ModSoft

Modellbasierte Software-Entwicklung mit UML 2 im WS 2014/15

Einführung

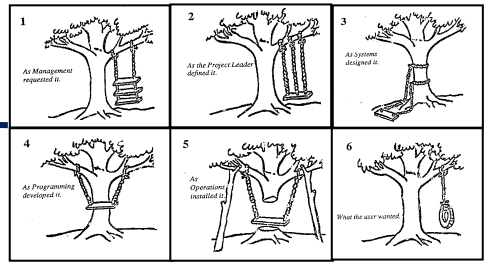
Prof. Dr. Joachim Fischer
Dr. Markus Scheidgen
Dipl.-Inf. Andreas Blunk

fischer@informatik.hu-berlin.de

Teil I: Einführung

1. MDD als Trend in der Software-Entwicklung
2. UML, ein erster Blick
3. Begriffe: Systeme und Modelle
4. Paradigmen der UML-Modellierung
5. Historie von UML
6. Modellierungselemente von UML im Überblick
7. Struktur des UML-Standards

Technische Grundlage für automatische Modelltransformation (Wdh.)



Anforderungen

unterschiedliche Modellierungssprachen mit Syntax und Semantik

Anforderungs-modelle

Umgebungs-modelle

Analysemodelle

Entwurfsmodelle

Implementie-rungsmodelle

Verteilungs-modelle

Plattform-modelle

Zielcode

Binärcode-Komponenten

emf
ECLIPSE MODELING FRAMEWORK

EMF
Eclipse Modeling Framework

einheitliche,
universelle
Konzeptbasis
für unterschiedliche
Modellklassen
mit freier konkreter Syntax

Ecore= EMOF

Meta-Modelle, formuliert mit MOF-Konzepten
zur Definition der abstrakten Syntax
von Sprache-1, … Sprache-n
dynamische Semantik ?

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

Wie wollen in ModSoft aber mehr !

Zentrale UML-Modellelement-Typen (Wdh.)

aus diesen Konzepten wird die gesamte UML aufgebaut

1. ***Classifiers***.

A classifier describes a **set of objects**. An object is an individual with a **state** and **relationships** to other objects. The state of an object identifies the values for that object of properties of the classifier of the object. (In some cases, a classifier itself may also be considered an individual; for example, see the discussion of static structural features in sub clause 9.4.3.)

2. ***Events***.

An event describes a set of **possible occurrences**. An *occurrence* is something that happens that has some consequence with regard to the system.

3. ***Behaviors***.

A behavior describes a set of **possible executions**. An *execution* is a performance of a set of actions (potentially over some period of time) that may generate and respond to **occurrences of events, including accessing and changing the state of objects**.

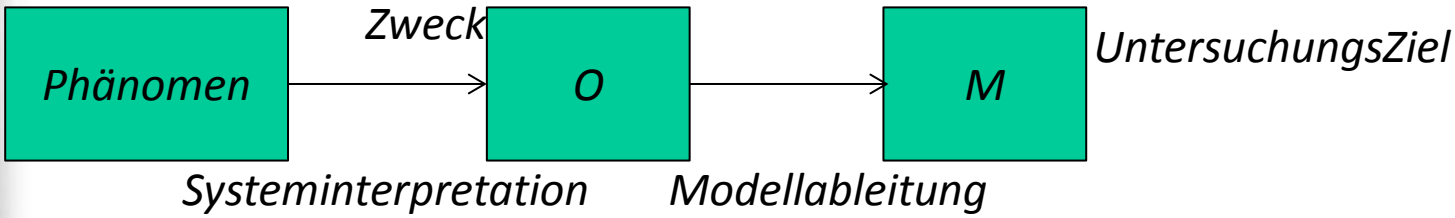
(As described in sub clause 13.2, behaviors are themselves modeled in UML as kinds of classifiers, so that executions are essentially modeled as objects. However, for the purposes of the present discussion, it is clearer to consider behaviors and executions to be in a separate semantic category than classifiers and objects.)

Quelle: OMG Unified Modeling Language TM (OMG UML), *Version 2.5*

Teil I: Einführung

1. MDD als Trend in der Software-Entwicklung
2. UML, ein erster Blick
3. Begriffe: Systeme und Modelle
4. Paradigmen der UML-Modellierung
5. Historie von UML
6. Modellierungselemente von UML im Überblick
7. Struktur des UML-Standards

Originale, Modelle betrachtet als System



- **Originale**

  sind Ausschnitte einer gedachten oder real existierenden Welt als System
  Systembegriff:
  - Systemzweck liegt vor/ d.h. Systemfunktionalität ist von außen erkennbar
  - Abgrenzung des Systems zu seiner (System-)Umgebung

    (was gehört zum System zur Erfüllung des Zwecks und was nicht)
  - Systemstruktur
    (Komponenten/Elemente des Systems und ihre Relationen untereinander),
  - Systemverhalten als Zusammenspiel des Verhaltens der einzelnen Komponenten bei Interaktion mit der Umgebung,

    wobei Rückkopplungsfreiheit zwischen System und Umgebung verlangt wird (evtl. notwenige Verschiebung der Systemgrenze)
- Originale als dynamische oder statische Systeme möglich
  (Struktur u. Verhalten sind zeitabhängig oder nicht)

Modelle: Präzisere Begriffsbestimmung

Modelle sind

- Abstraktionen von Originalen

  Abstraktionen= Vereinfachungen (in Zweck, Struktur, Verhalten)
- aus einer bestimmten Sicht
- mit einer bestimmten Zielstellung
- und damit vereinfachte Abbilder der Realität

Modelle sind

- Abstraktionen kompletter Systeme oder einzelner Systemelemente mit typischer Unterteilung: Struktur- und Verhaltensmodell

Modelle

- helfen, die zu entwickelnden Systeme besser zu verstehen
- ermöglichen die Spezifikation von Struktur und Verhalten komplexer Systeme
- dienen als Vorlagen zum Bau realer Systeme
- dokumentieren getroffene Entwurfsentscheidungen

Gibt es perfekte Modelle?

- kein einziges Modell,
  keine einzige Sicht ist ausreichend um ein komplexes System zu erfassen
  → es gibt kein Modell an sich

  dennoch:
  Entscheidung, welche Modelle erzeugt werden,
  hat großen Einfluss auf die Modelluntersuchung

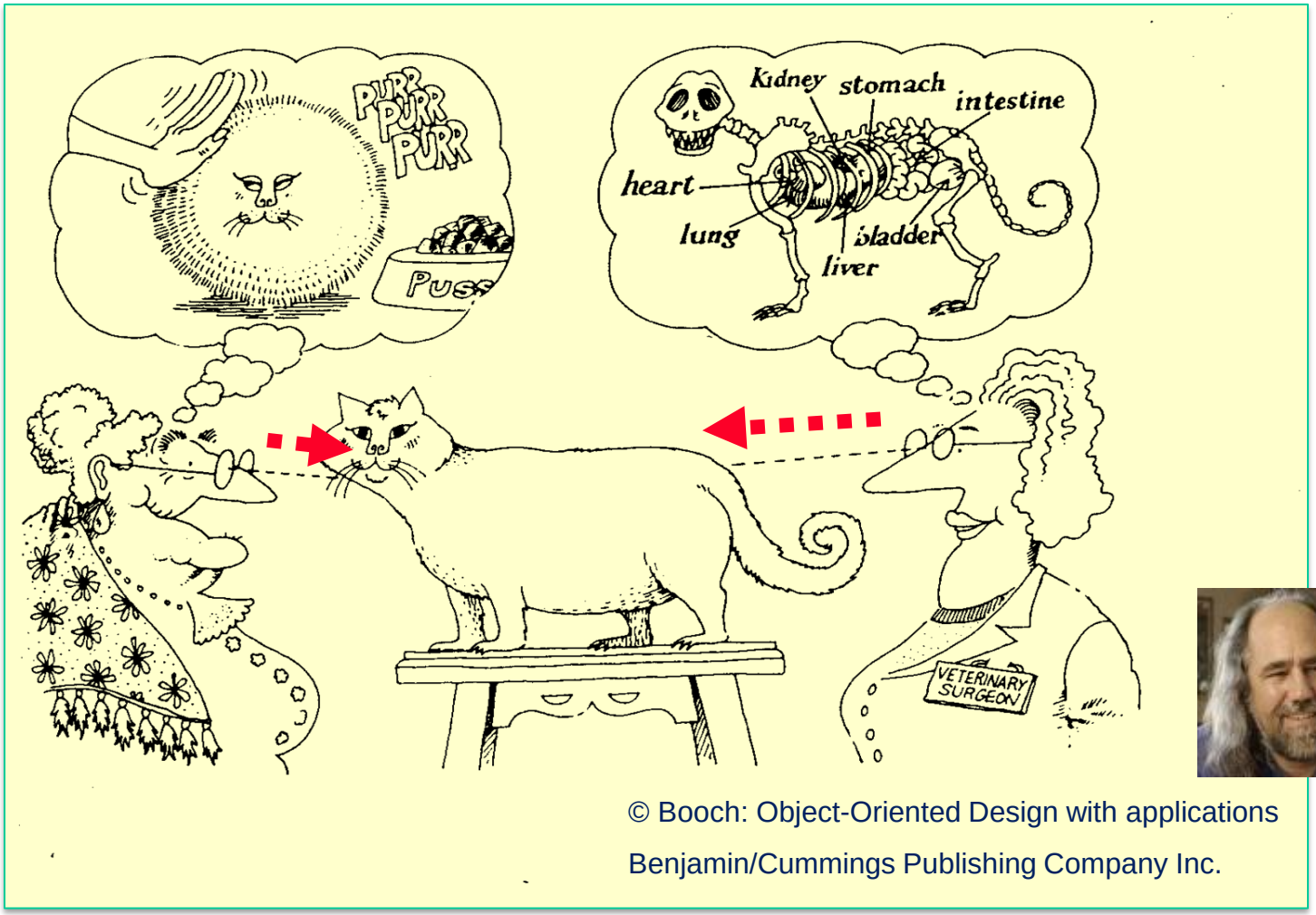
- die besten Modelle sind realitätsnah (aber Widerspruch insich)

Regel: Modelliere nicht so genau wie möglich, sondern nur so genau wie nötig.

- **Gefahr**: bereits bewährte Modelle
  werden ungeprüft für Untersuchungen
  mit anderem Untersuchungsziel eingesetzt

Regel: Verlieb Dich nicht in Dein Modell.

Modelle *in unterschiedlichen Sichten auf einen Realitätsausschnitt*

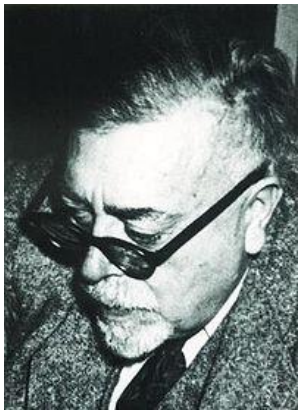


© Booch: Object-Oriented Design with applications

Benjamin/Cummings Publishing Company Inc.

Das jeweilige Untersuchungsziel bestimmt Sicht und Abstraktionsgrad

Begrenztheit von Modellen



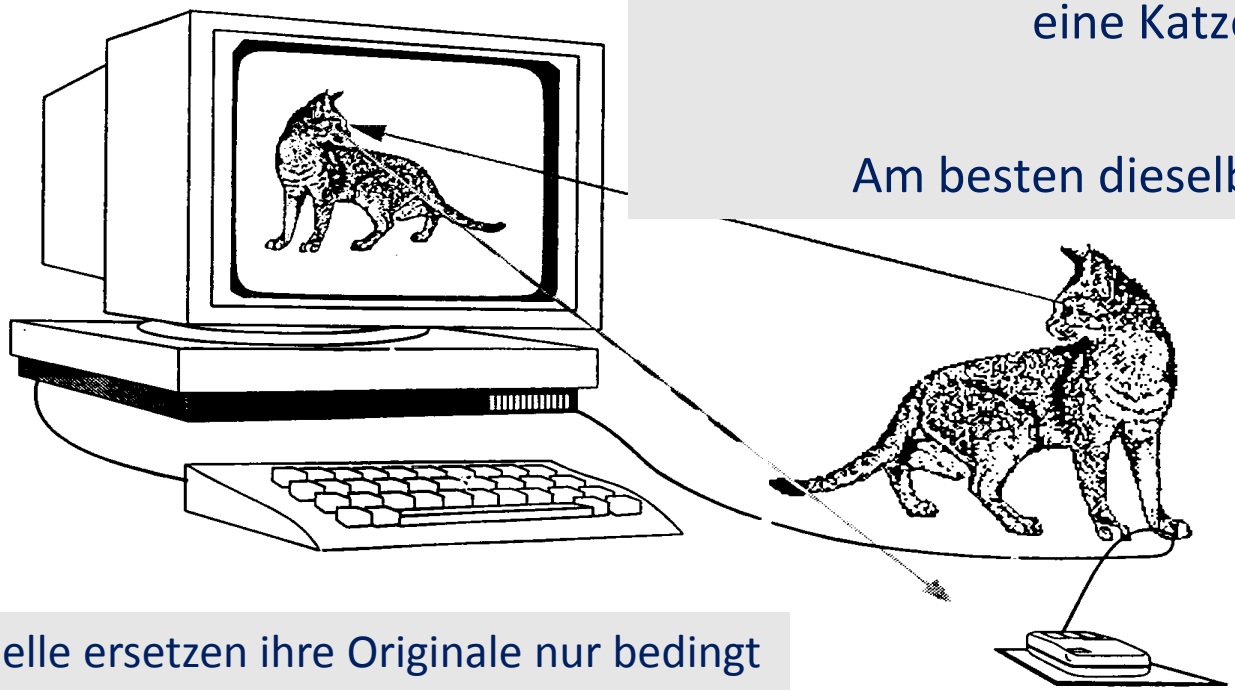
Norbert Wiener (1894-1964)
Begründer der Kybernetik, Kommunikation, Steuerung und Regelung

FRAGE: Was ist das beste Modell einer Katze ?

scherzhaft:

„Das beste Modell einer Katze ist ...
eine Katze.

Am besten dieselbe Katze.“



Modelle ersetzen ihre Originale nur bedingt

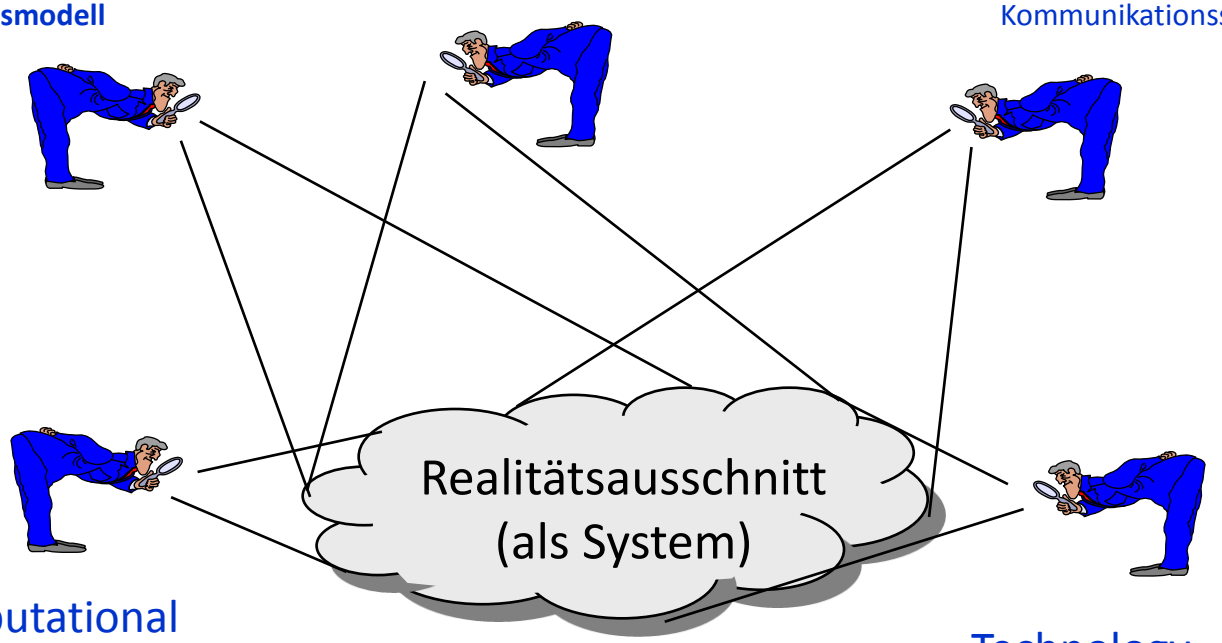
Modellierungskultur in der Softwareentwicklung: Sichtweisen

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
Kommunikationsstruktur



Computational
Dekomposition in verteilbare **Komponenten**
Definition von **Schnittstellen**

Technology
Implementationssprache
Werkzeuge
Plattformen

5 Modellklassen mit spezifischen Darstellungskonzepten

Modell (aus Sicht der UML-Autoren)

- Definition …aus dem UML-Standard

A **model** captures a view of a physical **system**.

It is an abstraction of the physical system, with a certain **purpose**.

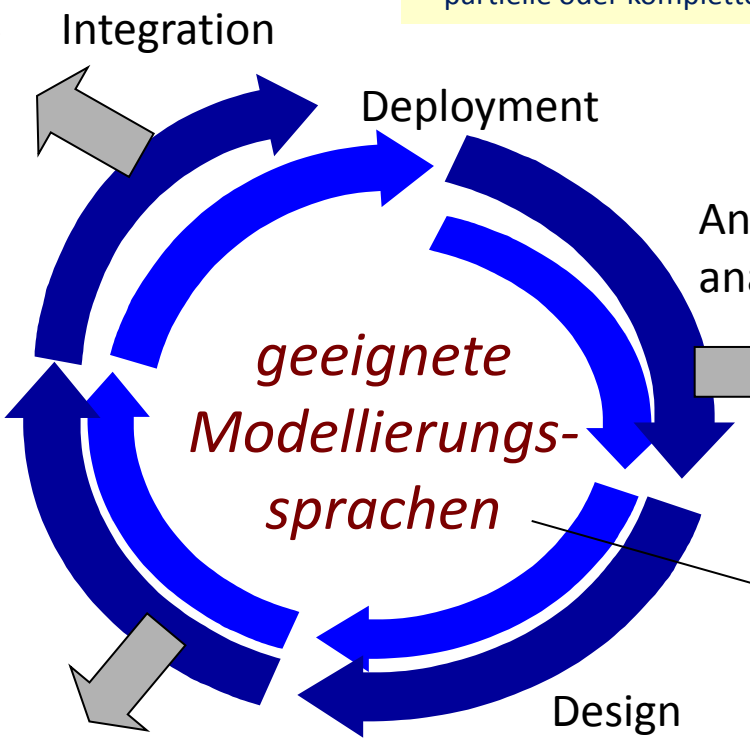
This purpose determines what is to be included in the model and what is irrelevant.

Thus the model completely describes
those aspects of the physical system
that are relevant to the purpose of the model,
at the appropriate level of detail.

SW-Entwicklungsprozess: *spiralförmig, inkrementell & iterativ*

MDD:= Model Driven Development
- SW-Entwicklung ist modellzentriert (Modelle begleiten gesamten SW-Lebenszyklus)
- automatische Transformationen für Modellübergänge
- spezifische Analysen (Checker, Simulatoren, …)
- partielle oder komplette Codegenerierung

Test funktionaler und nicht-funktionaler Rückkkopplungen



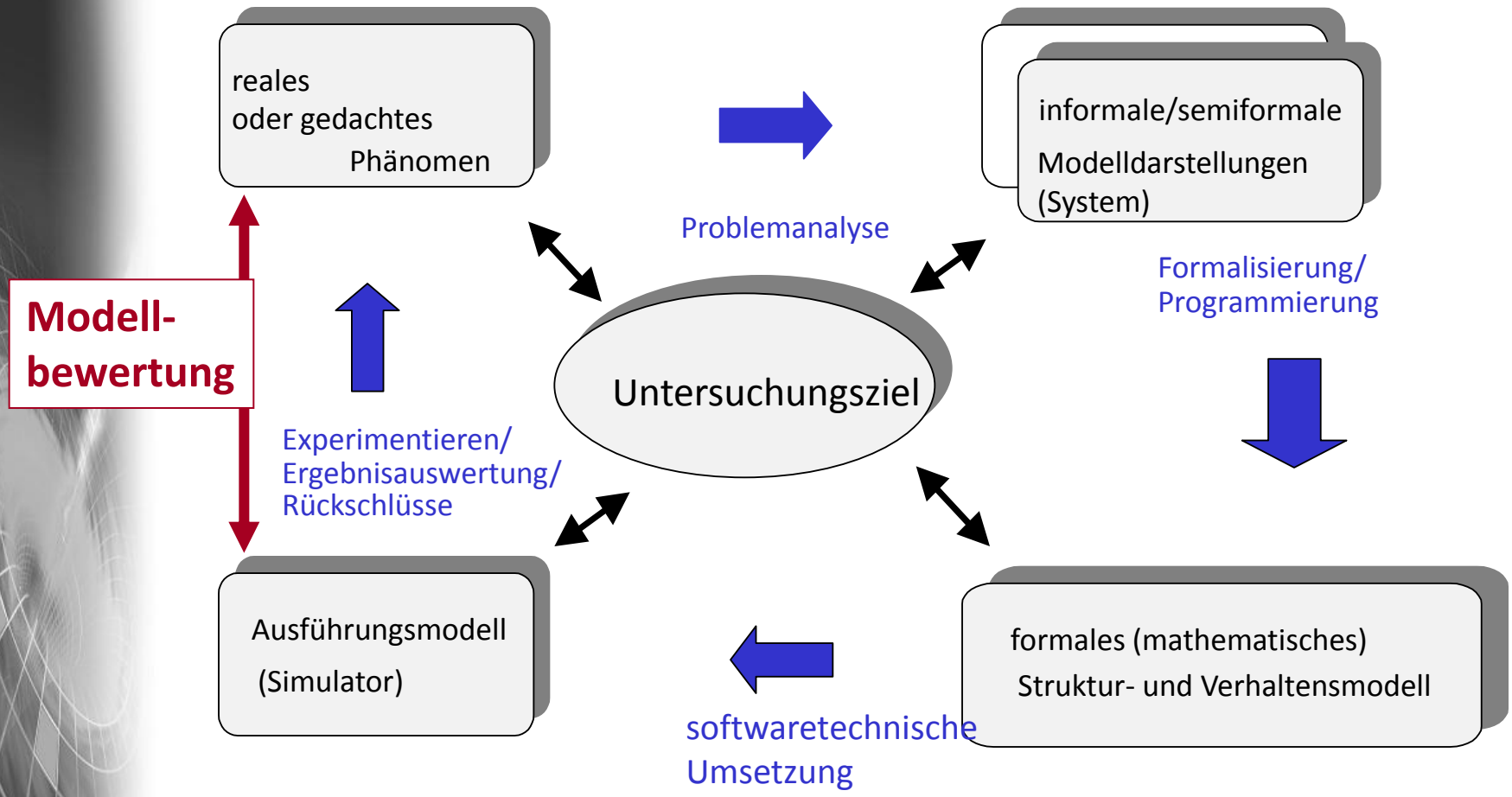
Test nicht-funktionaler Eigenschaften

Test funktionaler Eigenschaften

Allgemeine Vorgehensweise

bei der Computersimulation

~ Modul: OMSI

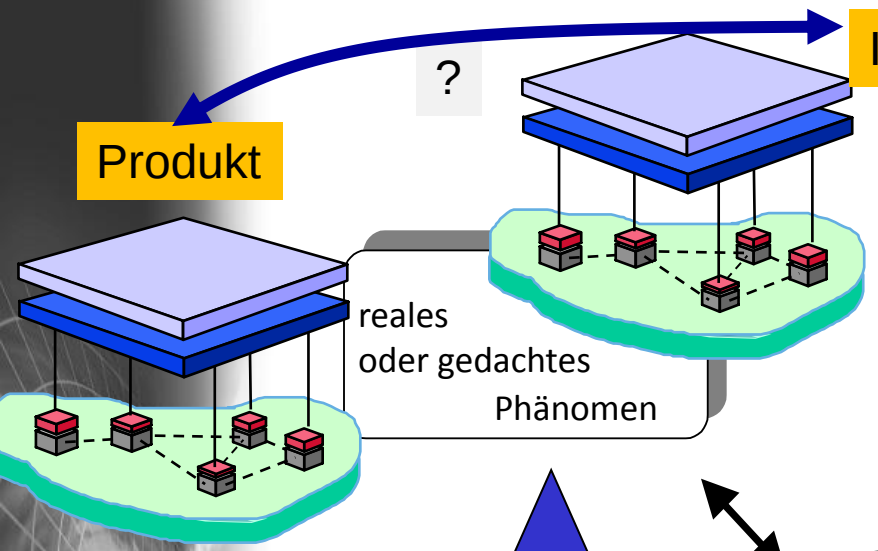


Modelle werden aus einer bestimmten Sicht bei Verfolgung eines bestimmten Untersuchungsziels abgeleitet

Spezielle Vorgehensweise

Sonderfall:
Modell und Original liegen als Software vor

Projekt
- 1996-98 IN-Vermittlungszentrale



informale/semiformale
Modelldarstellungen
(System)

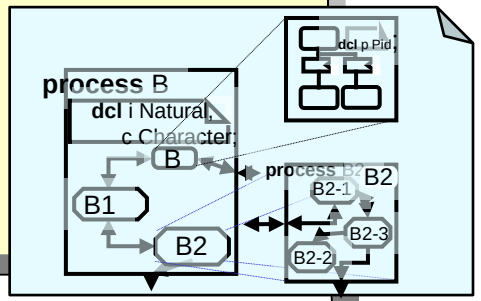
Problemanalyse

Software-Modelle

Zielcode Optimierung

C++ Zielcode-Generierung

Entwicklungsziel/
Untersuchungsziel



softwaretechnische Umsetzung

Ausführungsmodell
(ODEMx-Simulator)

Simulationscode-
Generierung:
ASN.1, SDL → C++

formales (mathematisches)
Struktur- u.
Verhaltensmodell

Funktionaler Test

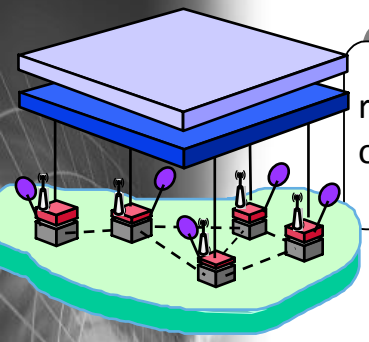
Spezielle Vorgehensweise

Projekt
- 2007-09 SOSEWIN (GK METRIK)

Idee

?

Produkt



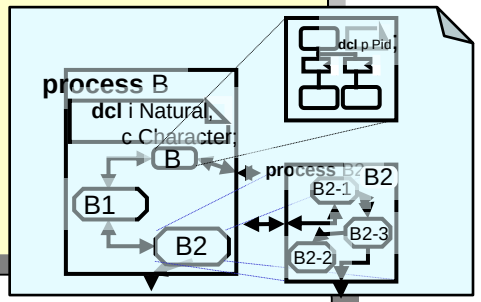
reales
oder gedachtes
Phänomen

informale/semiformale
Modelldarstellungen
(System)

Problemanalyse

Software-Modelle

Untersuchungsziel



C++ Zielcode-Generierung

softwaretechnische Umsetzung

Ausführungsmodell
(ODEMx-Simulator)

Simulationscode-
Generierung:
UML, ASN.1,SDL → C++

formales (mathematisches)
Struktur- u.
Verhaltensmodell

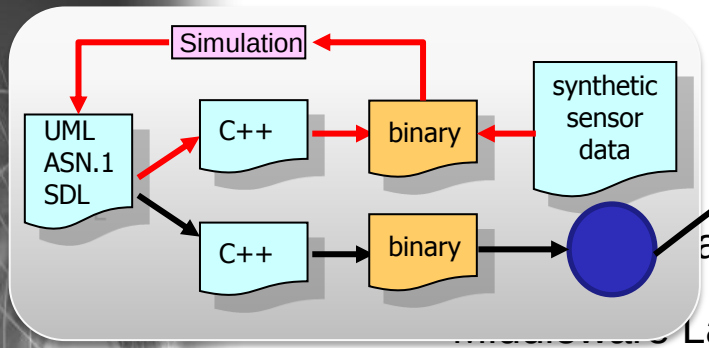
Ausführungsmodell
(ns3--Simulator)

Funktionaler Test, Leistungsbewertung

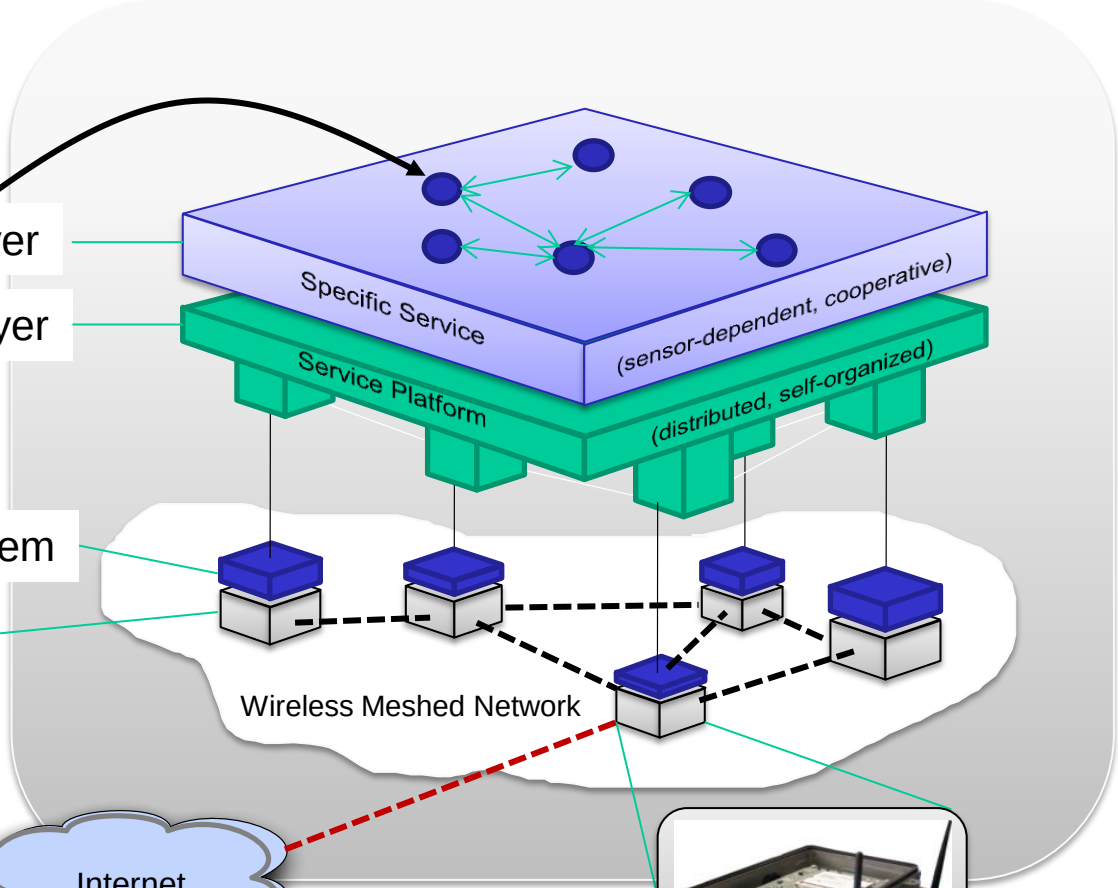
SOSEWIN Überblick

Self-organized Seismic Early Warning Information Network

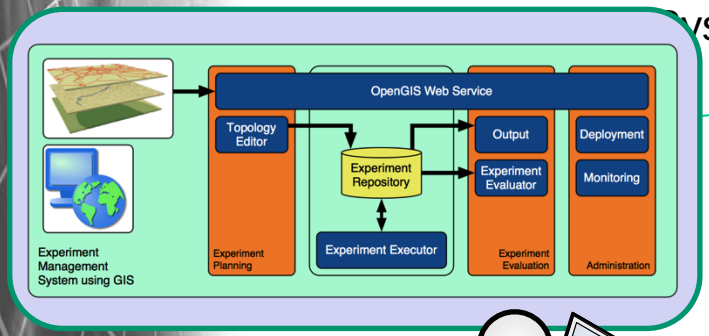
SW Entwicklungstechnologie
(Modelleditor, Simulator, Code -Generator, …)

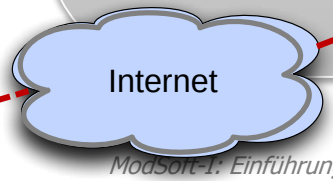


SOSEWIN-HW/SW Architektur



GIS-basierte Netzmanagement- und Experiment-Unterstützung





Netz-Prototyp in Istanbul

HU Berlin

GIS

Ataköy Istanbul

Internet

GFZ Potsdam

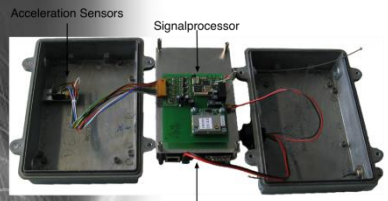
Kandilli Istanbul

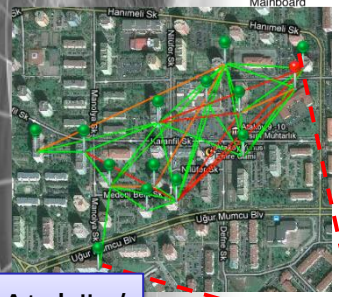
GIS

Istanbul-Infrastruktur-daten

SOSEWIN-Architektur

seit 2008

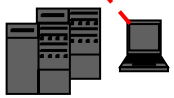




20 Knoten
0.5 km²

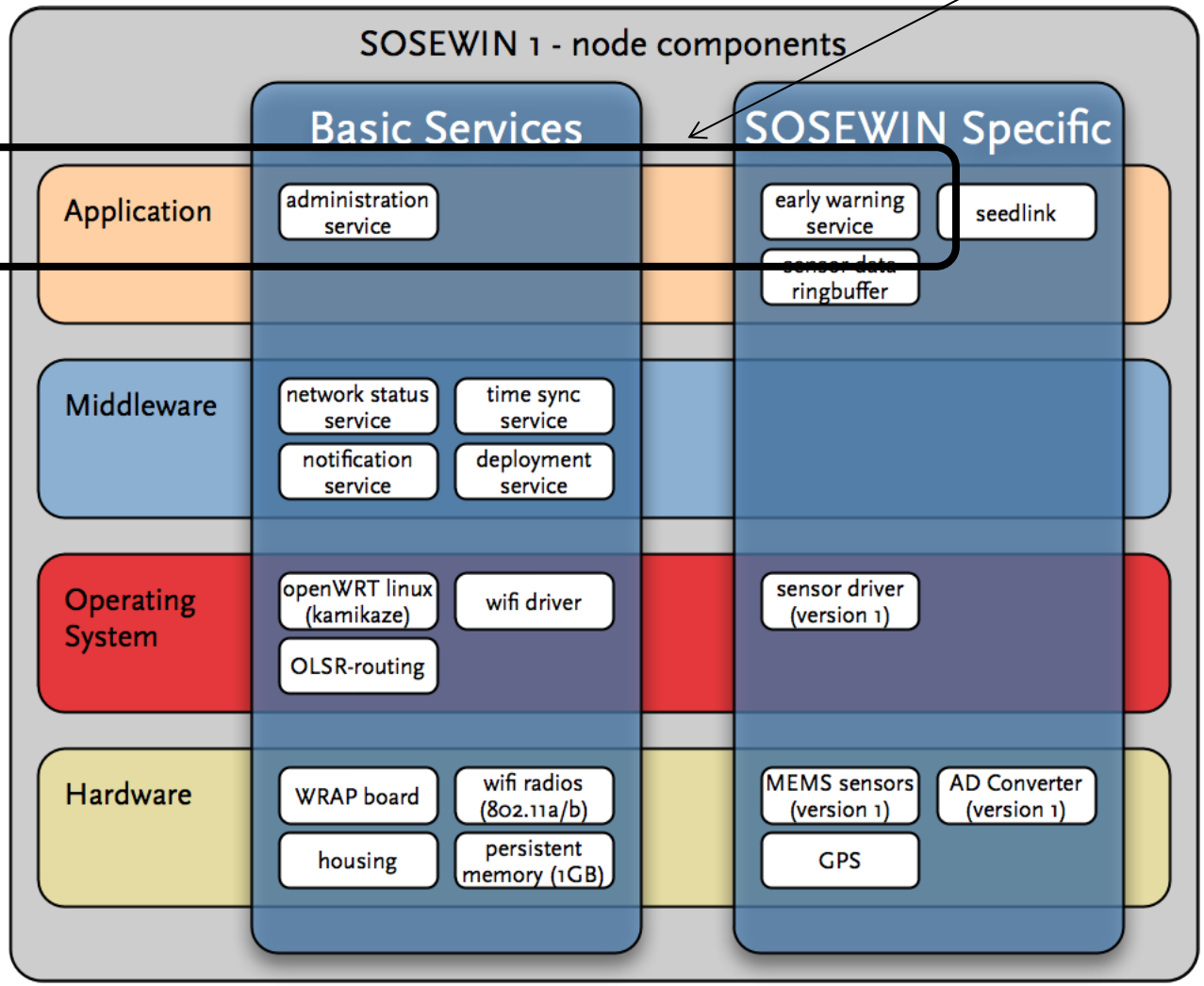
Ataköy/ Istanbul

Internet



Kandilli/ Istanbul
GFZ/ Potsdam
HUB/ Berlin

Achtung: modellgetriebene Entwicklung betraf "nur" …



Teil I: Einführung

1. MDD als Trend in der Software-Entwicklung
2. UML, ein erster Blick
3. Begriffe: Systeme und Modelle
4. Paradigmen der UML-Modellierung
5. Historie von UML
6. Modellierungselemente von UML im Überblick
7. Struktur des UML-Standards

Die Unified Modeling Language (UML)

... kombiniert Konzepte der

- Objektorientierten Modellierung
  (Klasse, Objekt, Beziehungen)
- Datenmodellierung
  (Entity-Relationship-Diagramme)
- Business-Modellierung
  (Work Flows)
- Komponentenmodellierung
  (Ports mit Schnittstellen: required/supported interfaces)
- Verhaltensmodellierung
  (Erweiterte Zustandsautomaten)
- ...

Konzepte der Objektorientierten Modellierung

- als bestimmendes u. durchgehendes Modellierungsparadigma von UML
  - Klassifikation /Exemplifikation
  - Instanz und Klassifizierer
  - Beziehungen zwischen Klassifizierern
    - Spezialisierung / Generalisierung (Wiederverwendung von Modellbausteinen)
    - abstrakte und konkrete Klassifizierer
    - aktive und passive Klassifizierer
  - Instanz als Dreiklang von Identität (Referenz), Struktur und Verhalten
  - Verhalten
    - Ursache-Wirkungsprinzip,
    - zeitliche Abhängigkeit,
    - Parallelität/Nebenläufigkeit,
    - Kommunikation
  - Beziehungen zwischen Instanzen / Instanzmengen
    - Kommunikation
    - Generierung/Terminierung
  - Vielgestaltigkeit von Objekt-Referenzen (Polymorphie)

Abstraktionsmechanismen

Objektorientierte Modellierung

… als Abgrenzung des Paradigmas zu

- funktionsorientiert
  - Funktionalität steht im Mittelpunkt der Modelle
  - strukturierte Entwicklung

- datenorientiert
  - Daten dienen der Modellbildung
  - Entity-Relationship-Modell

- objektorientiert
  - Daten und Funktionen werden als Einheit betrachtet
    Struktur (Zustand) und Verhalten sind
    identifizierbare Entitäten
  - Entwurfsempfehlungen für Software-Engineering: OOA, OOD,

UML-Modellierungsparadigmen (Überblick)

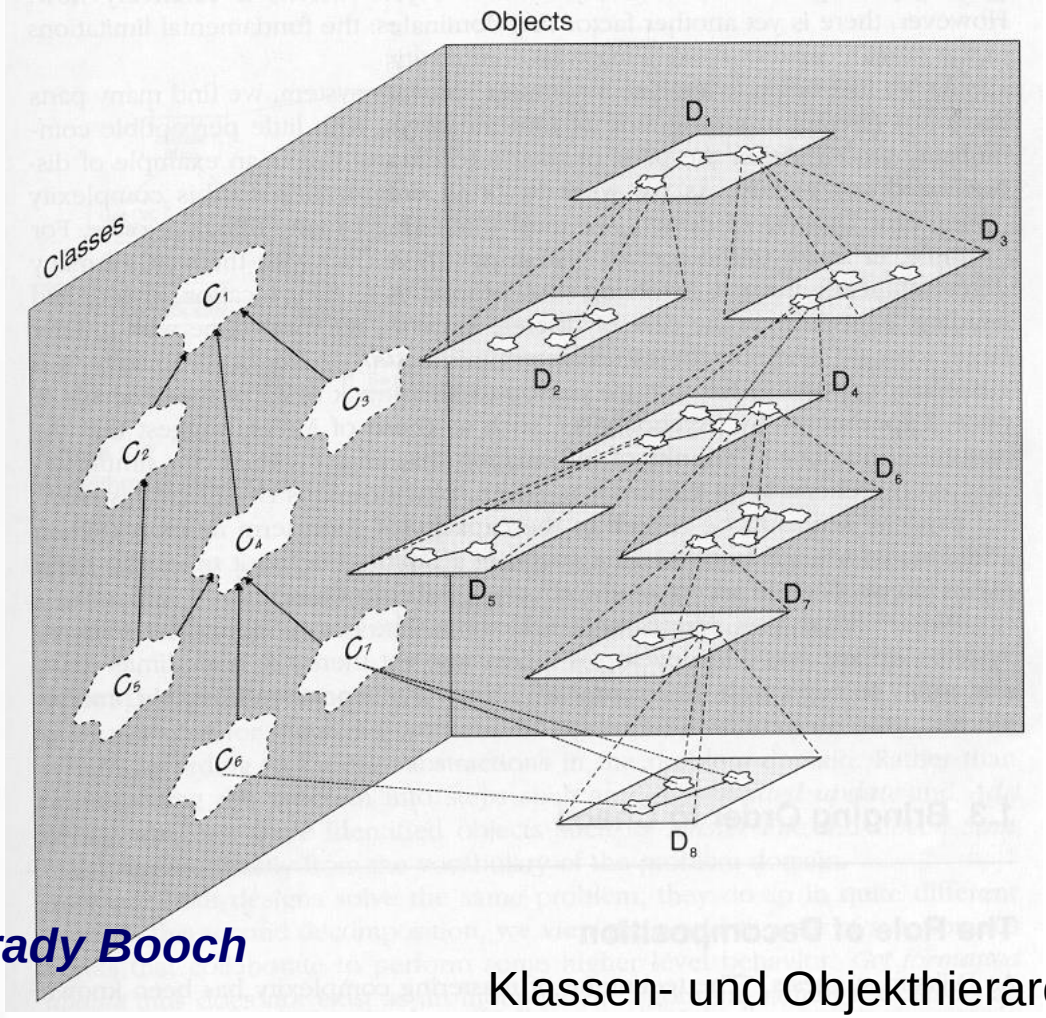
- Vereinheitlichung verschiedener ähnlicher OO-Vorgehensweisen von
  - Rumbaugh,
  - Booch,
  - Jacobsson, ...

  als

  Unified Modelling Language

- zusätzliche allgemeine Paradigmen
  - Modularisierung
  - Komposition/ Dekomposition
  - Erweiterungsmechanismen

Komposition / Dekomposition



Klassifikation

Spezialisierung

Exemplifikation

Komposition

Wiederverwendung

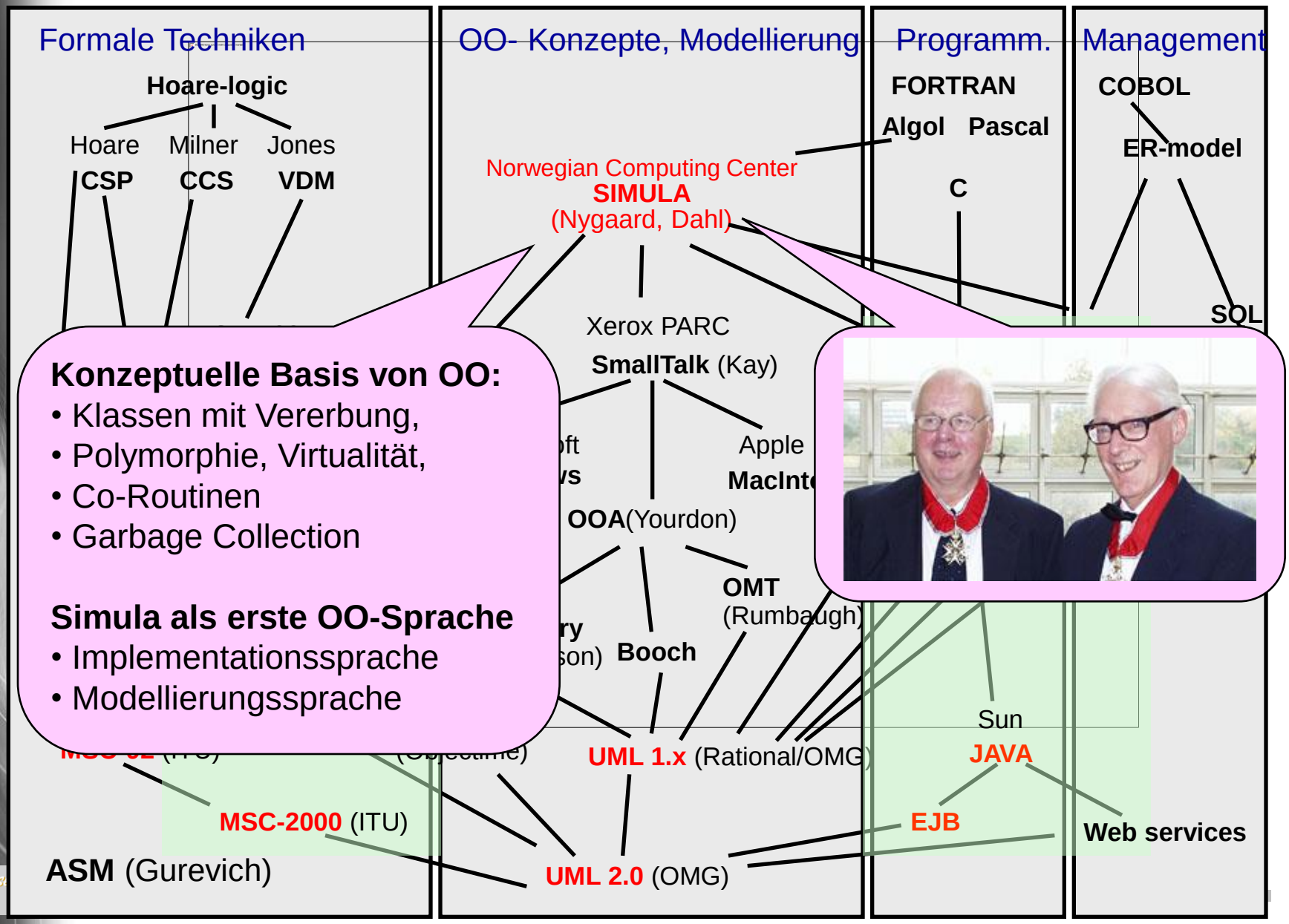
Grady Booch

Klassen- und Objekthierarchien

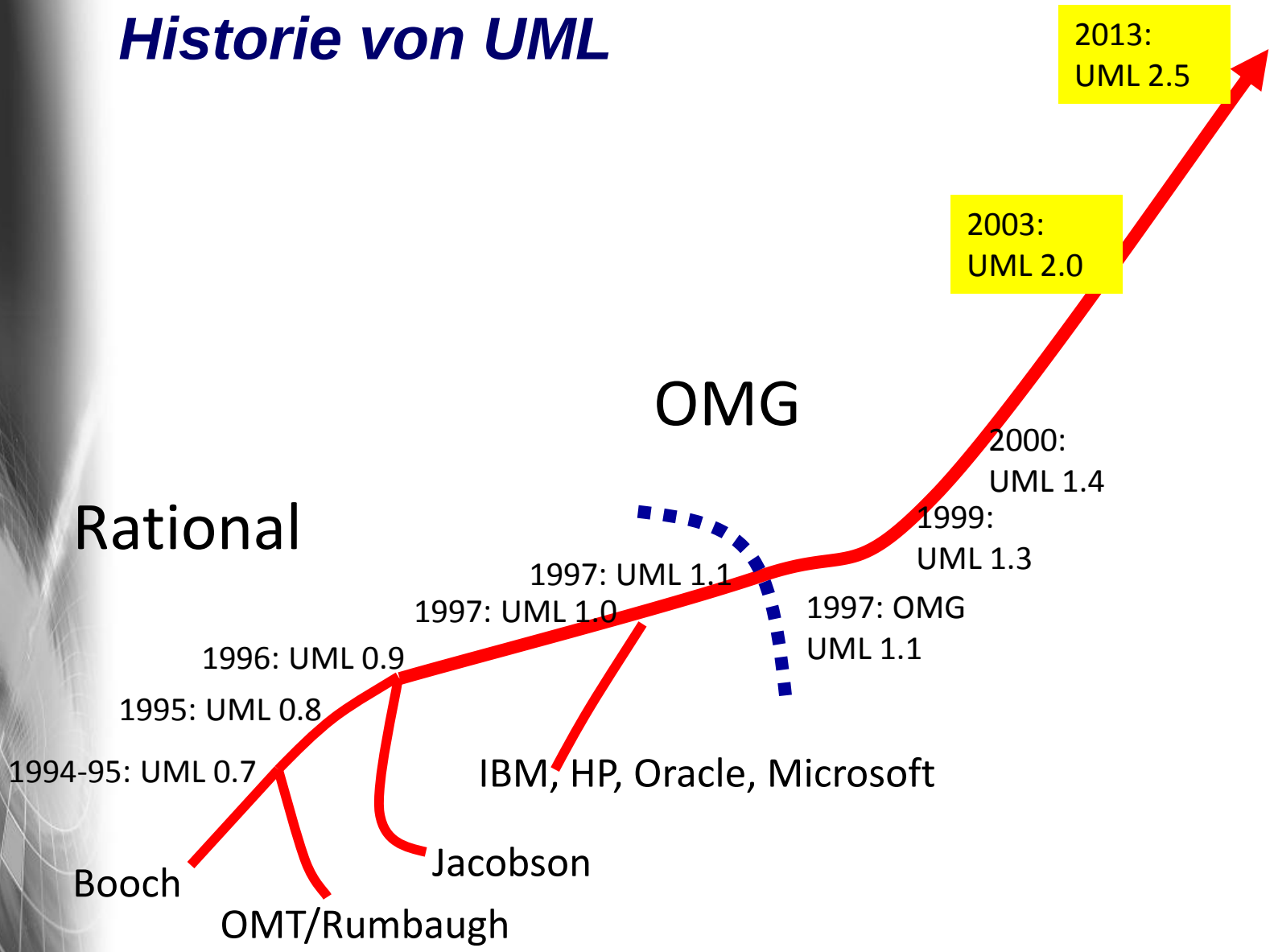
Teil I: Einführung

1. MDD als Trend in der Software-Entwicklung
2. UML, ein erster Blick
3. Begriffe: Systeme und Modelle
4. Paradigmen der UML-Modellierung
5. Historie von UML
6. Modellierungselemente von UML im Überblick
7. Struktur des UML-Standards

Die Ursprünge von UML



Historie von UML



Simula/ Nygaard, Dahl

Teil I: Einführung

1. MDD als Trend in der Software-Entwicklung
2. UML, ein erster Blick
3. Begriffe: Systeme und Modelle
4. Paradigmen der UML-Modellierung
5. Historie von UML
6. Modellierungselemente von UML im Überblick
7. Struktur des UML-Standards

UML-Grundkonzepte

- UML-Modellbausteine
  - Dinge/Enititäten (im Sinne von Abstraktionen) ➔ Klassen, Instanzen
  - Beziehungen zwischen den Entitäten
  - Diagramme zur Anordnung von Entitäten

- Regeln zur Komposition der Modellbausteine

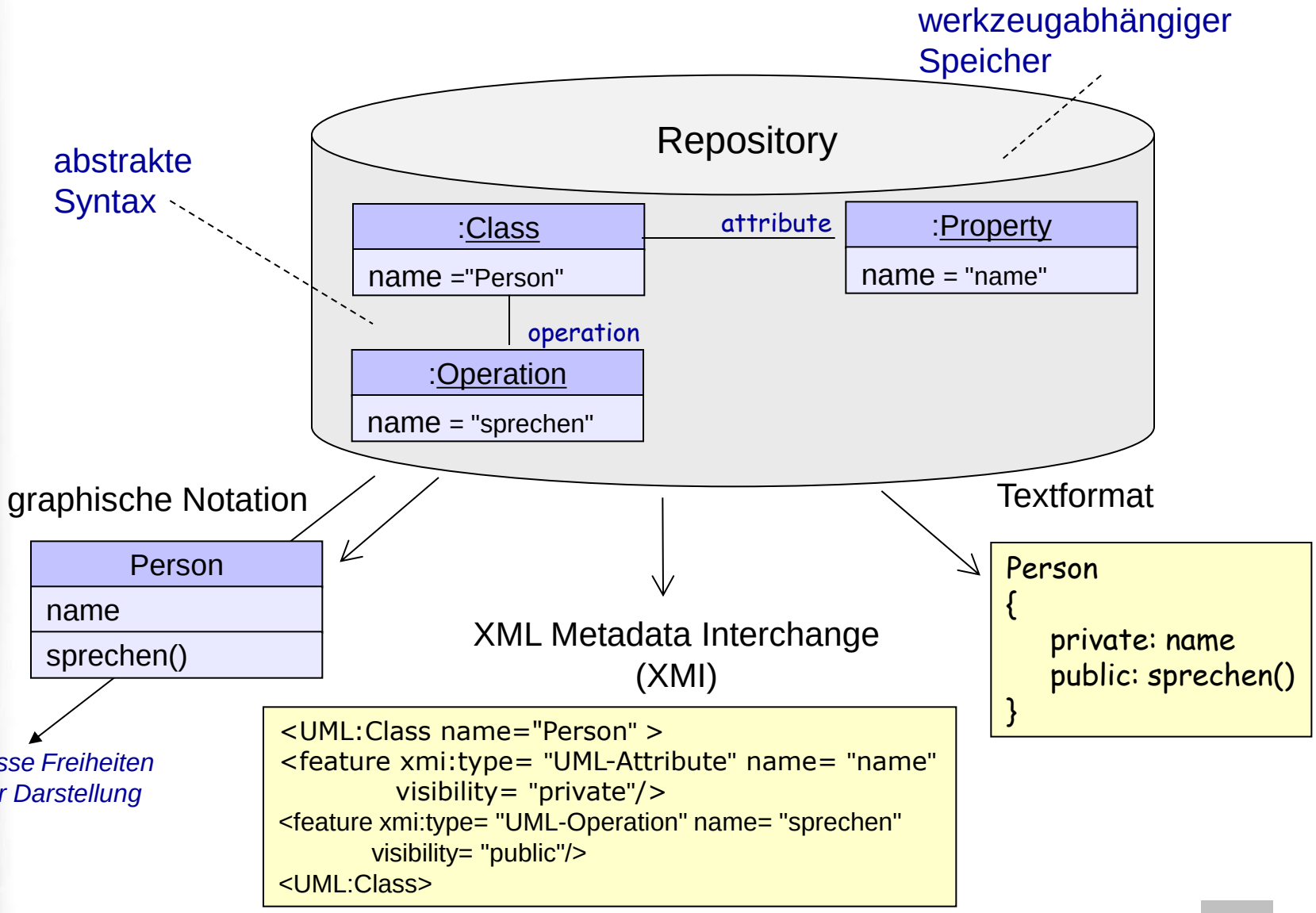
Repräsentation von UML-Elementen

- … erfolgt
  mittels grafischer Symbole (z.B. Rechteck für eine Klasse)

- aber ohne fest genormte Darstellung,
  sie müssen lediglich den Regeln des UML-Metamodells (abstrakte Grammatik) folgen

- alternative Repräsentationen zur graphischen Notation:
  - XML-konforme Strukturen
  - Codedateien
  - abstrakter Syntaxbaum des UML-Metamodells

Illustration: UML-Modell- Repräsentation



Die Statikregeln von UML (well-formed-ness rules)

- sind **Regeln**, die
  - gültige Anordnungen von Entitäten,
  - deren semantische Konsistenz und
  - deren fehlerfreie Funktion

  sichern

- als **syntaktische und semantische Festlegungen** von
  - Namen von Entitäten
  - Gültigkeitsbereichen
    - Kontexte, die Namen bestimmte Bedeutungen zuordnen
  - Sichtbarkeiten
    - wie Namen von anderen Entitäten gesehen und verwendet werden
  - Integritäten
    - korrekte und konsistente Beziehungen der Entitäten
  - Ausführungen
    - genaue Bedeutung der Ausführung/ Simulation eines dynamischen Modells

Die Statikregeln von UML *(Forts.)*

- Insbesondere mit Unterstützung
  evolutionärer Entwicklungen

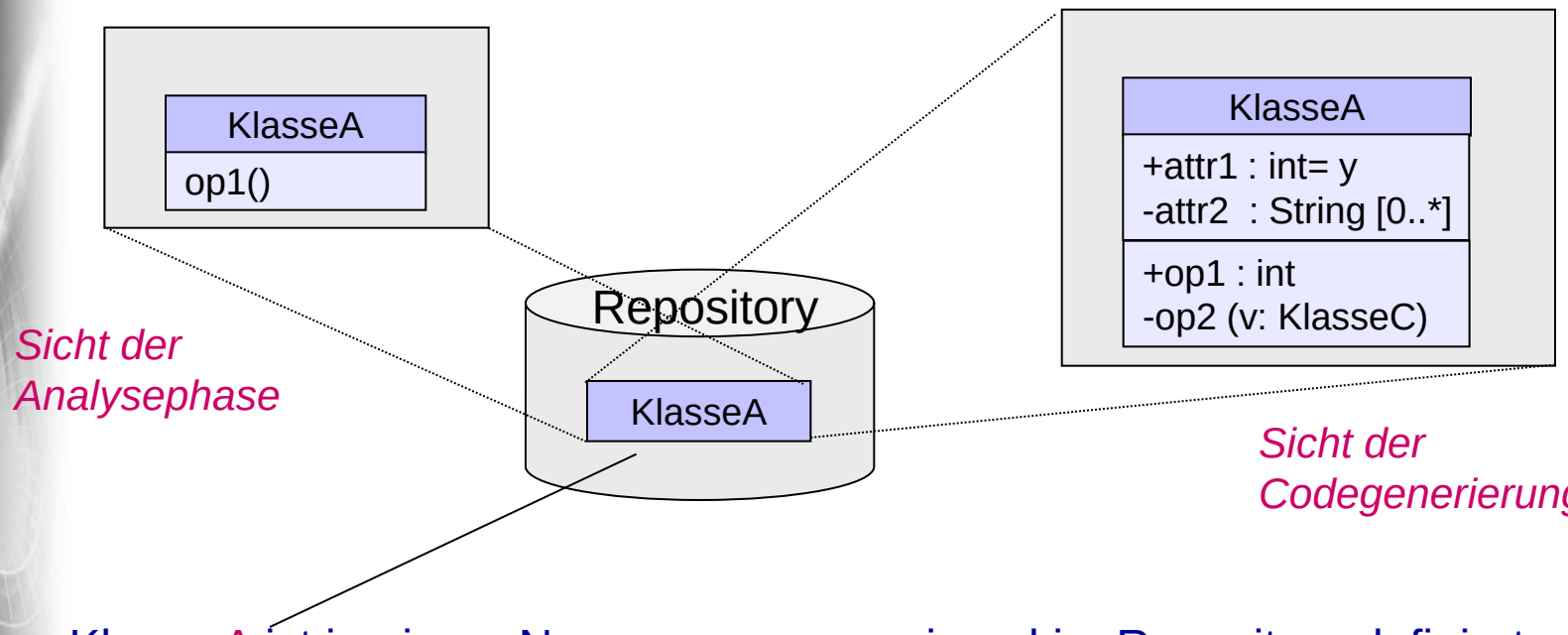
  Erkenntnis: Entwicklung unzulänglicher Modelle ist praktisch unvermeidlich

  - gültige Teildarstellungen
    - zur Vereinfachung sind gewisse Elemente verdeckt/ weggelassen
  - unvollständige Darstellungen
    - gewisse Elemente fehlen
  - Folge: zeitweilige Inkonsistenz
    - klar, dass in solchen Fällen keine Integrität garantiert werden kann

- Achtung:
  zusätzlich weist UML eine Reihe von semantischen **Variationspunkten** auf (z.B. undefiniertes Verhalten)

Sichten

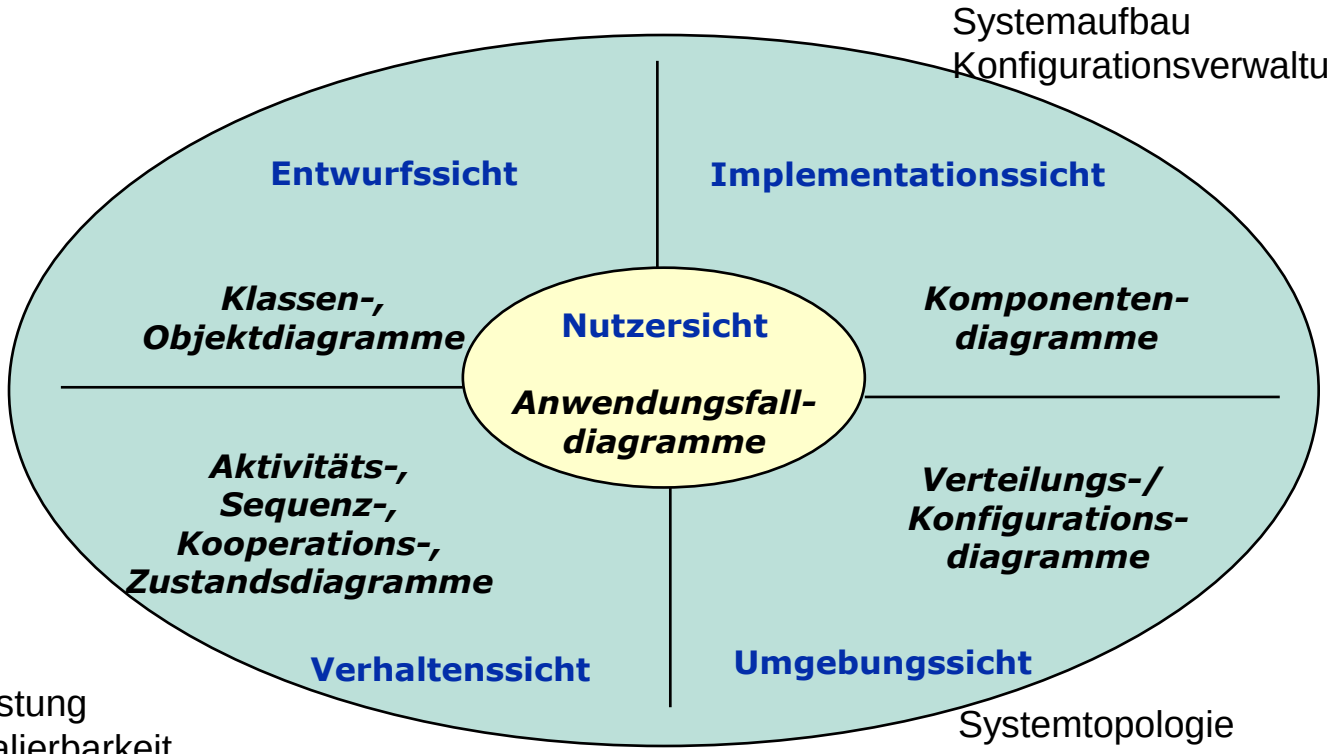
- ein und dasselbe Modellelement
  kann unterschiedlich detailliert für unterschiedliche Sichten dargestellt werden



Klasse A ist in einem Namensraum nur einmal im Repository definiert

Systemarchitekturmodell eines Systems

- … unter Verwendung von UML



Vier Arten von Modellierungselementen/Entitäten

nämlich zur

1. Bildung von Strukturen

2. Beschreibung von Verhalten

3. Bildung von Gruppierungen von Entitäten

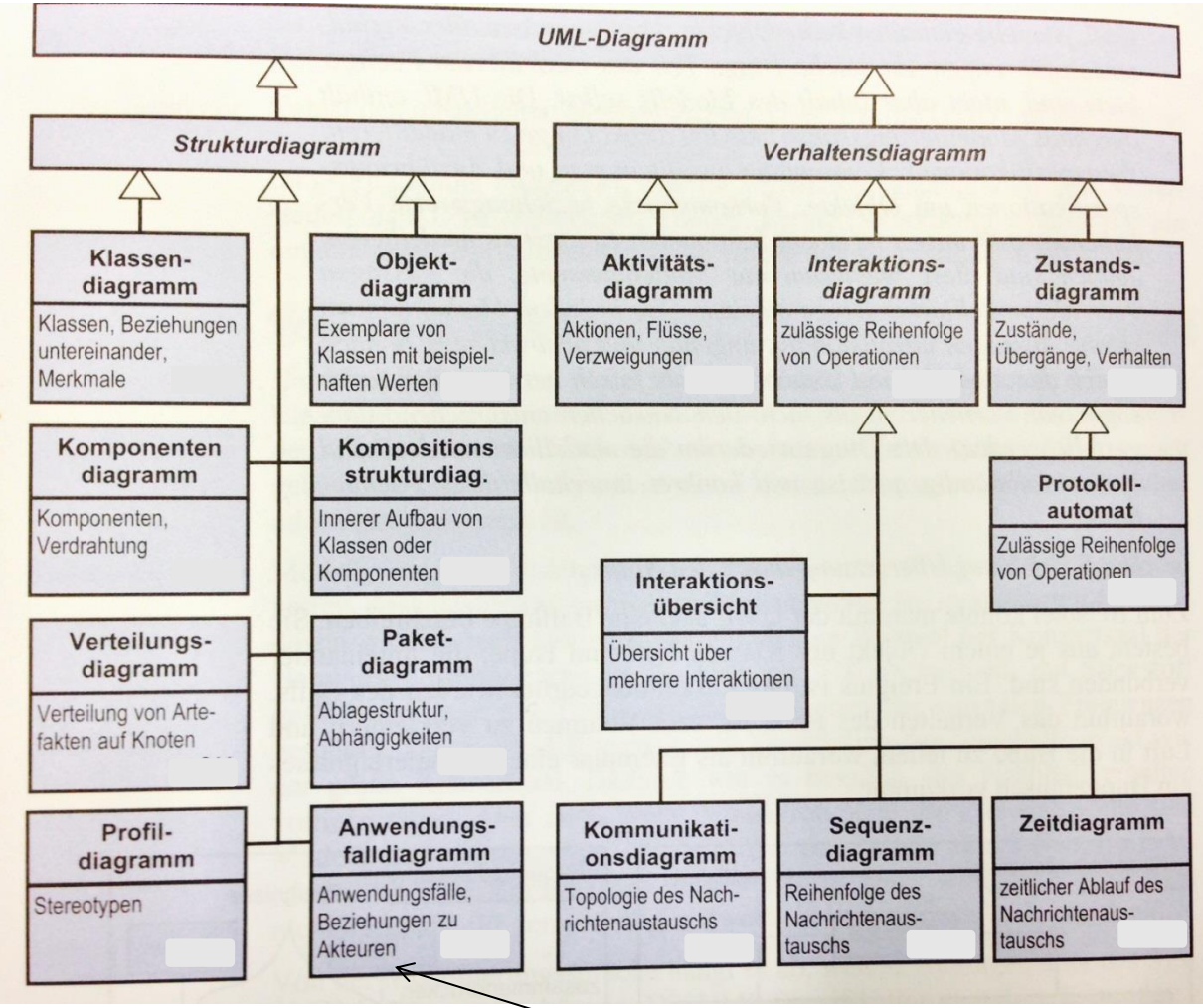
4. Formulierung von Anmerkungen

➔ damit lassen sich UML-Modelle komplett beschreiben

ACHTUNG: Überblick kommt gleich

UML-Diagrammarten

UML erlaubt aber auch beliebigen Mix in einem Diagramm (Tool-abhängig)



ACHTUNG: Abweichung
UML-Standard: Anwendungsfalldiagramm als Verhaltensdiagramm)