

Kurs OMSI im WiSe 2012/13

Objektorientierte Simulation mit ODEMx

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

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

Letzte Vorlesung

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 Klassifizierung
6. **Polymorphie** von (getypten) Referenzen

zusätzlich ...

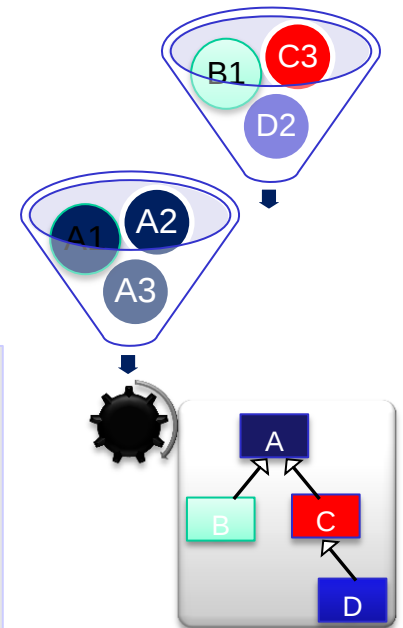
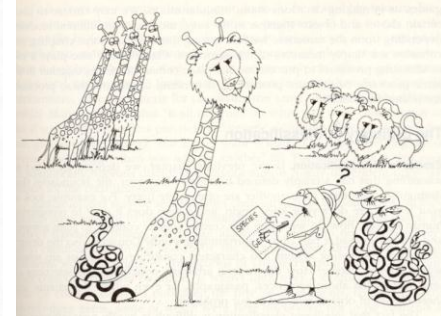
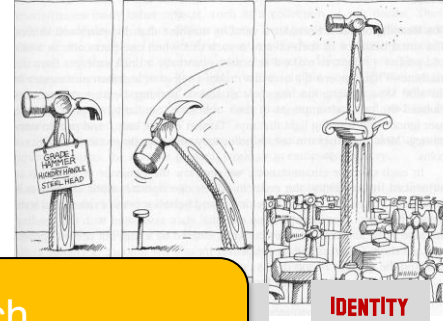
Gruppierung von Modellelementen

Komposition/ Dekomposition

Nebenläufigkeit/ Parallelität/ Synchronisation

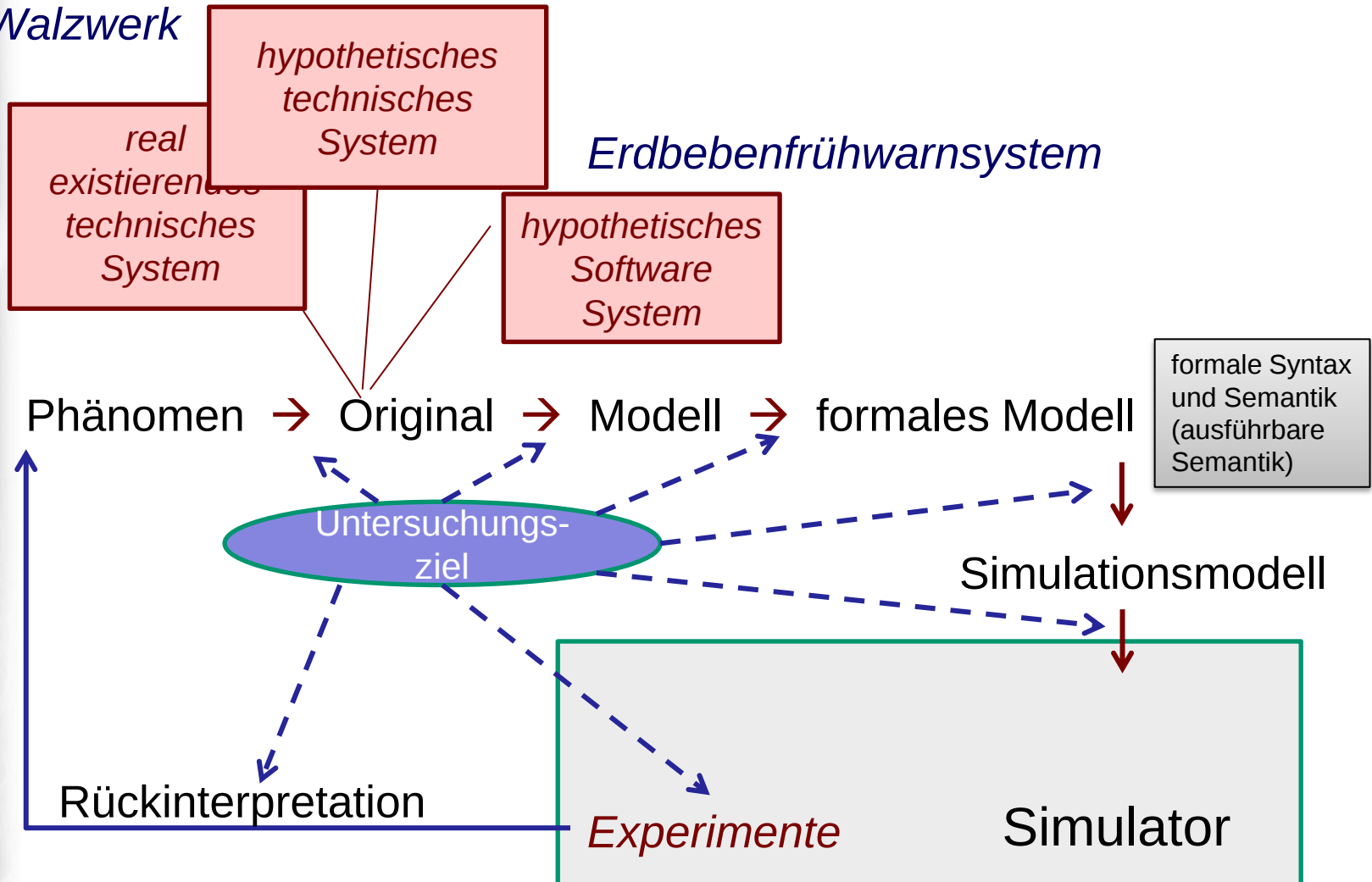
zeitdiskretes/
zeitkontinuierliches
Verhalten

nach Grady Booch



Konzepte, Beispiele

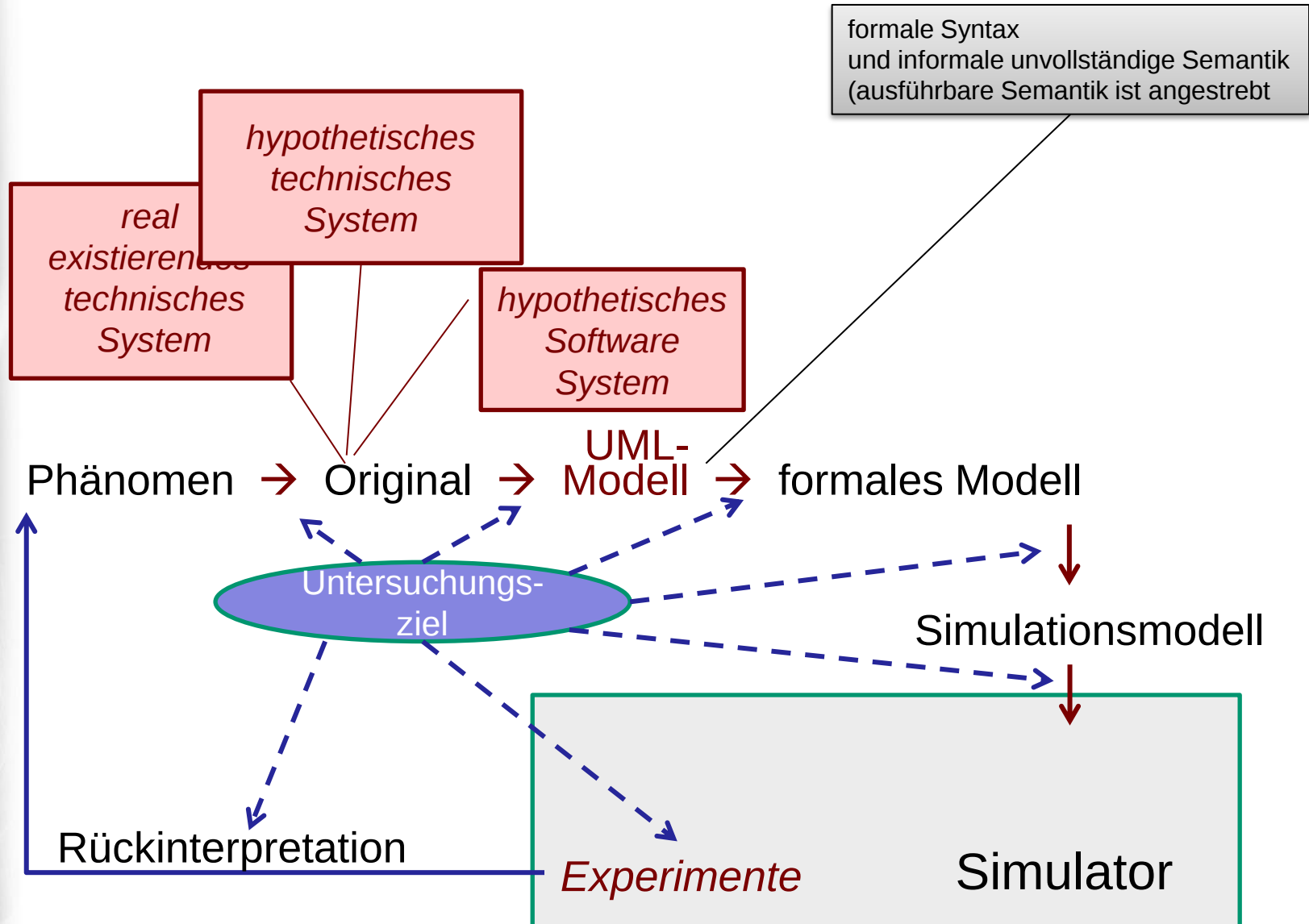
Walzwerk



1. Einführung

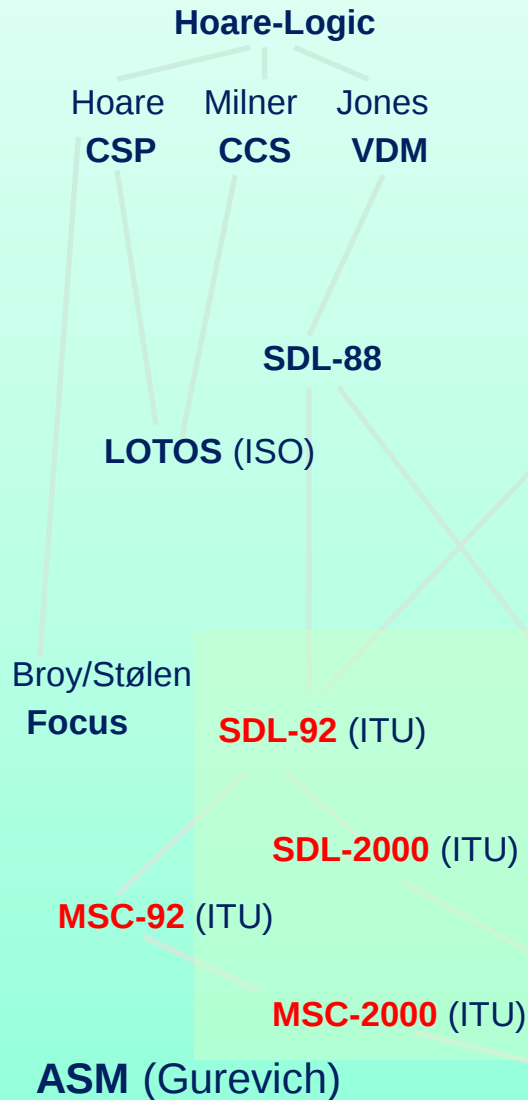
1. Systemsimulation – was ist das?
2. Ein Blick zurück in die Anfänge
3. Modelle und Originale
4. Modellierungssprachen, Simulationsumgebungen
5. Beispiele aus der aktuellen Forschung
6. Paradigma der objektorientierten Modellierung
7. **Einordnung von UML**
8. Klassifikation dynamischer Systeme
9. M&S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

Objektorientierung: Einordnung von UML

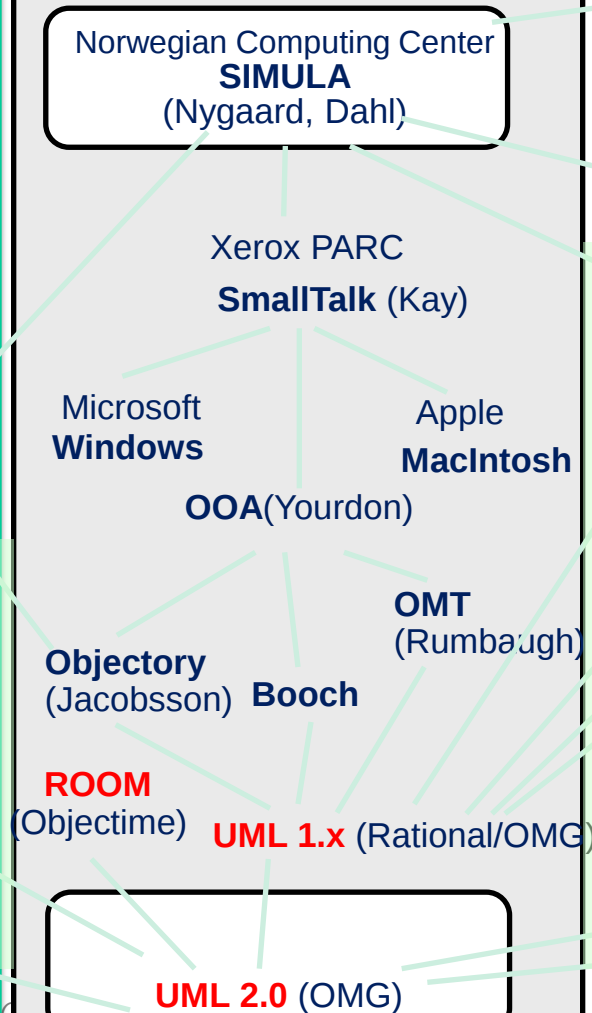


Wurzeln der Objektorientierten Modellierung/ Programmierung

Formale Techniken



OO- Konzepte, Modellierung



Programm.

FORTRAN
Algol **Pascal**

C
Bell Labs
C++

Sun
JAVA

EJB

Management

COBOL
ER-model

SQL
OODB

Corba

Web services

Wurzeln der Objektorientierten Modellierung/ Programmierung

Formale Techniken

Hoare-Logic

Hoare
CSP

Milner
CCS

Jones
VDM

OO- Konzepte, Modellierung

Norwegian Computing Center
SIMULA
(Nygaard, Dahl)

Programm.

FORTRAN

Algol Pascal

C

Management

COBOL

ER-model

Konzeptuelle Basis von OO:

- Klassen mit Vererbung,
- Objektreferenzen,
- Polymorphie, Virtualität,
- Co-Routinen,
- Garbage Collection
- Standardklassenbibliothek

Simula als erste OO-Sprache

- Implementationssprache
- Modellierungssprache
- Byte-Code

Xerox PARC
SmallTalk (Kay)

Apple
MacInt

OOA(Yourdon)

OMT
(Rumbaugh)

Booch

UML 1.x (Rational/OMG)

Sun
JAVA

EJB

Web services

MSC-2000 (ITU)

ASM (Gurevich)

UML 2.0 (OMG)



Wurzeln der Objektorientierten Modellierung/ Programmierung

Formale Techniken

Hoare-Logic

Hoare **CSP** Milner **CCS** Jones **VDM**

Konzeptuelle Basis von OO:

- Klassen mit Vererbung,
- Polymorphie, Virtualität,
- Co-Routinen
- Garbage Collection

OO- Konzepte, Modellierung

Norwegian Computing Center
SIMULA
(Nygaard, Dahl)

Xerox PARC
SmallTalk (Kay)

Apple
Macintosh
OOA(Yourdon)

Objectory
(Jacobsson) **Booch**

ROOM
(Objectime) **UML 1.x** (Rational/OMG)

UML 2.0 (OMG)

Programm.

effiziente Programmausführung
(explizite Speicherverwaltung)
C with Classes

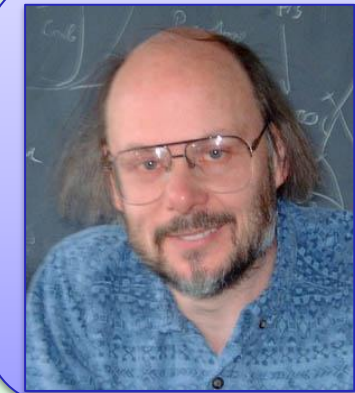
Management

OODB

SQL

Corba

Bell Labs
++



EJB

Web services

Broy/Stølen
Focus

SDL-92 (ITU)

SDL-2000 (ITU)

MSC-92 (ITU)

MSC-2000 (ITU)

ASM (Gurevich)

Die UML



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



551 v. Chr. bis 479 v. Chr.

- 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

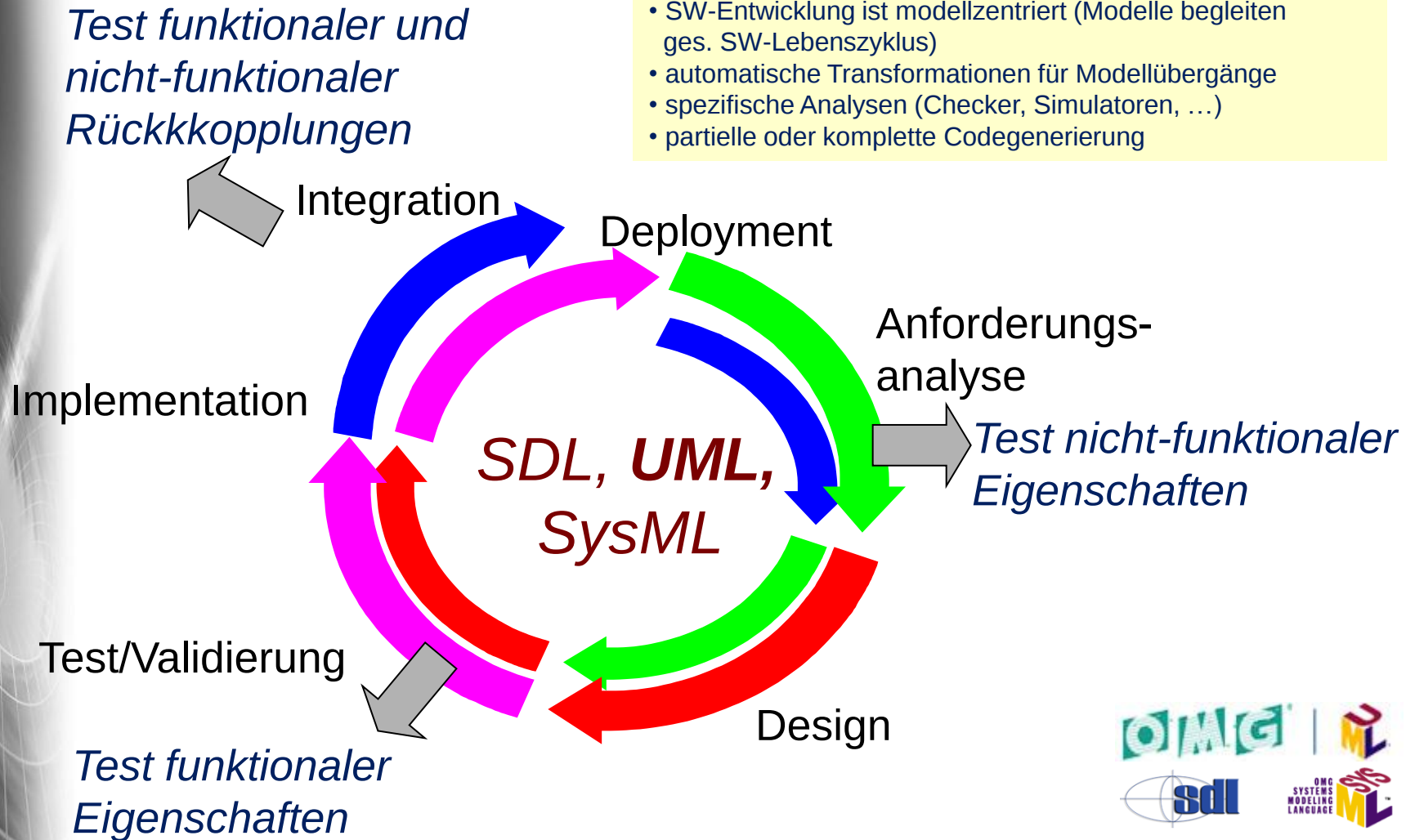
UML-Charakterisierung

- 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 $\leftarrow$ $\rightarrow$ 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*

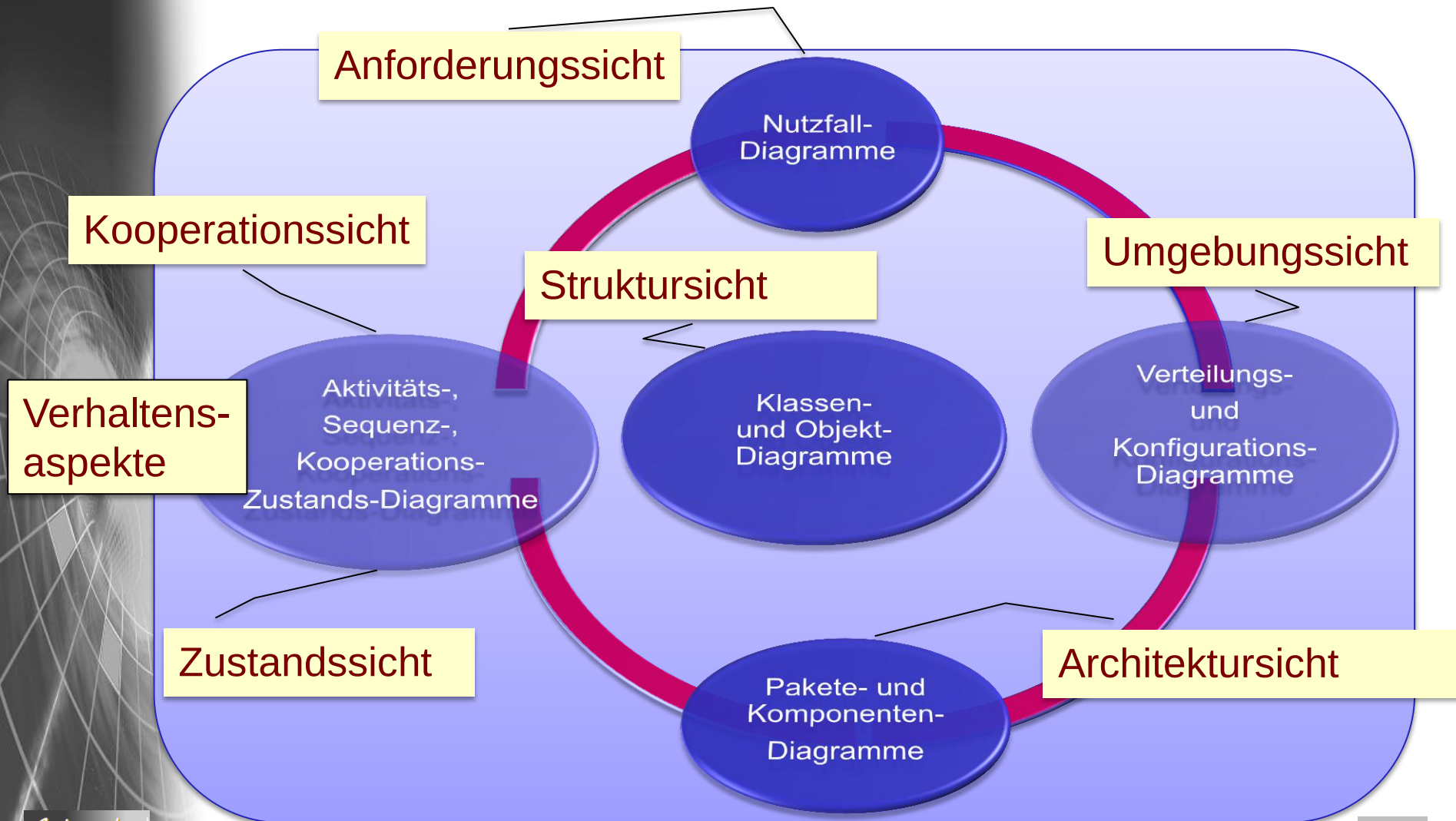
MDD:= Model Driven Development

- SW-Entwicklung ist modellzentriert (Modelle begleiten ges. SW-Lebenszyklus)
- automatische Transformationen für Modellübergänge
- spezifische Analysen (Checker, Simulatoren, ...)
- partielle oder komplette Codegenerierung

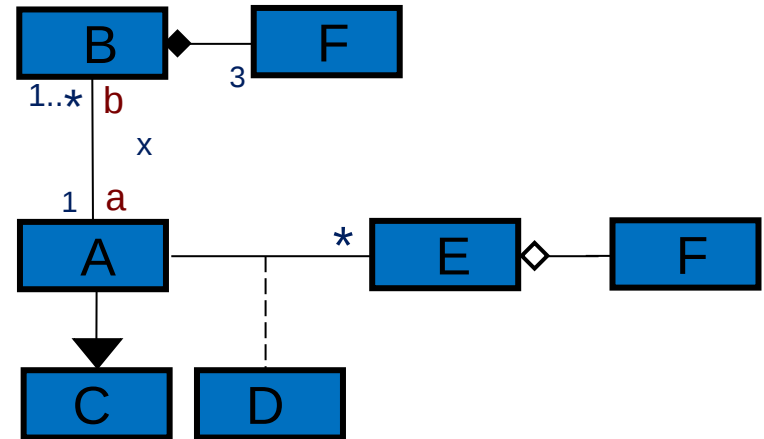
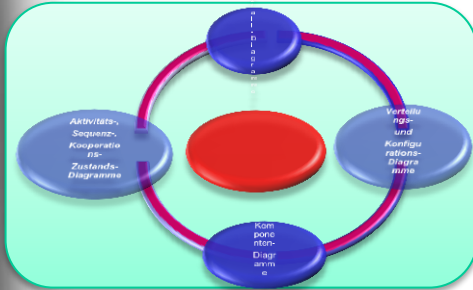


Diagrammarten ~ UML-Teilsprachen

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,
jedes 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 (Komposition)
permanente Teil-Ganzes-Beziehung
- Jedes B-Objekt besitzt 3 F-Objekte als (Aggregation)
zeitweilige Teil-Ganzes-Beziehung

Klassen und Objekte

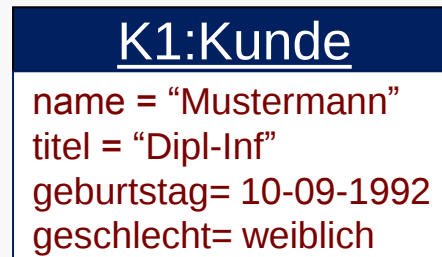
passive Klasse

aktive Klasse



- 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

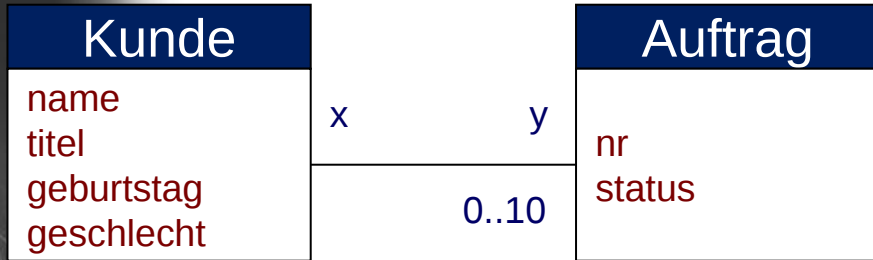
Attribut-Belegung
(Zustand zu einem
Zeitpunkt)
des Kunden-Objektes K1



:Kunde

anonymes Objekt

Dualität von Attributen und Assoziationsenden

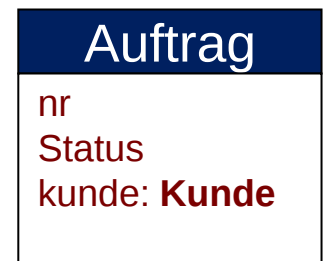


Lesart:

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

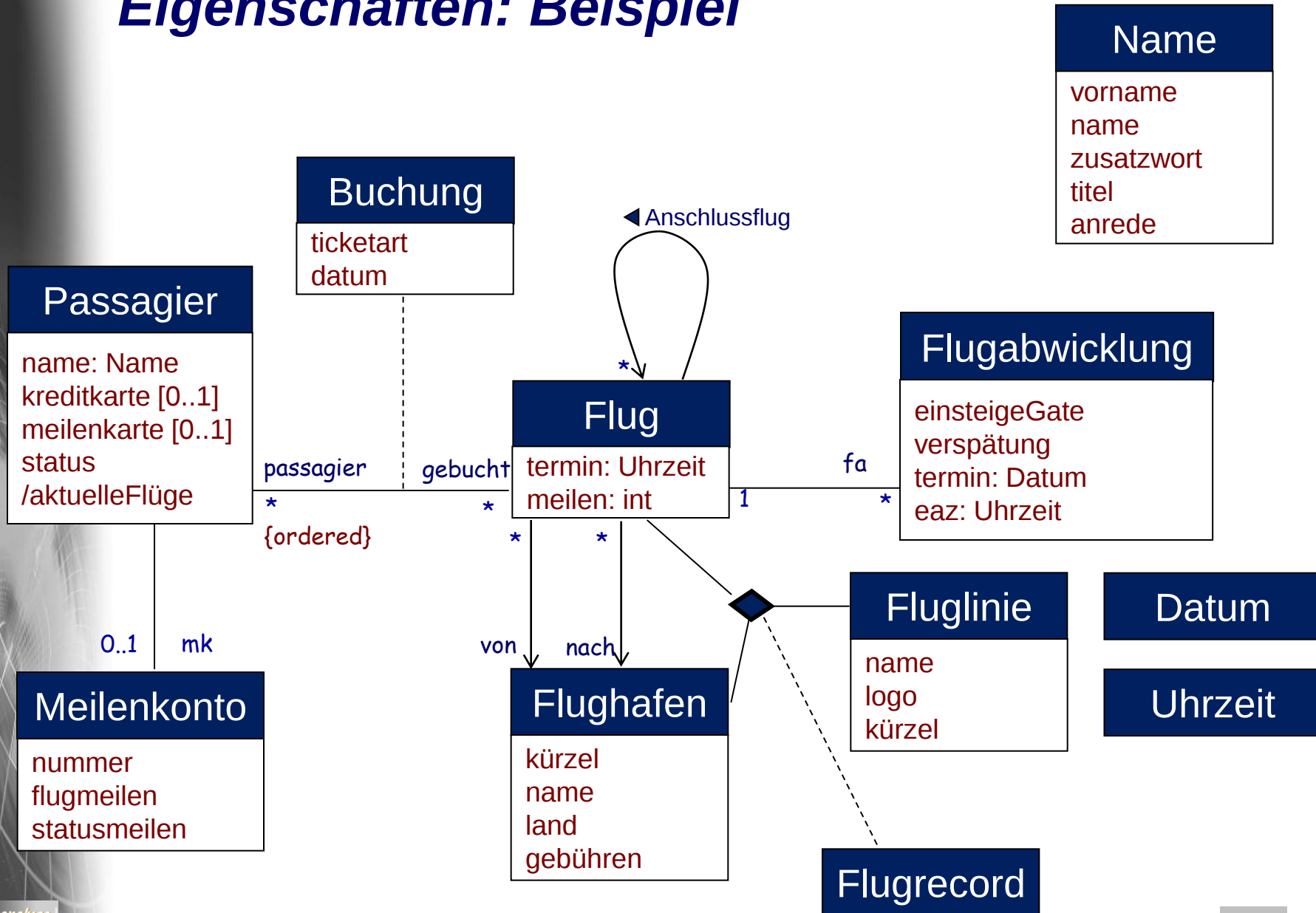
- ... für Assoziationsenden (auch für Attribute)
in {...} als Constraint

- / (abgeleitete Assoziation)
- readonly $\leftarrow \rightarrow$ unrestricted
- composite
- redefines Attr
- subsets Attr
- union
- unique $\leftarrow \rightarrow$ nonunique
- ordered $\leftarrow \rightarrow$ unordered
- seq (Vor.: Multiplizität > 1)
- bag (Vor.: Multiplizität > 1)

default

bedingt kombinierbar

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

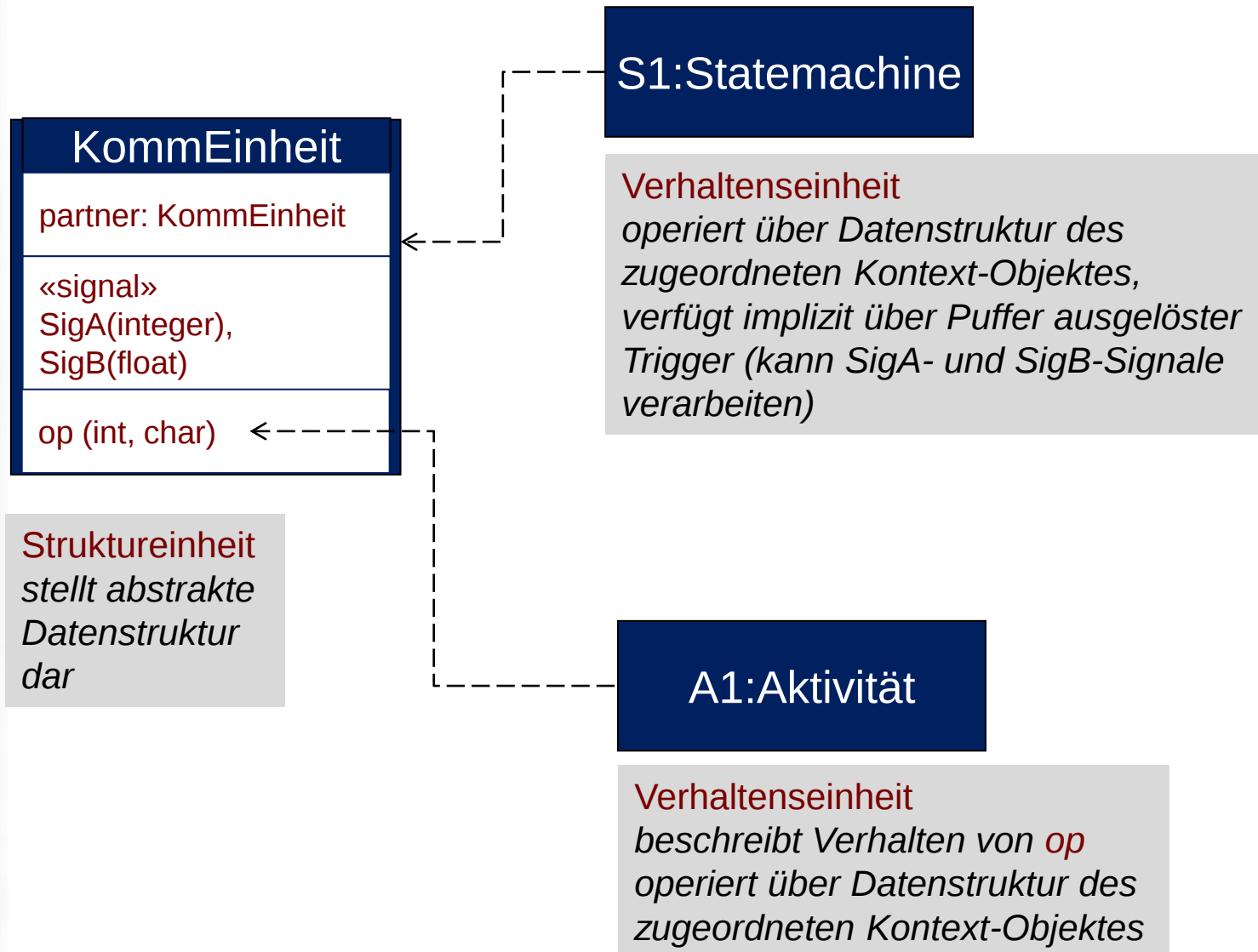
- Eintrittsaktionen
- Do-Aktivität
- Austrittsaktionen

C) Zustand als Classifier

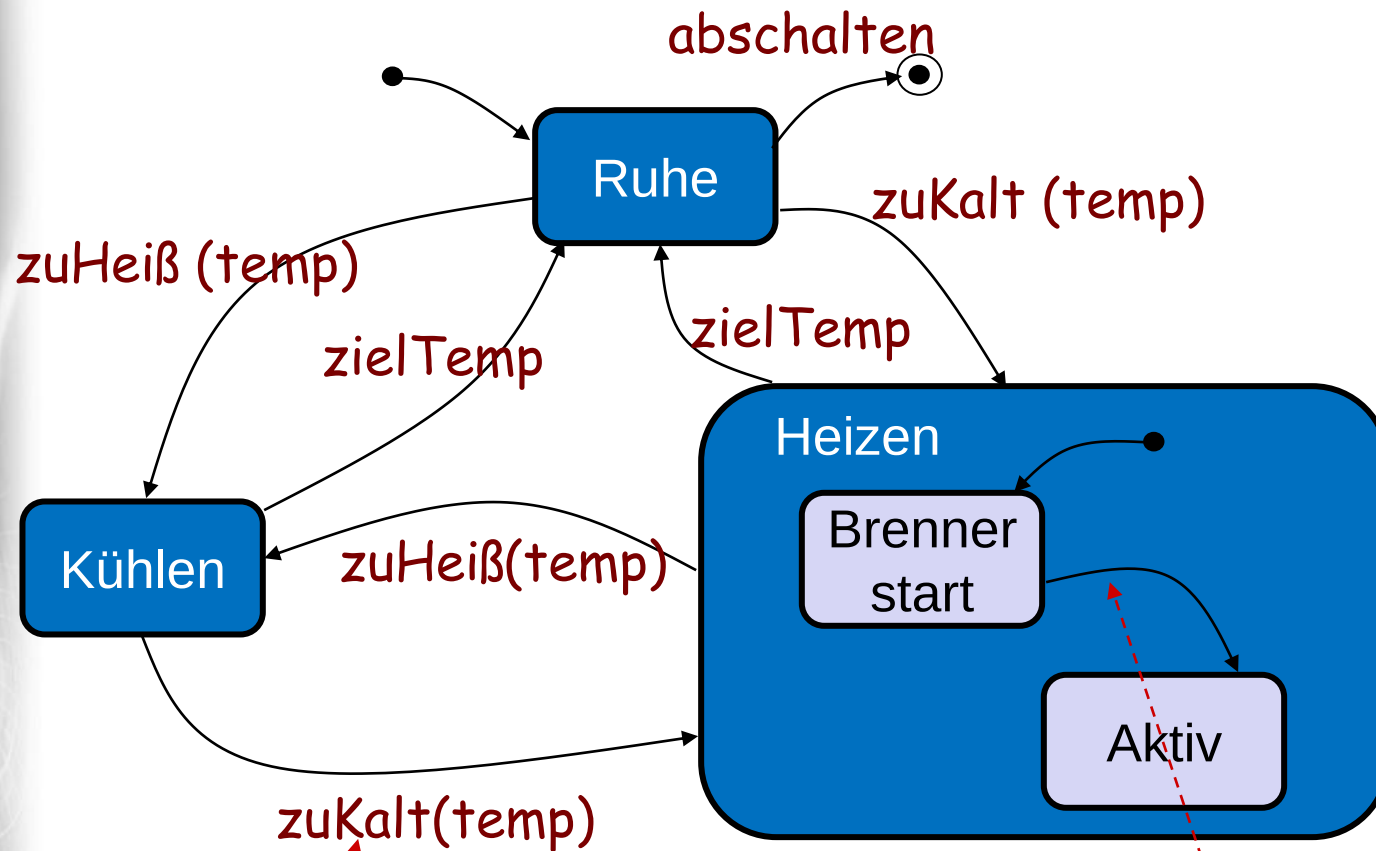
(Vererbung, Instanziierung, ...)

Im Kontext einer aktiven Klasse

Klassen – Aktivitäten - Zustandsmaschinen

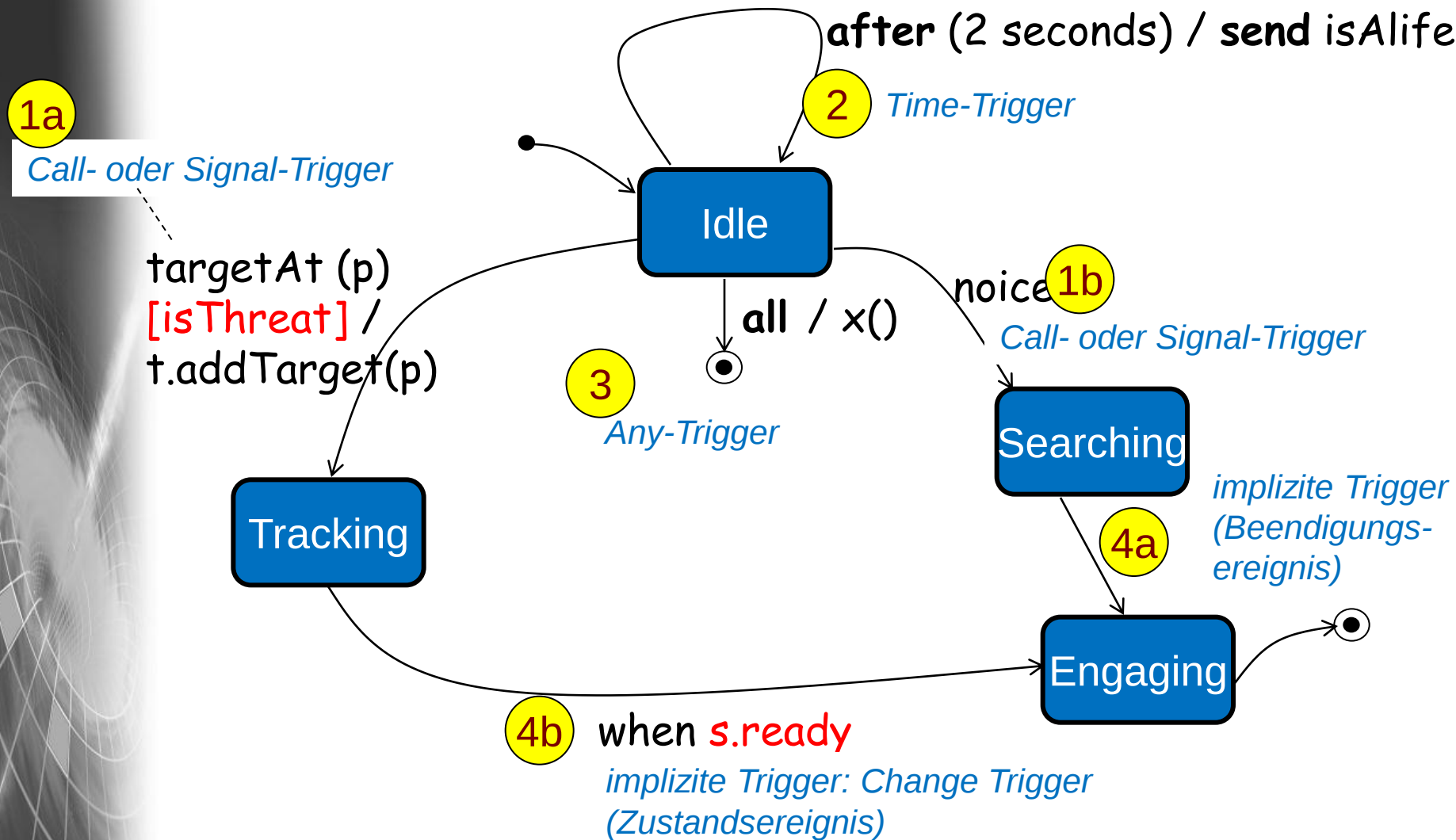


Zustandsdiagramm: Heizungssteuerung

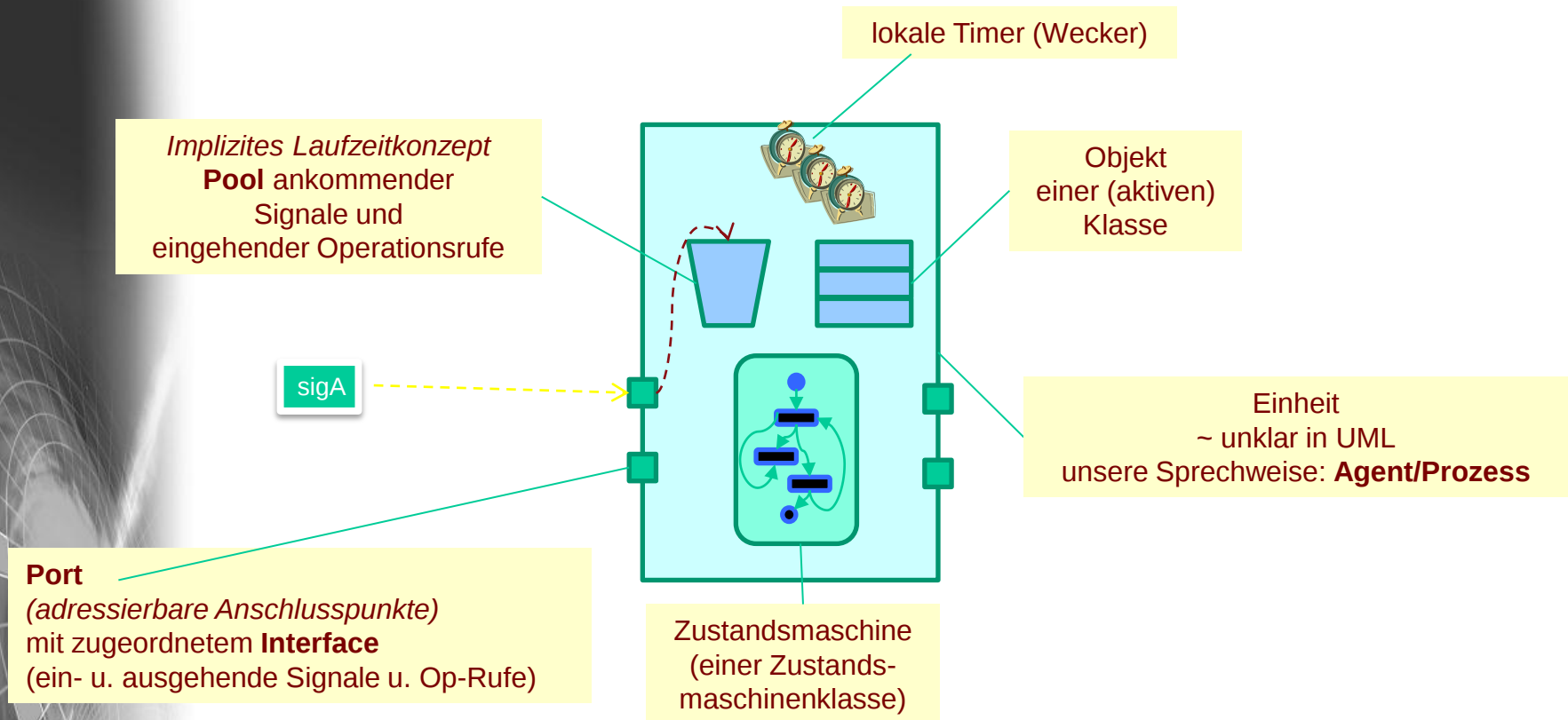


- Ereignisse lösen Transitionen aus, wenn dafür Trigger vorhanden sind
- Trigger können explizit angegeben sein, oder sie ergeben sich implizit

Verschiedene Trigger-Arten



Automaten zur Laufzeit



Zustand eines Agents zu einem Zeitpunkt:

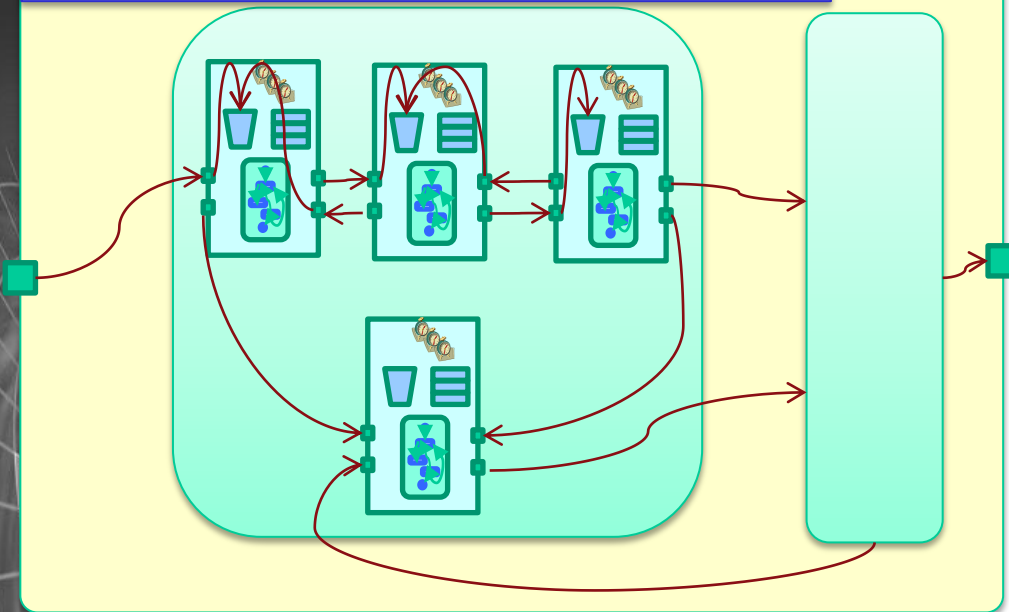
- Stand der gestarteten Timer
- Belegung des Empfang-Pools (inklusive aktueller Parameter der Signale u. Op-Rufe)
- Wertebelegung der Attribute des zugeordneten Objektes
- Zustand der Zustandsmaschine
- Befehlsregister (falls im Zustandsübergang)

Achtung (noch nicht behandelt):

- Spezialisierung (Vererbung) von Zustandsautomaten)
- hierarchische Zustände
- orthogonale Zerlegung (in parallel) agierende Untermaschinen

System als Agent-Ensemble

- Agenten kommunizieren asynchron



UML-Probleme

- **Pool-bearbeitung:**
sog. semantischer Variationspunkt
(Reihenfolge, ...)
- **Adressierung** von Signalen
(nicht komplett gelöst,
Bekanntmachung der Agenten/Ports)
- **Übertragung** von Signalen
(ungelöst: Zeitverbrauch, Sicherheit)
- semantischer Variationspunkt
Offenheit bzgl. **Action-Sprache**
Datentypen, Anweisungen
- Do-Aktivität kann sowohl zeitdiskret als
auch **zeitkontinuierlich** sein

Es gibt kein UML-Werkzeug, das das Agent-Konzept allgemein unterstützt.

ABER: es gibt praktikable Teillösungen

SDL (Specification and Description Language) ← zeitdiskret

1. Einführung

1. Systemsimulation – was ist das?
2. Ein Blick zurück in die Anfänge
3. Modelle und Originale
4. Modellierungssprachen, Simulationsumgebungen
5. Beispiele 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

Verhaltens- und Zustandsgrößen

Modellierungsaspekte realer oder gedachter Phänomene

- Existenz/Substanz (Ausdehnung in Raum und Zeit)
repräsentiert durch statische und dynamische Objekt-Strukturen
- Verhaltensgrößen (messbare Eigenschaften)
repräsentiert durch Werte der Objektattribute
- Veränderung der Substanz (dynamisches Verhalten)
repräsentiert durch interagierende Objekte aktiver Klassen in Abhängigkeit einer Modellzeit bei Nutzung von Objekten passiven Klassen

essentielle Verhaltensgrößen

- nicht immer beobachtbar
- **Zustandsgrößen** als ausgezeichnete Verhaltensgrößen (spielen eine zentrale Rolle bei der Modellierung)

Zustandsgrößen

... sind

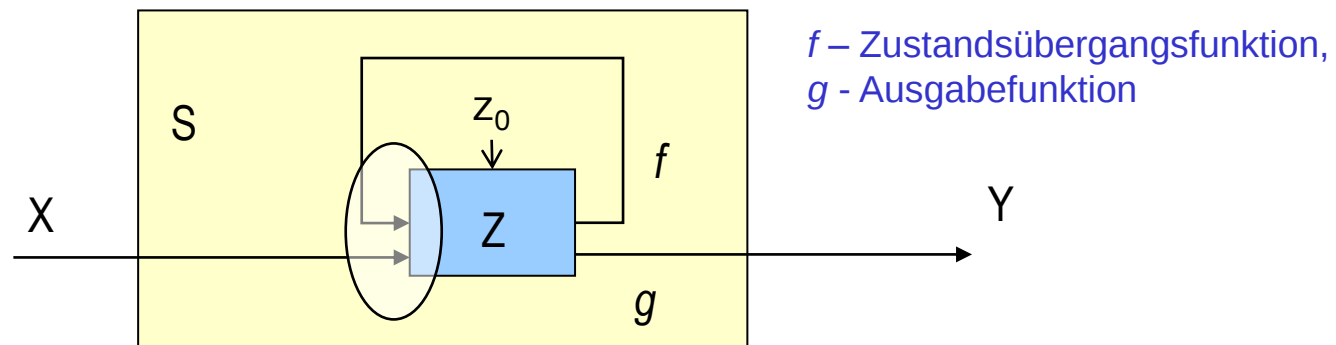
- Modellbeschreibungsgrößen, aus denen sich der Zustand eines Systems **vollständig** ergibt (Gedächtnis eines Systems)
→ Basis der Verhaltensbeschreibung
- Zustandsgrößen sind voneinander **unabhängig**
→ eine Zustandsgröße kann nicht als Kombination anderer Zustandsgrößen dargestellt werden
- sind **nicht immer eindeutig** definierbar
- sind i. Allg. strukturierte Größen



Zustandsänderungen (Prinzip)

- Sei Z n -dimensionaler Zustandsvektor (Zustand z = Belegung von Z), der Zustandsgrößen eines (Teil-)Systems S zu diesem Zeitpunkt beschreibt
- der (neue) Zustand ergibt sich aus dem bisherigen (aktuellen) Zustand bei Berücksichtigung von "Zuwachs" und "Reduktion (negativer Zuwachs)" für die Zustandsgrößen im betrachteten Zeitraum des Zustandswechsels
- ausgehend von einem ausgezeichneten Anfangszustand z_0

Zeit $t_0 \quad t_1 \quad t_2 \quad t_3$
 $z_0 \rightarrow z_1 \rightarrow z_2 \rightarrow z_3 \rightarrow \dots$ Zustandsentwicklung in Abhängigkeit der Zeit



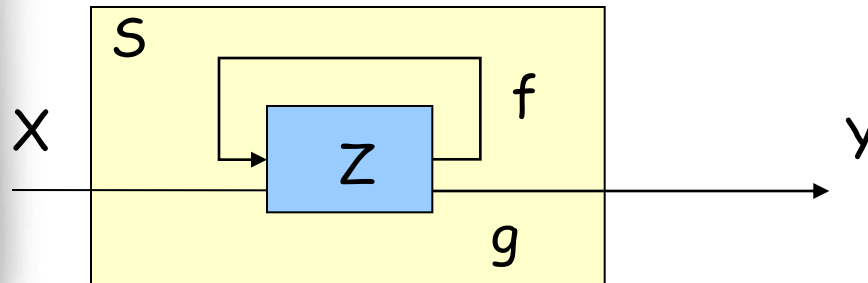
Eingabe X
 Zuwachs
 im Zeitintervall $(t_k, t_{k+1}]$

Änderung des Zustandsvektors Z
 im Zeitintervall $[t_k, t_{k+1}]$
 in Abhängigkeit von $x(t_{k+1}), z(t_k)$

Ausgabe Y
 äußere Reaktion
 des Systems auf die Eingabe
 im Zeitintervall $(t_k, t_{k+1}]$

Allgemeine (Teil-)Systemdefinition

dient mehr der Klassifikation von Verhaltensmodellen

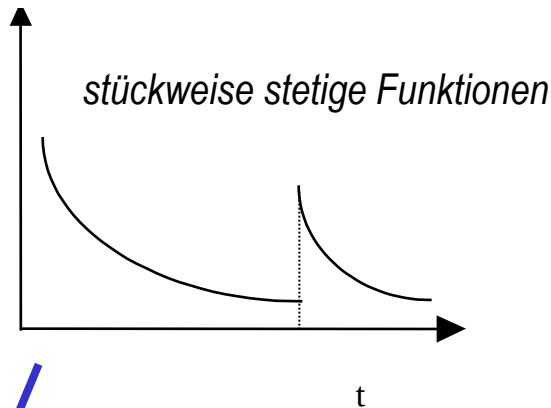


$$S = (Z, z_0, X, Y, f, g, \text{time})$$

- Z Menge der möglichen Zustände
- $z_0 \in Z$ Anfangszustand
- X Menge der möglichen Eingaben
- Y Menge der möglichen Ausgaben
- **time** Zeitbasis als $(T, \leq, t_0)$ mit
 - Menge möglicher Zeitpunkte T ,
 - einer Ordnungsrelation $\leq$ und
 - einem minimalen Element t_0
- f $Z \times X \times T \rightarrow Z$ als Zustandsübergangsfunktion
- g $Z \times X \times T \rightarrow Y$ als Ausgabefunktion

Arten von Zustandsänderungen

zeitkontinuierliche
Zustandsänderung



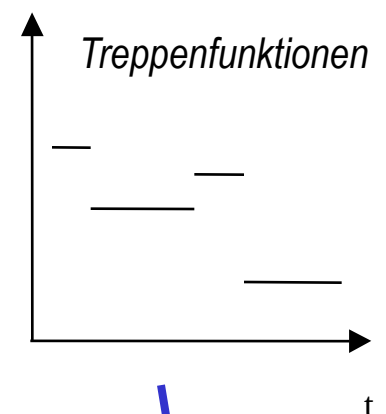
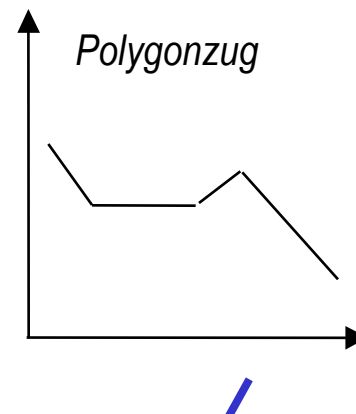
$$z'(t) = f(z(t), x(t), t)$$

mit $z(t) \in Z, x(t) \in X, t \in T$

Differentialgleichungen

numerische
Lösungsverfahren

zeitdiskrete
Zustandsänderung



$$z(t_{n+1}) = f(z(t_n), x(t_{n+1}), t_{n+1})$$

mit $z(t_n) \in Z, x(t_{n+1}) \in X, t_{n+1} \in T$

Differenzengleichungen
zelluläre Automaten

Ereignissimulationen

Prozesssimulation

Verhaltensklassen zeitkontinuierlicher Art

Differentialgleichungen

n-ter Ordnung (**erster Ordnung**)

partielle
Differentialgleichungen

gewöhnliche
Differentialgleichungen

nichtlineare
Differentialgleichungen

lineare
Differentialgleichungen

Randwertaufgaben

Anfangswertaufgaben

Klassifikationsmöglichkeiten

- zeitkontinuierliche Prozesse im Beschreibungsmodell
- Approximation durch zeitdiskrete Prozesse (Anwendung numerischer Integrationsverfahren)
- zeitdiskrete Prozesse als Folgen von Ereignisrealisierungen im Simulationsmodell

ODEMx unterstützt:

Gewöhnliche DGL-Systeme 1.Ordnung als AWA
(linearer und nichtlinearer Art)