Kurs OMSI
im WiSe 2013/14

Objektorientierte Simulation mit ODEMx

Prof. Dr. Joachim Fischer
Dr. Klaus Ahrens
Dipl.-Inf. Ingmar Eveslage

fischer|ahrens|eveslage@informatik.hu-berlin.de

1. Einführung

1. Systemsimulation – was ist das?
2. Ein Blick zurück in die Anfänge
3. Modelle und Originale
4. Modellierungssprachen, Simulationsumgebungen
5. Bespiele aus der aktuellen Forschung
6. Paradigma der objektorientierten Modellierung
7. Einordnung von UML
8. Klassifikation dynamischer Systeme
9. M&S eines Niedertemperaturofens
10. Scheduler für zeitdiskrete und zeitkontinuierliche Systemmodelle

Objektorientierung: Modellierung, Programmierung

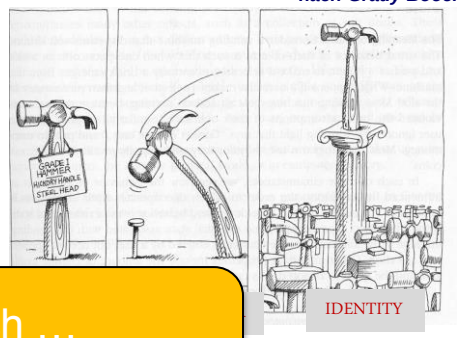
Objektorientiertes Abstraktionskonzepte

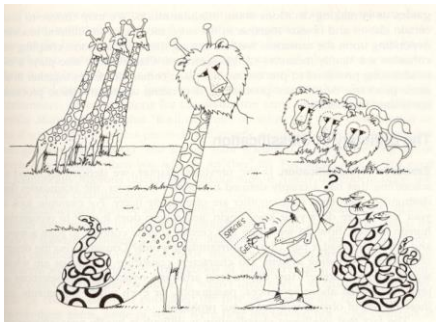
1. eigenverantwortlich handelnde, interagierende Dinge (Objekte)
   - Zustand (Attribute)
   - individuelles Verhalten (Methoden, Dienste)
   - Identität (Referenz)
2. Klassen (Definition von Objekten)
3. Unterscheidung zw.
   - aktiven und
   - passive Klassen
4. Identifikation verschiedenster Beziehungen zwischen Instanzen (bzw. Instanzmengen)
   - Navigierbarkeit
   - Abhängigkeit
   - …
5. Beziehung zwischen Klassen
   - Spezialisierung / Generalisierung
   - abstrakte und konkrete Klassifiziere
6. Polymorphie von (getypten) Referer

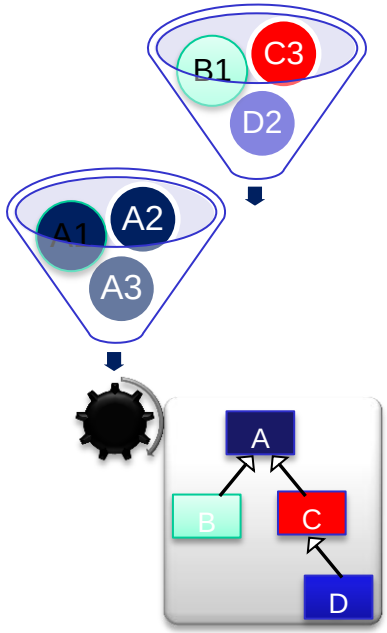
zusätzlich …

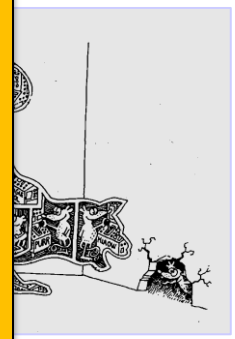
- Gruppierung von Modellelementen
- Komposition/ Dekomposition
- Nebenläufigkeit/ Parallelität/ Synchronisation
- zeitdiskretes/ zeitkontinuierliches Verhalten

nach Grady Booch

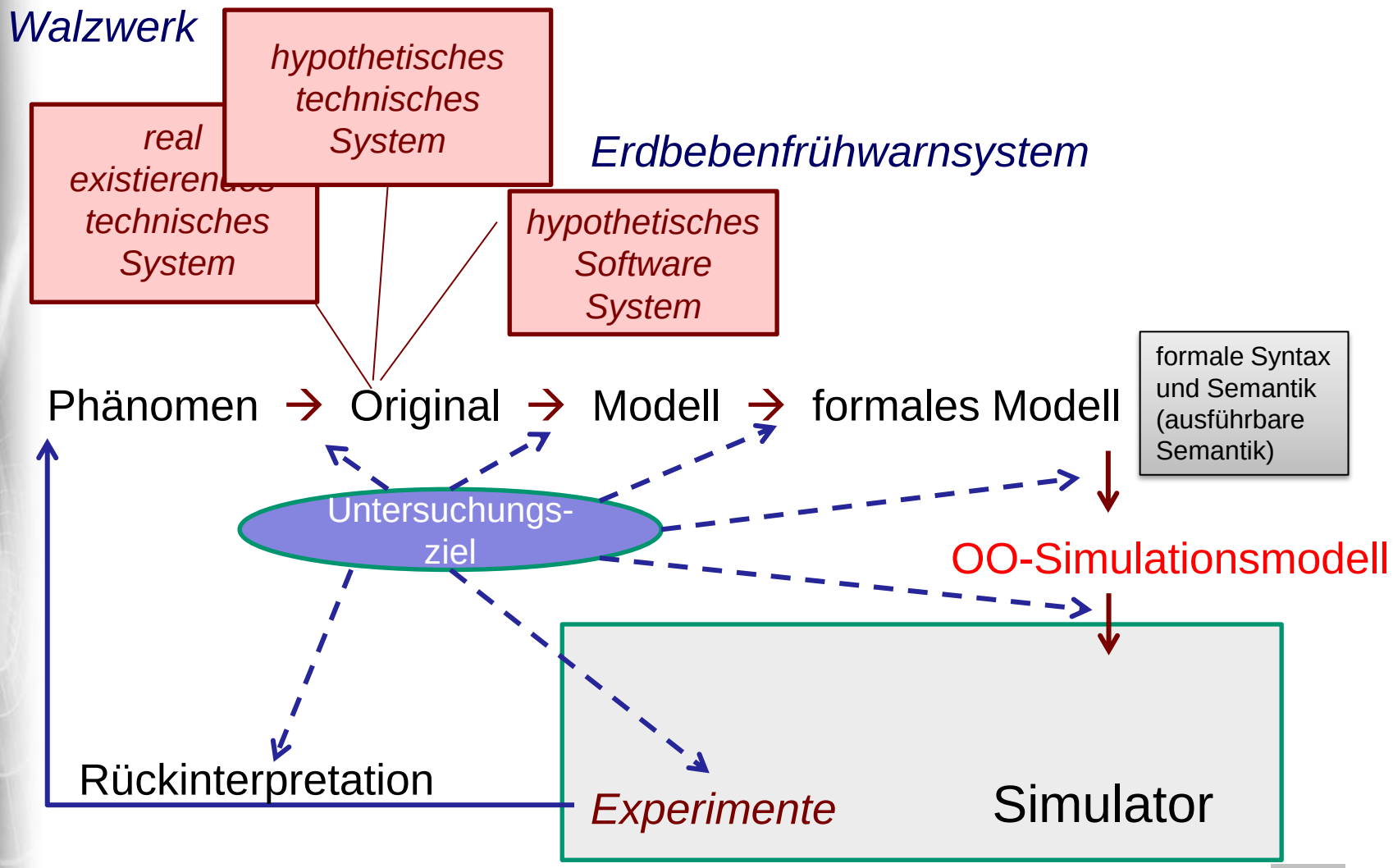








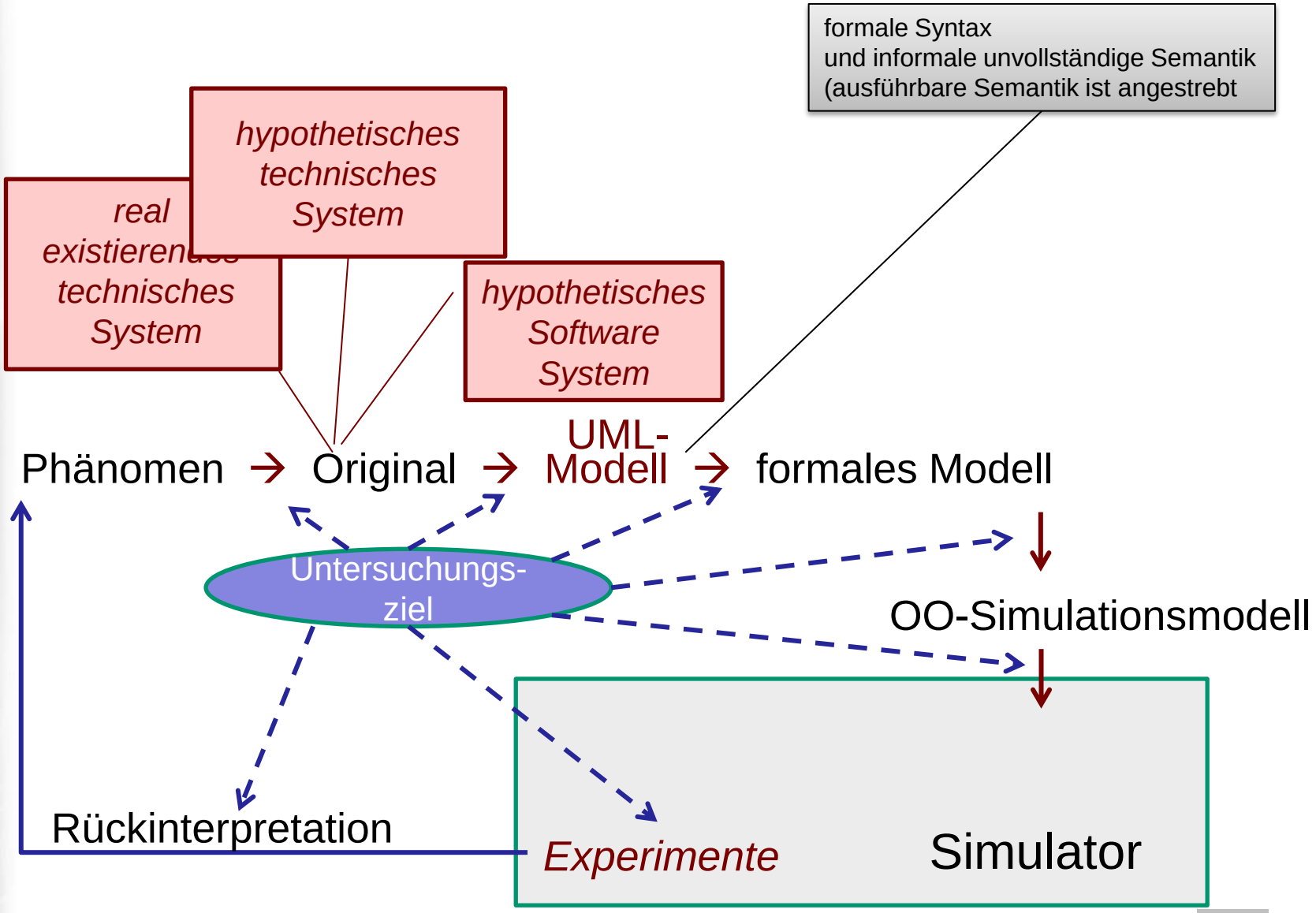
OO-Simulationsmodell



1. Einführung

1. Systemsimulation – was ist das?
2. Ein Blick zurück in die Anfänge
3. Modelle und Originale
4. Modellierungssprachen, Simulationsumgebungen
5. Bespiele aus der aktuellen Forschung
6. Paradigma der objektorientierten Modellierung
7. Einordnung von UML
8. Klassifikation dynamischer Systeme
9. M&S eines Niedertemperaturofens
10. Scheduler für zeitdiskrete und zeitkontinuierliche Systemmodelle

Objektorientierung: Einordnung von UML



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

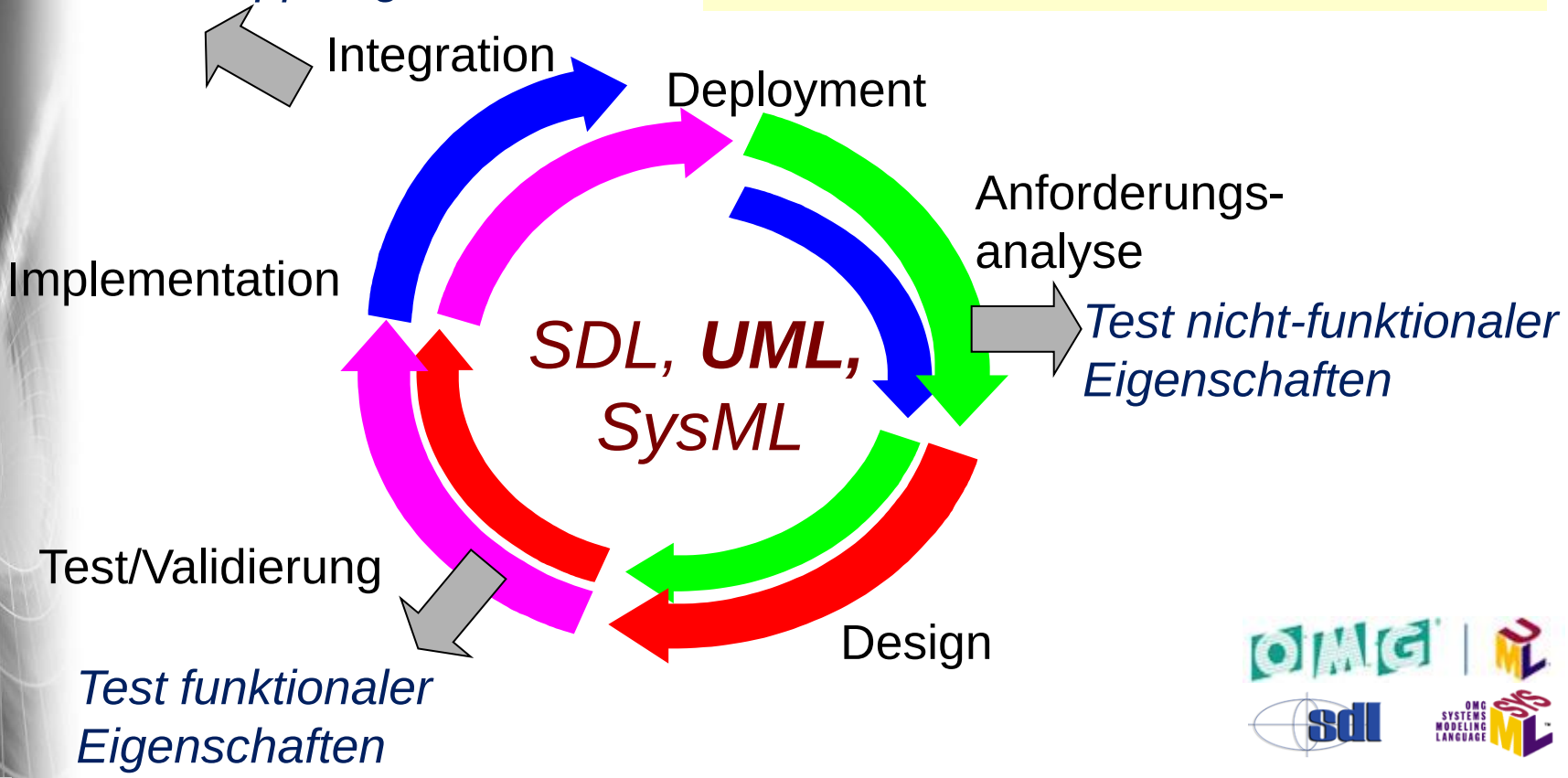
UML-Charakterisierung

- Modellierungssprache mit informal definierter **dynamischer Semantik** (statische Semantik ist mit UML und OCL definiert)
- Dynamische Semantik, enthält sog. semantische Variationspunkte (z.T. alternative semantische oder offene semantische Festlegungen)
- → Problem für Compiler (Zielcode, Simulatorcode)
- **Action-Sprache** von UML befindet sich in Entwicklung (Einfluss auf Datentypen in UML)
  Referenzsemantik ← →Wertesemantik ?
  - SDL-ASN.1-UML-Compiler (HU Berlin) benutzt C als Action-Sprache
  - OMSI-Vorlesung benutzt C++ als Actionsprache (Magic-Draw als Tool)

SW-Entwicklungsprozess: *spiralförmig, inkrementell & iterativ*

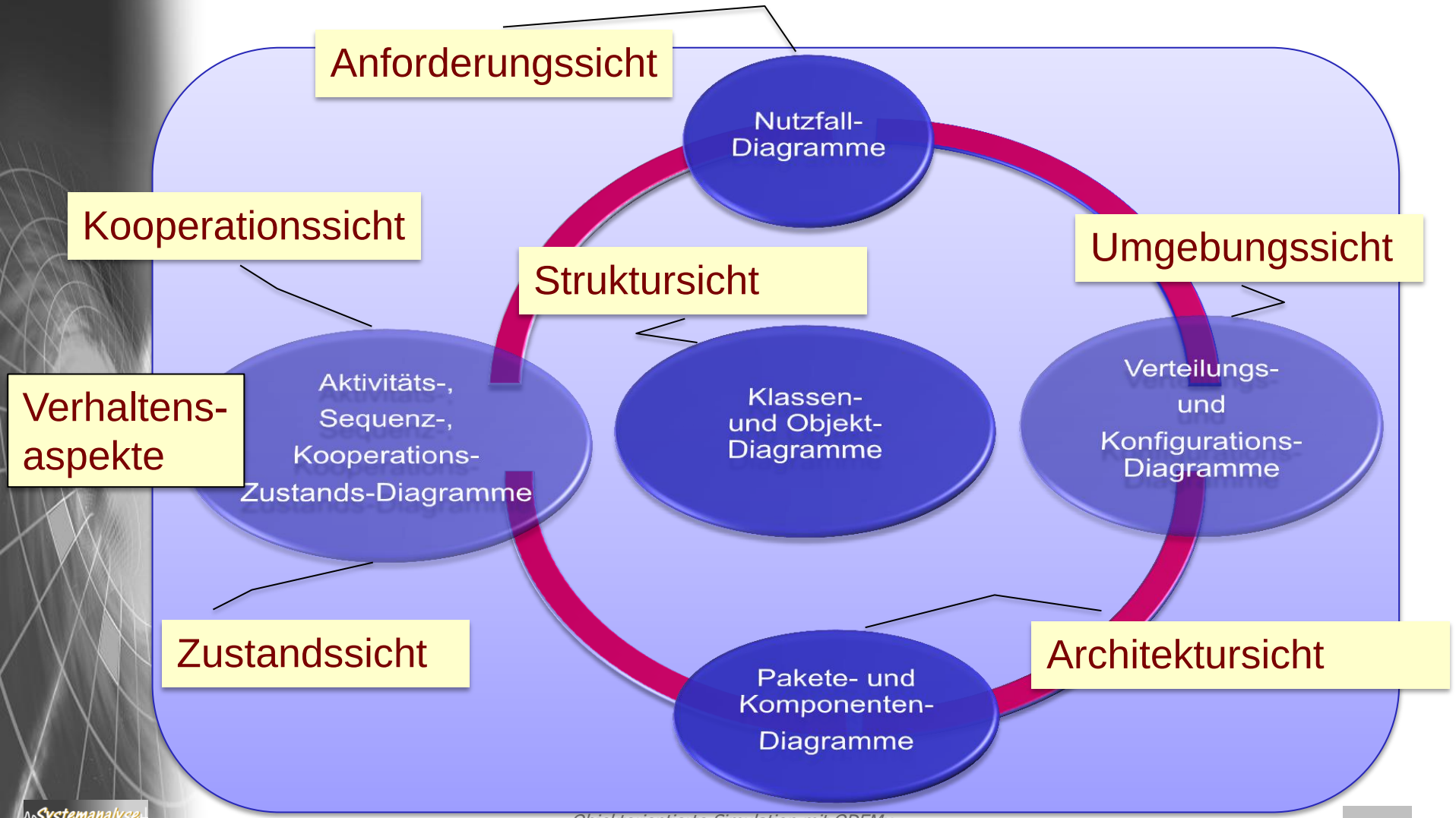
MDD:= Model Driven Development
- SW-Entwicklung ist modellzentriert (Modelle begleiten ges. SW-Lebenszyklus)
- automatische Transformationen für Modellübergänge
- spezifische Analysen (Checker, Simulatoren, …)
- partielle oder komplette Codegenerierung

Test funktionaler und nicht-funktionaler Rückkkopplungen

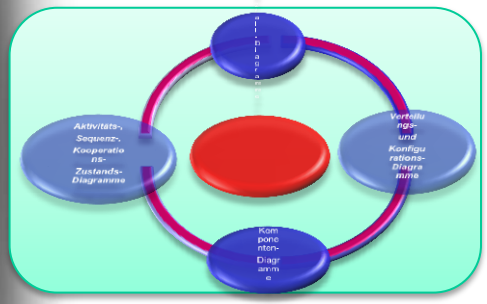


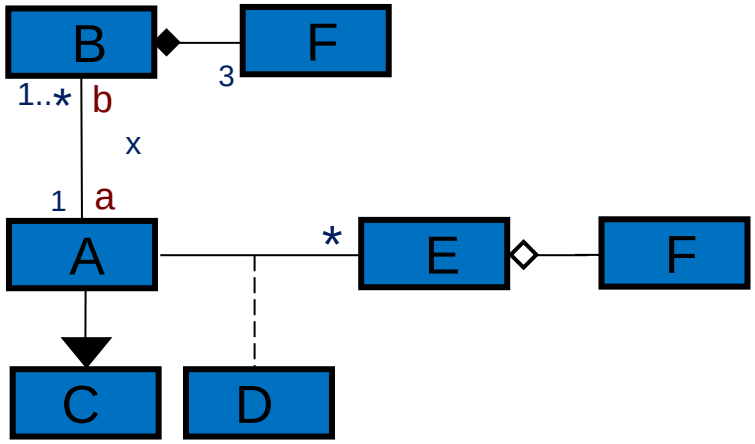
Diagrammarten ~ UML-Teilsprachen

Modellklassen zur Beschreibung von dynamischen Systemen



Klassendiagramme





- A steht mit B in einer Beziehung namens x
  jedem A-Objekt sind 1 bis beliebig viele B-Objekte zugeordnet,
  jedem B-Objekt ist genau ein A-Objekt zugeordnet
- A ist eine Spezialisierung von C
- D ist eine Assoziationsklasse, die die Beziehung zwischen A- und E-Objekten beschreibt
  jeder A-E-Link ist durch ein D-Objekt charakterisiert,
  dieses hat zwei Attribute: A-Referenz, E-Referenz
- Jedes E-Objekt besitzt ein F-Objekt (Agrregation)
  temporäreTeil-Ganzes-Beziehung
- Jedes B-Objekt besitzt 3 F-Objekte als (Komposition)
  permanente Teil-Ganzes-Beziehung

Klassen und Objekte

passive Klasse

aktive Klasse

Kunde

Kunde

name
titel
geburtstag
geschlecht

Kunde

name: **String**
titel: **String**
geburtstag: **Datum**
geschlecht: **Geschlecht**

alter(): **Integer**

Ticket
{abstract}

KommEinheit

partner: **KommEinheit**

«**signal**»
SigA(**integer**),
SigB(**float**)

op (**int**, **char**)

- beliebige **Detailierungsgrade** in der Darstellung einer Klasse möglich
  Tools sichern, dass in einem Namensraum keine Widersprüche zwischen Darstellungen einer Klasse auftreten

- Sichtbarkeit kann individuell eingeschränkt werden
  + public, # protected, - private, ~ package

Attribut-Belegung (Zustand zu einem Zeitpunkt) des Kunden-Objektes K1

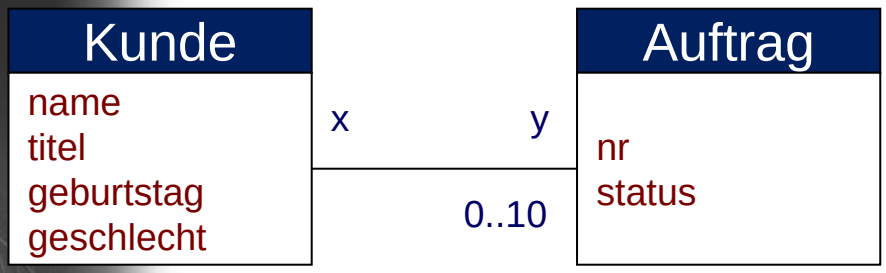
K1:Kunde

name = “Mustermann”
titel = “Dipl-Inf”
geburtstag= 10-09-1992
geschlecht= weiblich

:Kunde

anonymes Objekt

Dualität von Attributen und Assoziationsenden



Lesart:
jedem Kunde-Objekt ist eine Kollektion y von Aufträgen zugeordnet

jedem Auftrag-Objekt ist genau ein Kunde-Objekt x zugeordnet

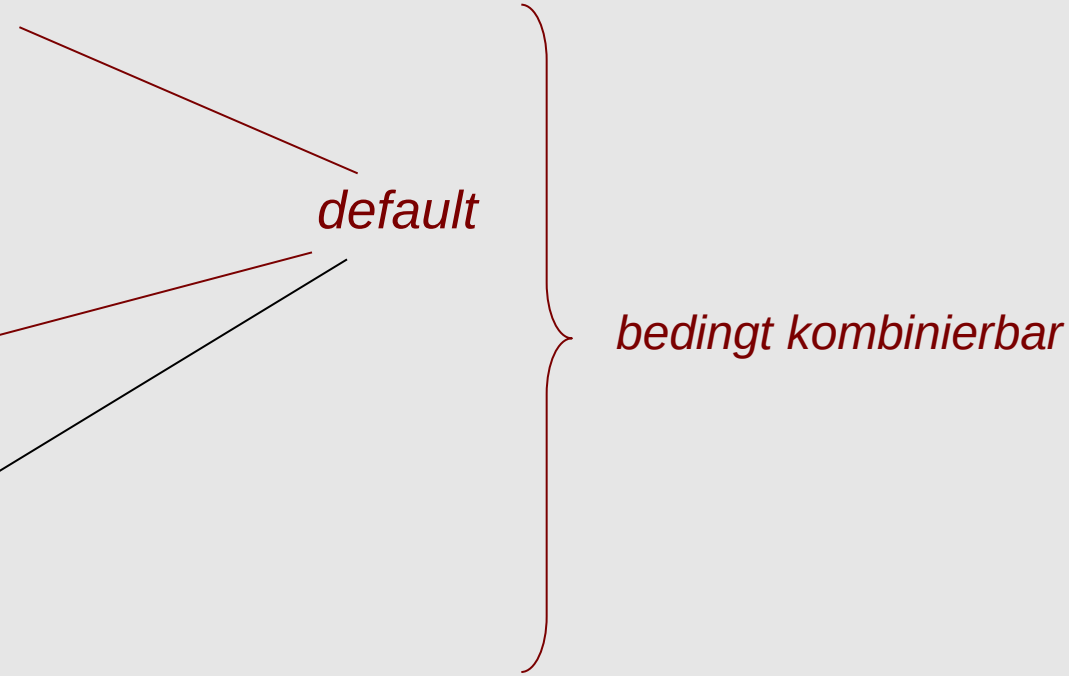
- *Das Assoziationsende x (vom Typ Kunde) dient der Navigation und entspricht einem Attribut der Klasse Auftrag*
- *Das Assoziationsende y (vom Typ Menge über Auftrag) entspricht einem Attribut von Kunde*

Kardinalität als Eigenschaften von Attributen/Assoziationsenden

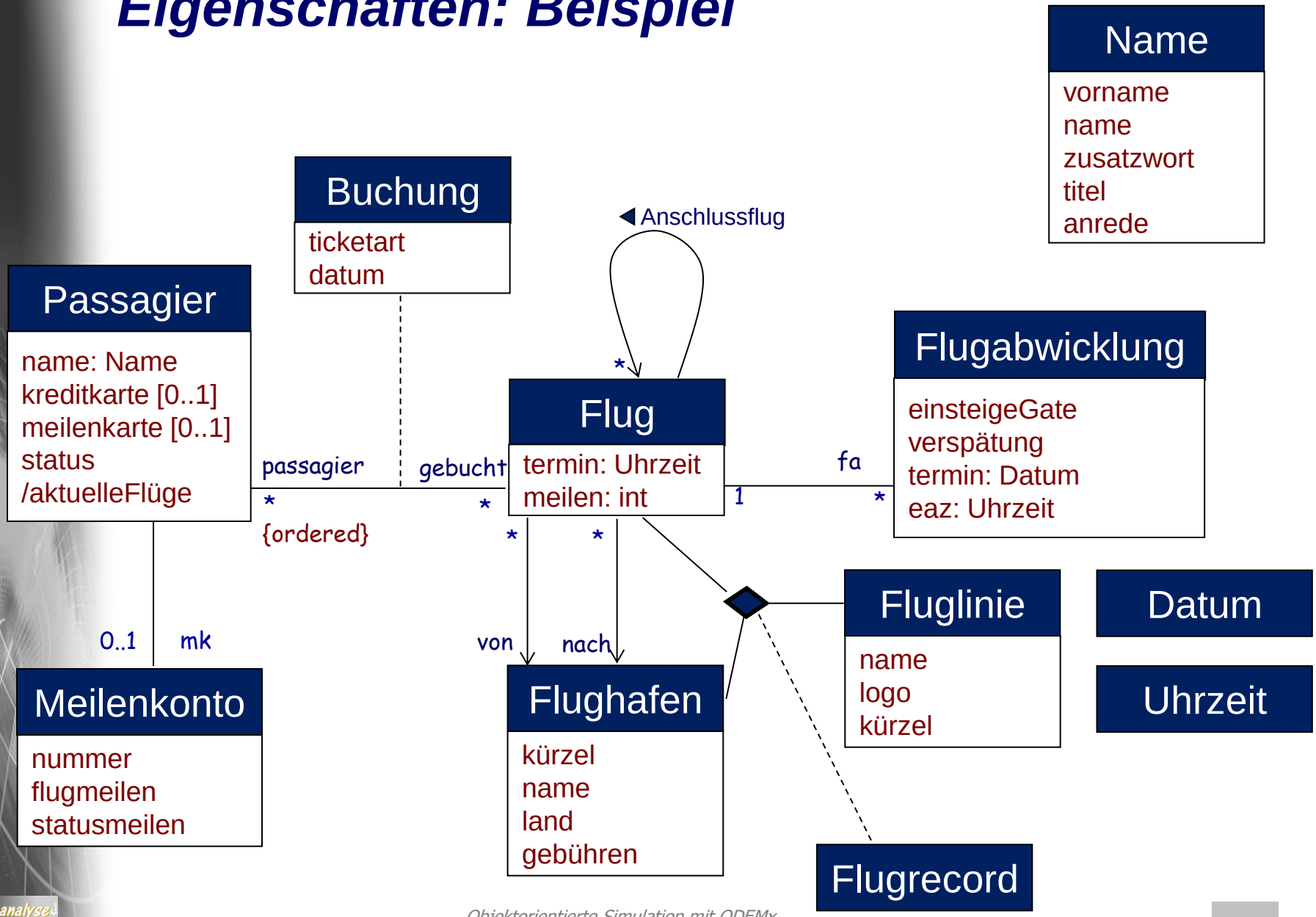
falls Assoziationsenden nicht benannt worden sind

Liste möglicher Eigenschaften

- *… für Assoziationsenden (auch für Attribute) in {…} als Constraint*
  - / (abgeleitete Assoziation)
  - readonly ← → unrestricted
  - composite
  - redefines Attr
  - subsets Attr
  - union
  - unique ← → nonunique
  - ordered ← → unordered
  - seq (Vor.: Multiplizität > 1)
  - bag (Vor.: Multiplizität > 1)



Eigenschaften: Beispiel



Erweiterter Endlicher Zustandsautomat

Endlicher Zustandsautomat
- endliche Anzahl von Grundzuständen
- ein Startzustand
- Zustandsübergang mit möglichen Aktionen

A) Erweiterung eines Endlichen Zustandsautomaten
- Variablen
- Timer
- Pool von Nachrichten (asynchrone Komm.)
- Trigger (Zustandsübergänge)
- Nachrichten/Remote-Op-Rufe
- Guards (als Bedingungen über Zustandsgrößen
- Timer

B) Zustandserweiterung
- Eintrittsaktionen
- Do-Aktivität
- Austrittsaktionen

C) Zustand als Classifier
(Vererbung, Instanziierung, …)

Im Kontext einer aktiven Klasse

Klassen – Aktivitäten - Zustandsmaschinen