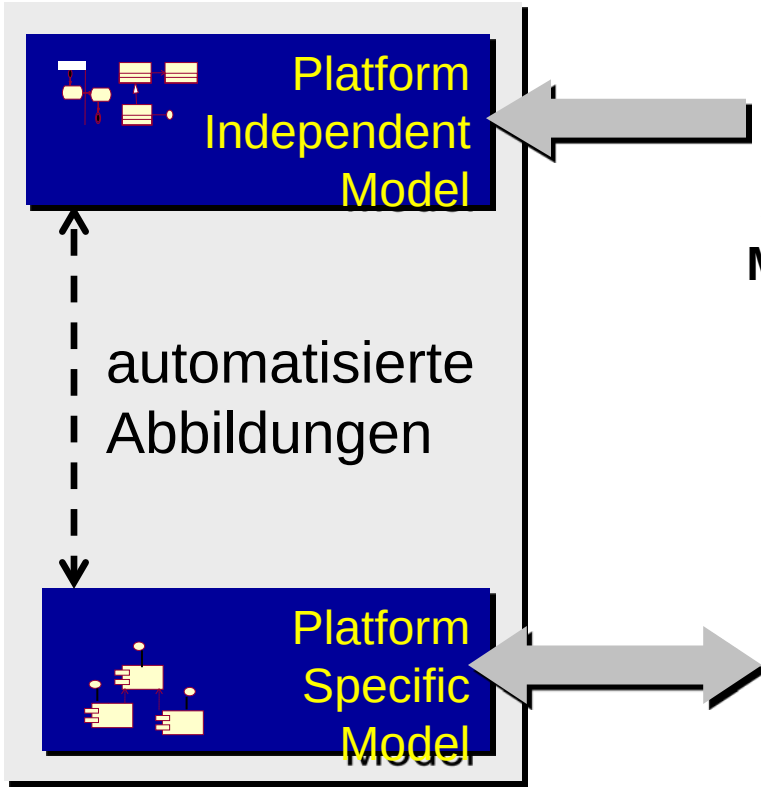
Vorteil der Modellkategorisierung

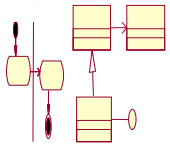
- Unabhängigkeit vom „Next Big Thing“:

  Aus stabilen Modellen spezifischer Anwendungsklassen können Realisierungsmodelle für unterschiedliche Softwareplattformen gewonnen werden.



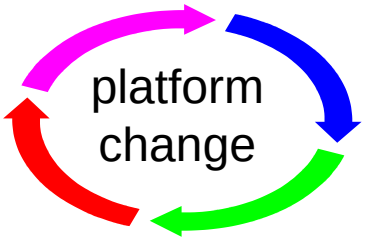
Umgang mit Änderungen





stabile Modelle

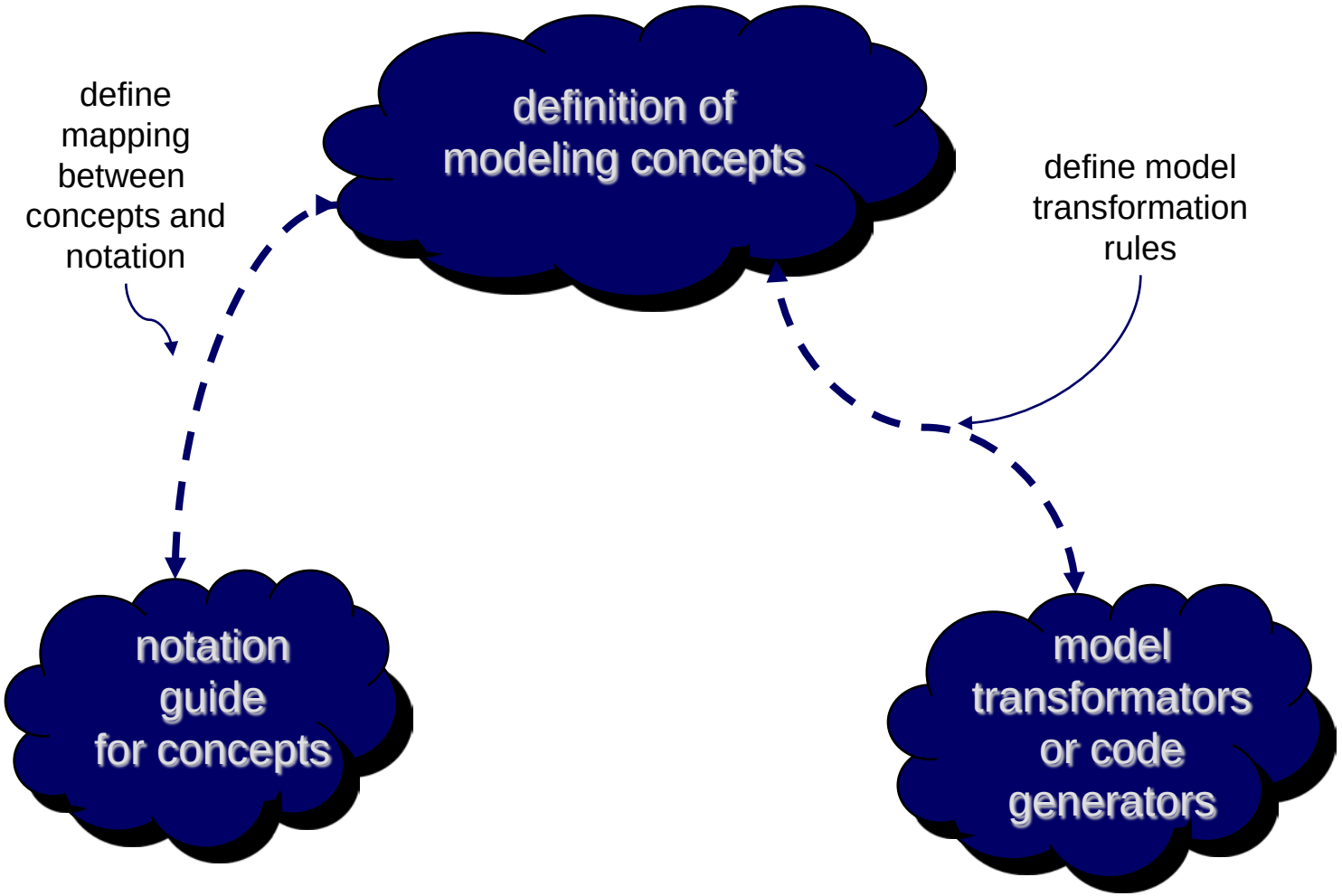
- (formale) Semantik
- Infrastrukturunabhängigkeit
- lange Lebensdauer
- geringerer Wartungsaufwand



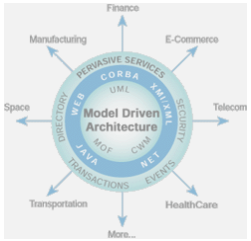
- verfahrensunabhängige Infrastrukturevolution
- Integration von bestehenden Anw.
- Verringerung der Migrationskosten

Trennung von
- Konzeptdefinition,
- Notation und
- Modelltransformation

„Separation of Concerns“



Vorteile der MDA



- Programmiersprachen-, Hersteller- und Middleware-unabhängig
- Skalierbarkeit, Robustness, Sicherheit ... implementiert in der bestmöglichen Art und Weise
- Reduktion der gesamten Entwicklungskosten
- Bessere Wiederverwendbarkeit durch Model Repositories
- Geringere Entwicklungszeit für neue Anwendungen
- Reibungslose Integration über Grenzen unterschiedlicher Middleware-Plattformen hinweg
- Schnelleres Einbeziehen neuer Technologien in existierende Systeme

Einführung

1. Organisatorisches
2. Trends in der Software-Entwicklung
3. Modelle in UML
4. Grundsätzliches zu Systemen und Modellen
5. Modelle in SysML

UML-Charakterisierung

- ist eine Notation/Sprache, keine Methode,
  Framework für UML-ähnliche Modellierungssprachen

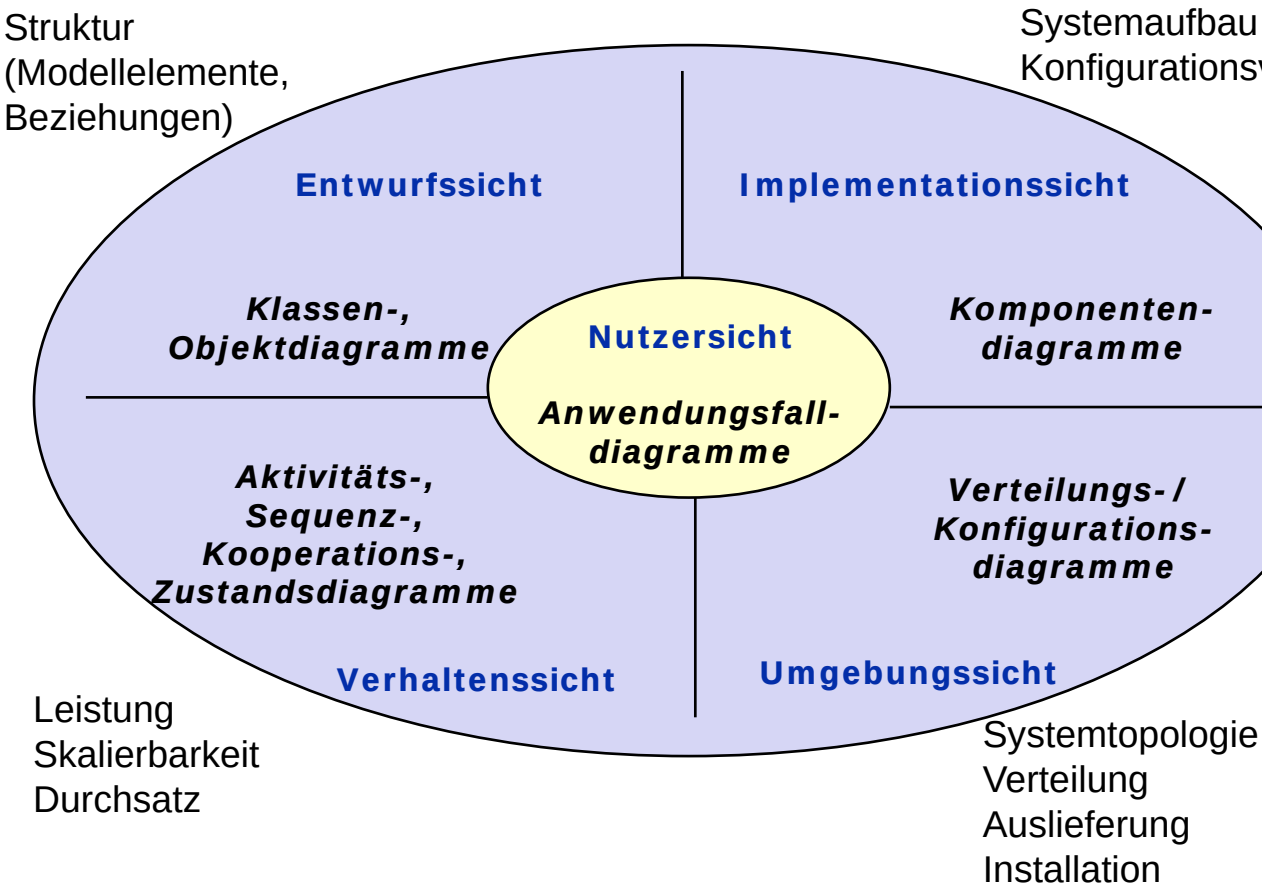
- abstrahiert von
  - architekturellen Vorgaben,
  - Design- und Implementierungs-Styles,
  - Technologien (Software, Hardware, Infrastrukturen, ...),
  - Entwicklungsprozessen

- standardisiert
  - Begriffswelt (Modellierungskonzepte),
  - Semantik (Bedeutung der Modellierungskonzepte),
  - visuelle Darstellung (Notation der Modellierungskonzepte)

- führt Ideen verschiedener Techniken zusammen
  - Booch, OMT, Jacobsson, ROOM, SDL, EDOC, MSC, Component Based Modeling, ...

Systemarchitekturmodell eines Systems

- … unter Verwendung von UML



Vier Arten von Modellierungselementen/Entitäten

nämlich zur

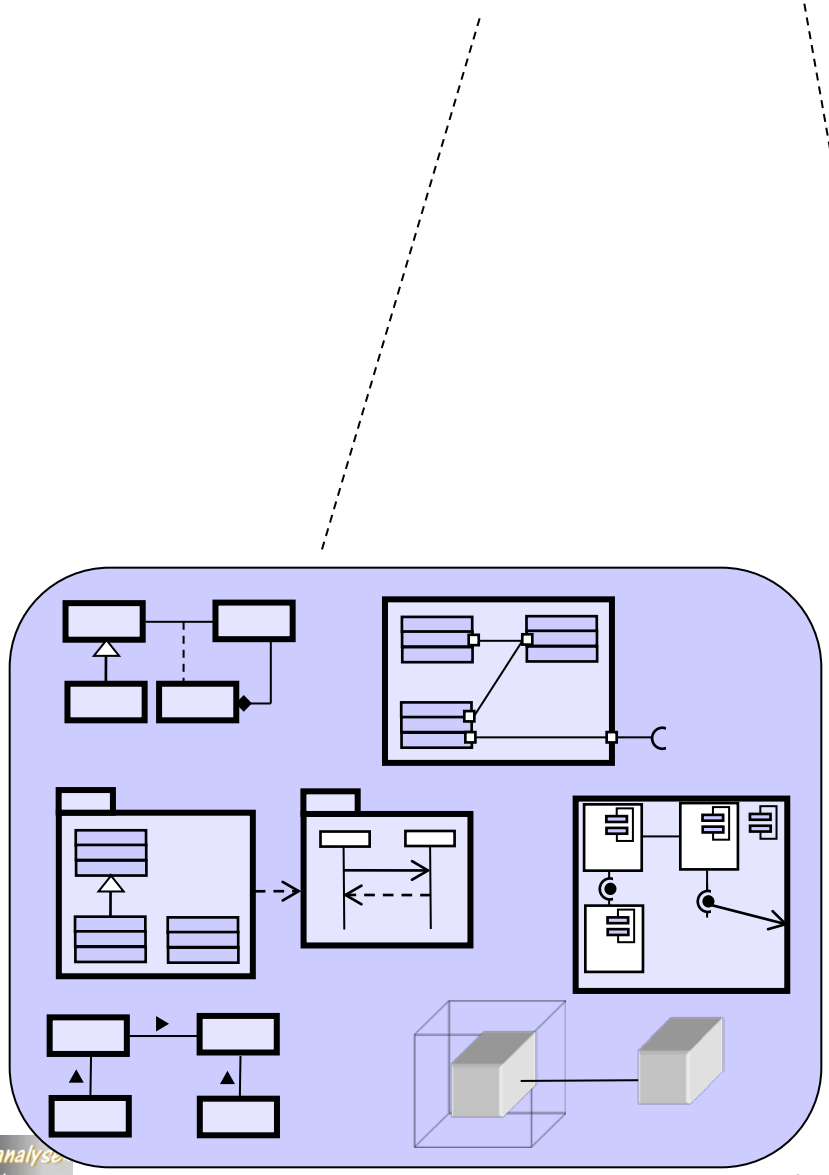
- Bildung von Strukturen
- Beschreibung von Verhalten
- Bildung von Gruppierungen von Entitäten
- Formulierung von Anmerkungen

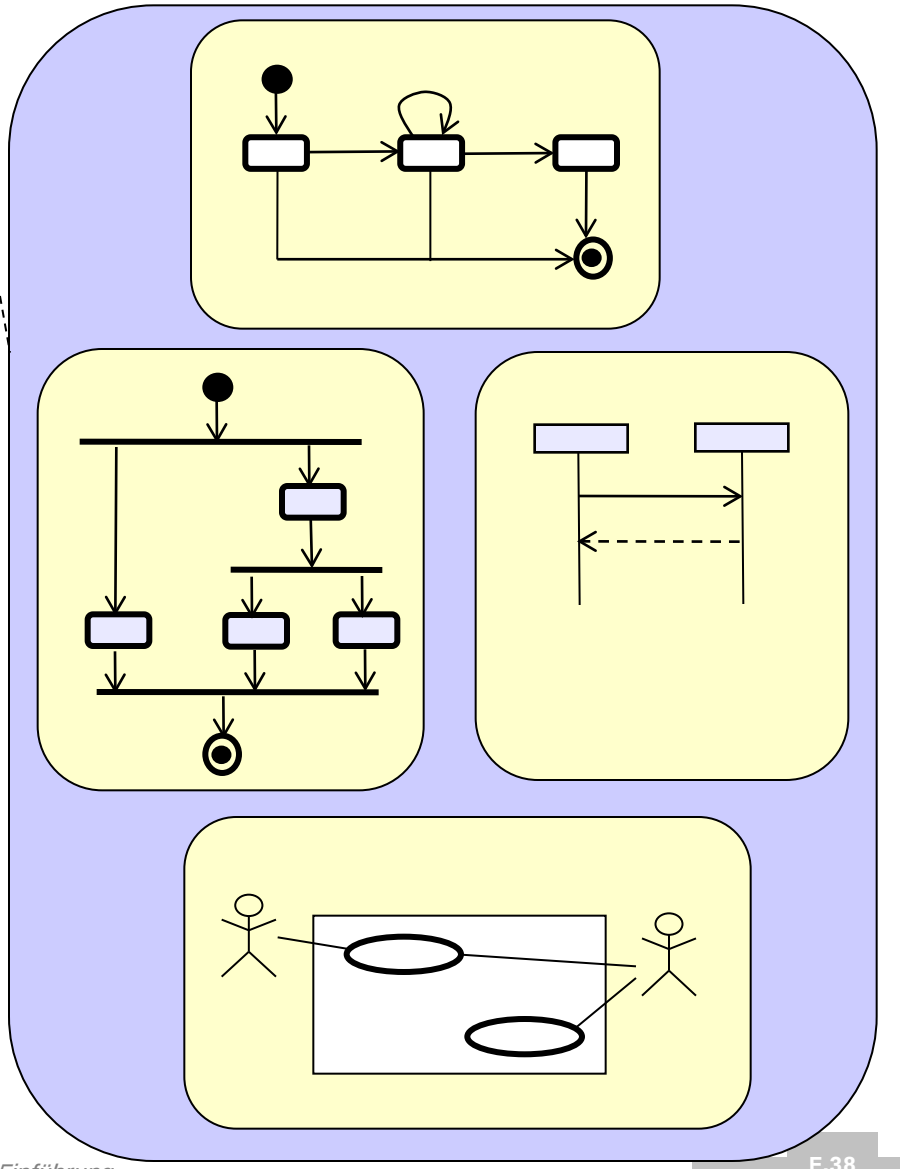
Objekte struktur- und verhaltensbeschreibender Klassen bilden Strukturen

Objekte verhaltensbeschreibender Klassen bei Nutzung Objekte normaler Klassen

➔ damit lassen sich UML-Modelle komplett beschreiben

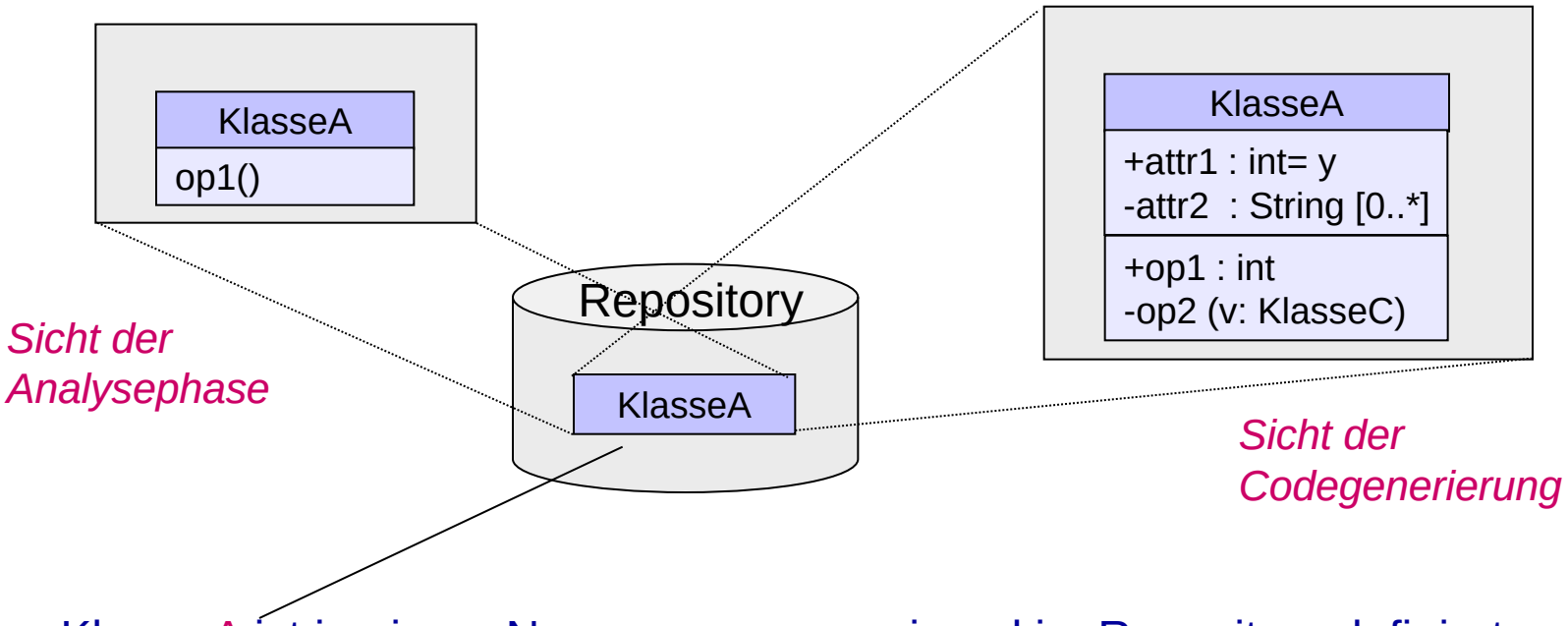
Diagrammarten zur Anordnung von Entitäten: Struktur und Verhalten





Sichten

- ein und dasselbe Modellelement
  kann unterschiedlich detailliert für unterschiedliche Sichten dargestellt werden



Klasse A ist in einem Namensraum nur einmal im Repository definiert

Die UML

551 v. Chr. bis 479 v. Chr.

„Wenn die Sprache nicht stimmt,
ist das was gesagt wird, nicht das, was gemeint ist.“
(Konfuzius)

- UML = Unified Modeling Language
- … ist zunächst Standardsprache (der OMG) zur Visualisierung, Spezifikation, Konstruktion und Dokumentation komplexer Softwaresysteme
- … kombiniert Konzepte der
  - Objektorientierten Modellierung
  - Datenmodellierung (Entity-Relationship-Diagramme)
  - Business-Modellierung (Work Flows)
  - Komponentenmodellierung
  - Verhaltensmodellierung (Erweiterte Zustandsautomaten)
  - ...
- UML-Modelle sind in erster Linie graphische Repräsentationen in Form von Diagrammen