VORLESUNG

Automatisierung industrieller Workflows

Teil B: Die Spache UML

Joachim Fischer

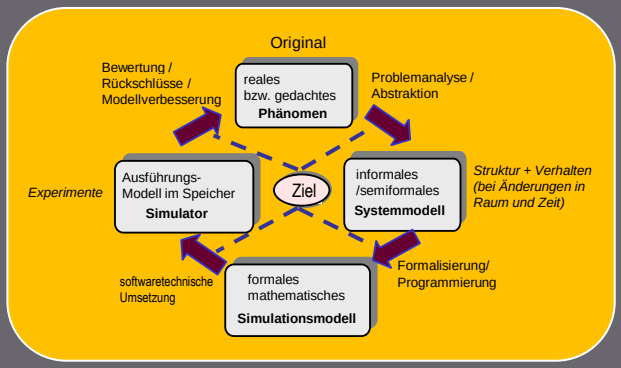
Inhalt

- **Teil A**
  Aspekte von Modellierung und Simulation dynamischer Systeme
- **Teil B**
  Die Modellierungssprache UML
- **Teil C**
  Die ausführbare Modellierungssprache SLX
- **Teil D**
  Modellierung von Lieferketten

- **A.1**
  Systemsimulation – was ist das?
- **A.2**
  Ein Blick zurück in die Anfänge
- **A.3**
  Modelle und Originale
- **A.4**
  Modellierungssprachen, Simulationsumgebungen
- **A.5**
  Bespiele aus der aktuellen Forschung
- **A.6**
  Paradigma der objektorientierten Modellierung
- **A.7**
  Klassifikation dynamischer Systeme
- **A.8**
  Schmiedewerke Gröditz GmbH
- **A.9**
  Zusammenfassung

Zusammenfassung (1)

... formuliert als Fragensammlung

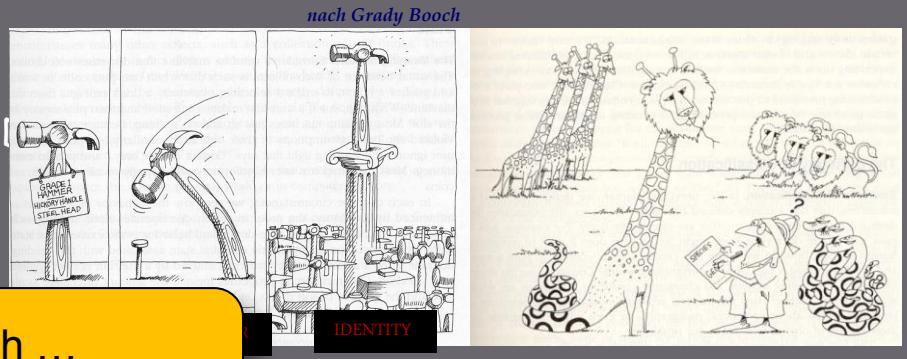


Computersimulation: Charakteristischer ‘Scientific Workflow’

- Charakterisieren Sie Computersimulation als Untersuchungsmethode.
- Was versteht man unter einem Simulator?
- Welche Zeitkonzepte lassen sich bei der Computersimulation unterscheiden?
- Was versteht man unter Echtzeitsimulation?
- Wie kommt man zu ausführbaren Simulationsmodellen? (Bedeutung Modellierungssprachen)
- Warum kann es kein perfektes Simulationsmodell geben? Erläutern Sie die Bedeutung des Untersuchungsziels für den Scientific Workflow “Computersimulation”.
- Was versteht man unter einem (dynamischen) System?
- Welche Bedeutung haben Strukturäquivalenzen von Original- und Modellsystem und entsprechende Verhaltensanalogien für die Computersimulation?
- Wonach werden Modellsysteme klassifiziert?
- Erläutern Sie die Spezifik des Scientific Workflows für die Phase einer Simulationsmodellentwicklung und die Phase einer Modellnutzung. Was versteht man unter Modellvalidierung?

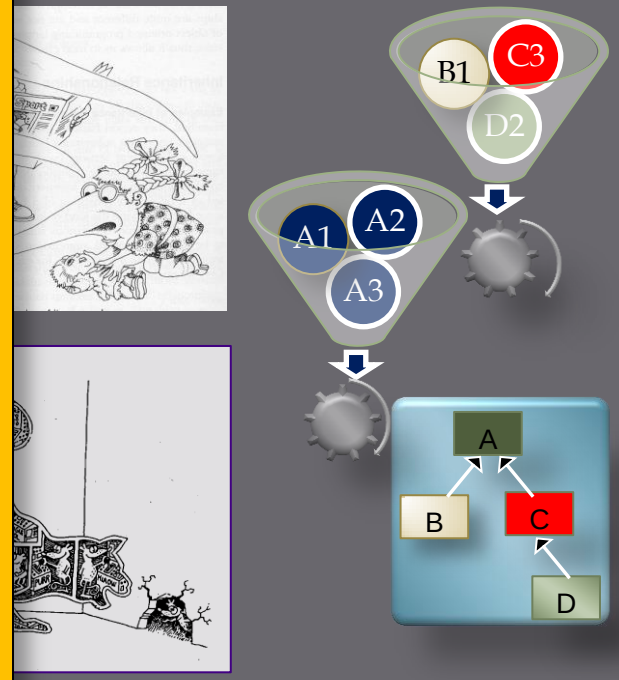
Modellierungsparadigmen (Wdh.)

- Objektorientiertes Abstraktionskonzepte
- eigenverantwortlich handelnde, interagierende [
  - Zustand (Attribute)
  - individuelles Verhalten (Methoden, Dienste)
  - Identität (Referenz)
- Klassen (Definition von Objekten)
- Unterscheidung zw.
  - aktiven und
  - passive Klassen
- Identifikation verschiedenster Bezi
  (bzw. Instanzmengen)
  - Navigierbarkeit
  - Abhängigkeit
  - …
- Beziehung zwischen Klassen
  - Spezialisierung / Generalisierung
  - abstrakte und konkrete Klassifizie
- Polymorphie von (getypten) Referen



zusätzlich …

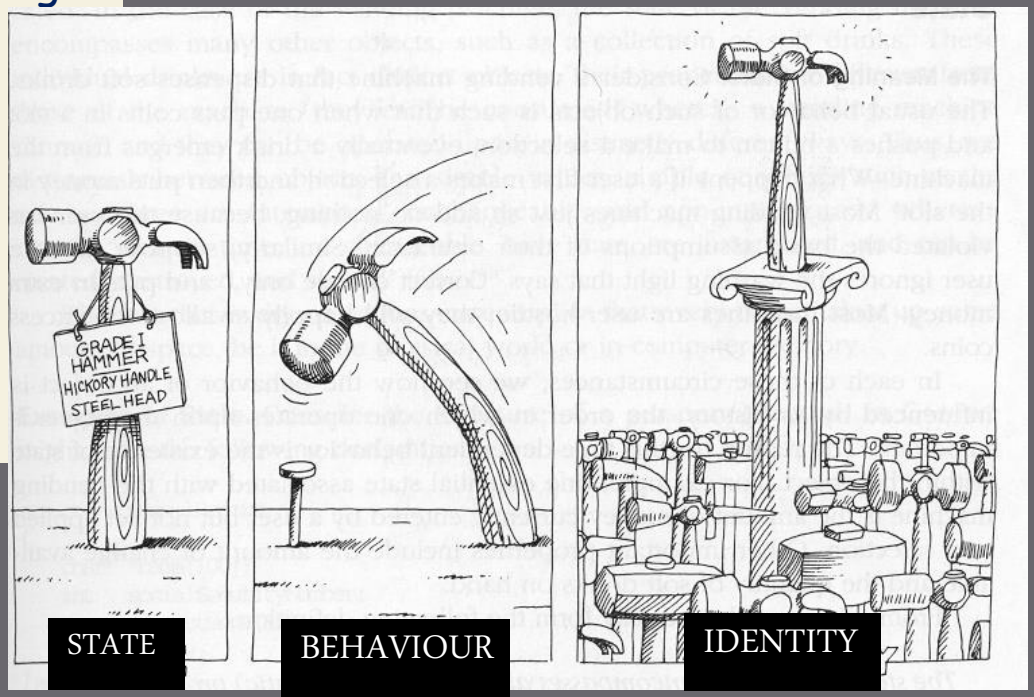
- Gruppierung von Modellelementen
- Komposition/ Dekomposition
- Nebenläufigkeit/ Parallelität/ Synchronisation
- zeitdiskretes/ zeitkontinuierliches Verhalten



OO-Abstraktionskonzept: Objekt (Wdh.)

eigenverantwortlich interagierende Dinge oder Akteure (Objekte)

nach Grady Booch



~ Vorgänge in RAUM und zeit

<u>charakterisiert durch:</u>

- Zustand (Attribute)
- individuelles Verhalten als Lebenslauf (Methoden, Dienste)
- Identität (Referenz)

<u>Bsp-1:</u> ein Thread (als Objekt einer aktiven Klasse) führt Operationen über Objekte passiver Java-Klassen aus

grundsätzlich:
selbständig/aktiv – unselbständig/passiv

→ Klasseneigenschaft

momentan:
aktiv – blockiert

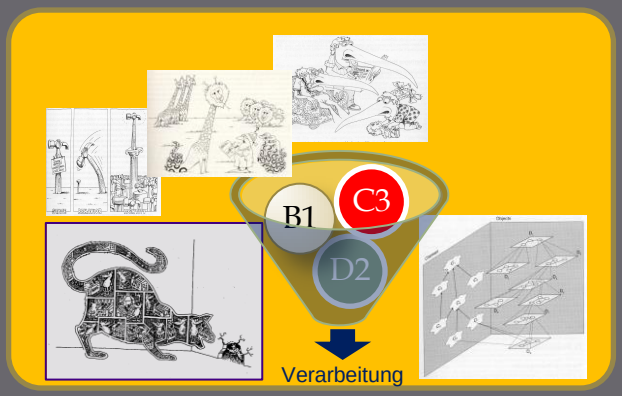
momentan:
benutzt – unbenutzt

→ Objekteigenschaft

<u>Bsp-2:</u> eine Stauchpresse (als Objekt einer aktiven Klasse) führt Verformungsoperationen über Eisenwerkstücke (Objekte passiver Klassen aus)

Zusammenfassung (2)

... formuliert als Fragensammlung



Paradigma der objekt-orientierten Modellierung

- Erläutern Sie das Paradigma der OO-Modellierung.
- Worin besteht die Natürlichkeit dieses Paradigmas bei der Abstraktion eines Originalsystems zu einem Modellsystem?
- Was versteht man allgemein unter Zustandsgrößen bei der Betrachtung eines Systems?
  Wie repräsentiert sich der Zustand eines Systems, wenn es konsequent objekt-orientiert beschrieben ist?

Inhalt

- **Teil A**
  Aspekte von Modellierung und Simulation dynamischer Systeme
- **Teil B**
  Die Modellierungssprache UML
- **Teil C**
  Die ausführbare Modellierungssprache SLX
- **Teil D**
  Modellierung von Lieferketten

- **B.1**
  Wozu UML im Kontext der Computersimulation?
- **B.2**
  UML-Teilsprachen, Sprachkonzepte
- **B.3**
  Klassendiagramme
- **B.4**
  Verhaltensbeschreibung
- **B.5**
  OCL

Die UML

551 v. Chr. bis 479 v. Chr.

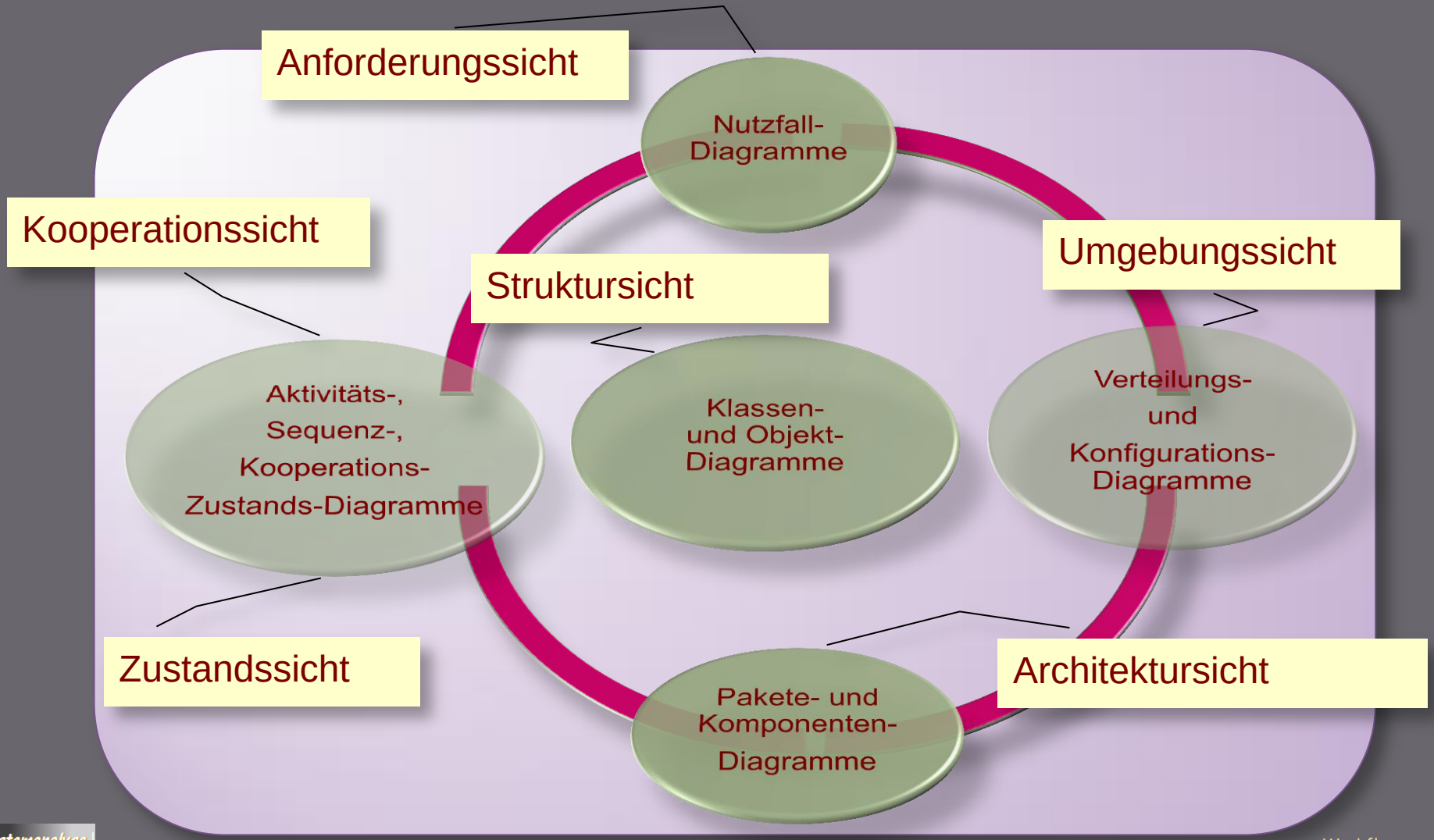
„Wenn die Sprache nicht stimmt,
ist das was gesagt wird, nicht das, was gemeint ist."
(Konfuzius)

- 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)

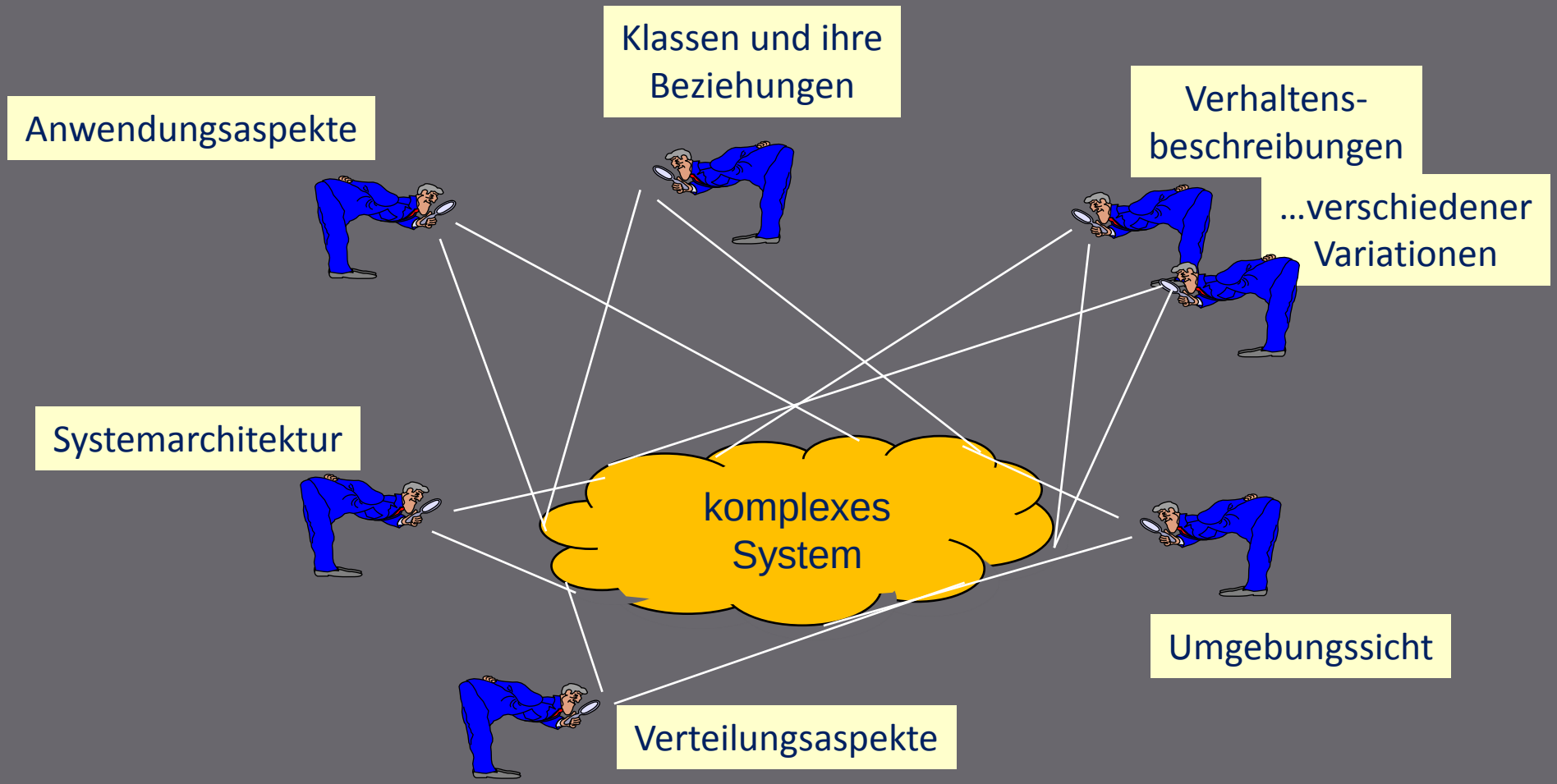
  …
- UML-Modelle sind in erster Linie graphische Repräsentationen in Form von Diagrammen

Diagrammarten ~ UML-Teilsprachen

Modellklassen zur Beschreibung von dynamischen Systemen

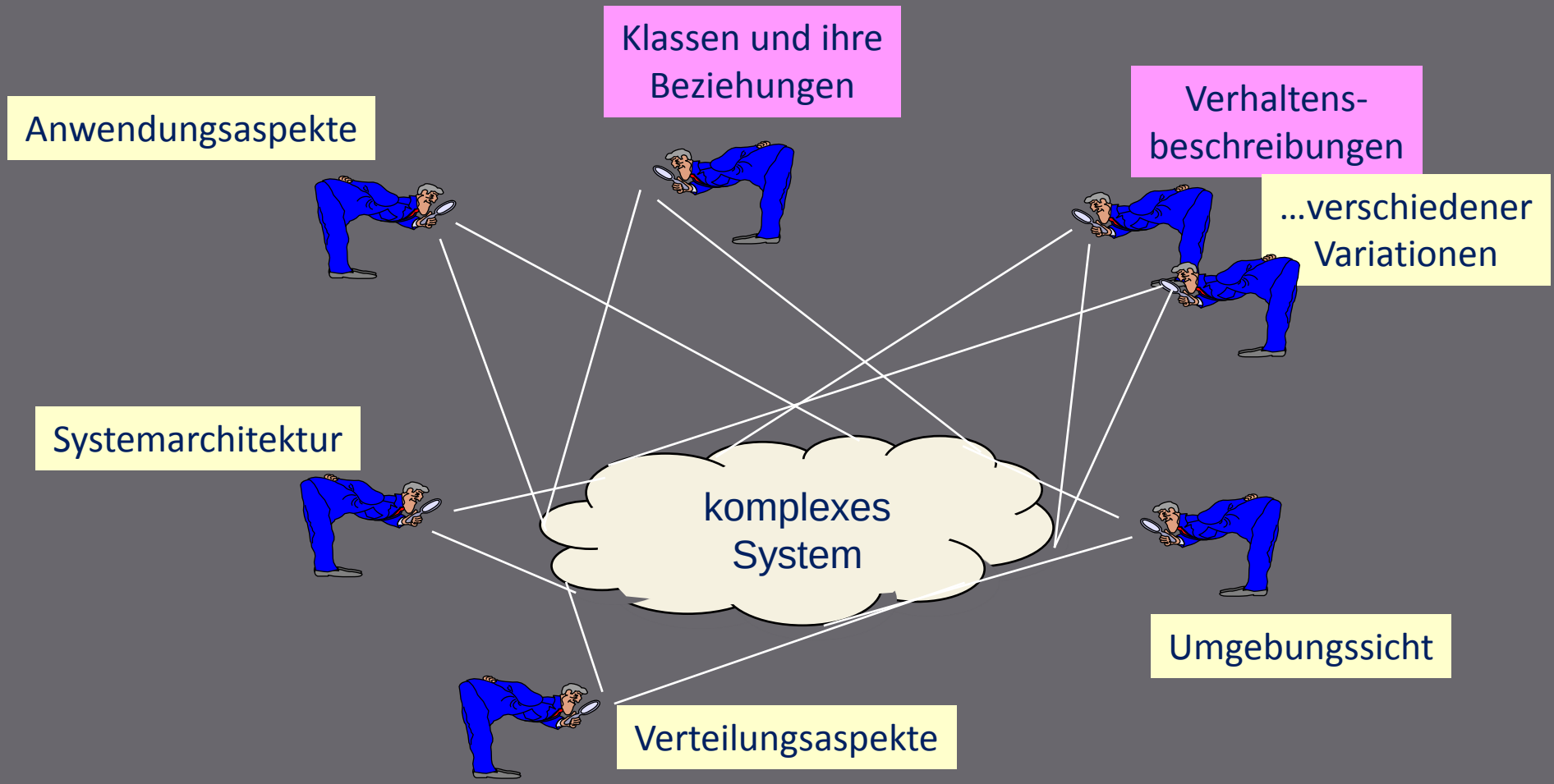


UML-Sichtweisen auf ein komplexes HW/SW-System



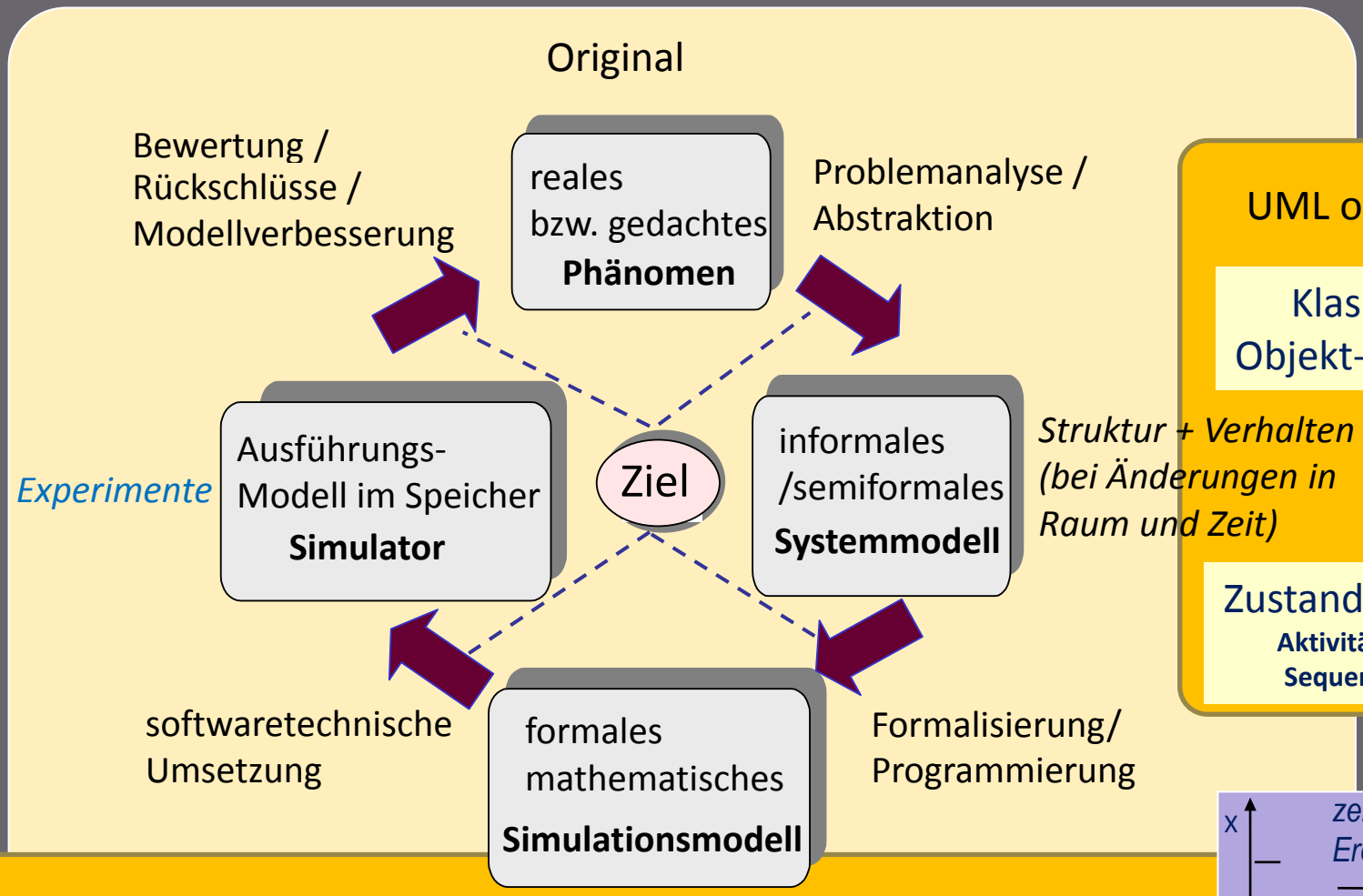
➔ verschiedene Teilsprachen (Diagramme)

UML-Sichtweisen auf Systeme, die uns interessieren



Achtung: Wir wollen mit UML nicht das Simulationsprogrammsystem beschreiben, sondern das zu simulierende System als ein erstes semiformales Modell

Präzisierung unseres Scientific Workflows



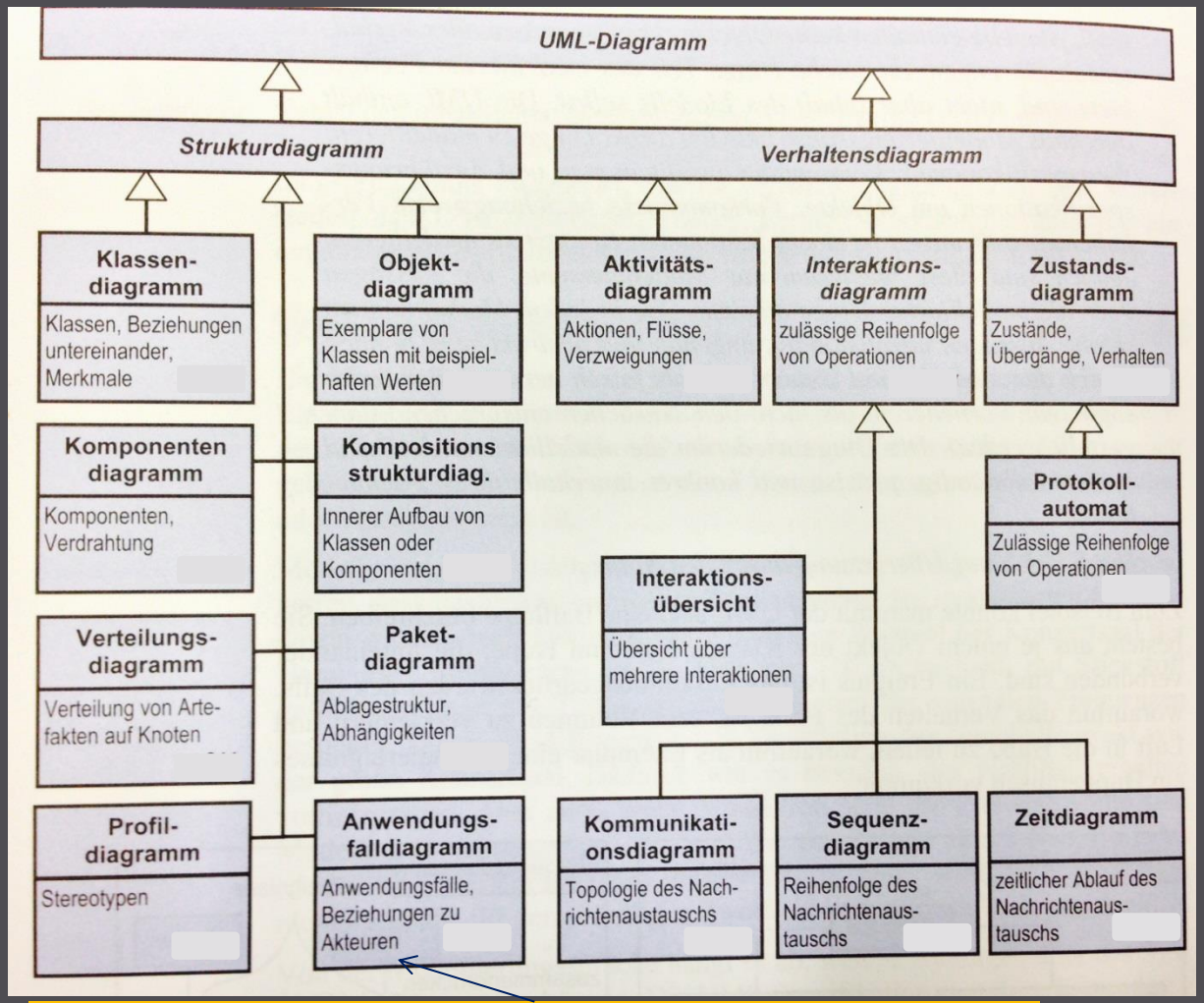
Inhalt

- **Teil A**
  Aspekte von Modellierung und Simulation dynamischer Systeme
- **Teil B**
  Die Modellierungssprache UML
- **Teil C**
  Die ausführbare Modellierungssprache SLX
- **Teil D**
  Modellierung von Lieferketten

- **B.1**
  Wozu UML im Kontext der Computersimulation?
- **B.2**
  UML-Teilsprachen, Sprachkonzepte
- **B.3**
  Klassendiagramme
- **B.4**
  Verhaltensbeschreibung
- **B.5**
  OCL

UML-Diagramm

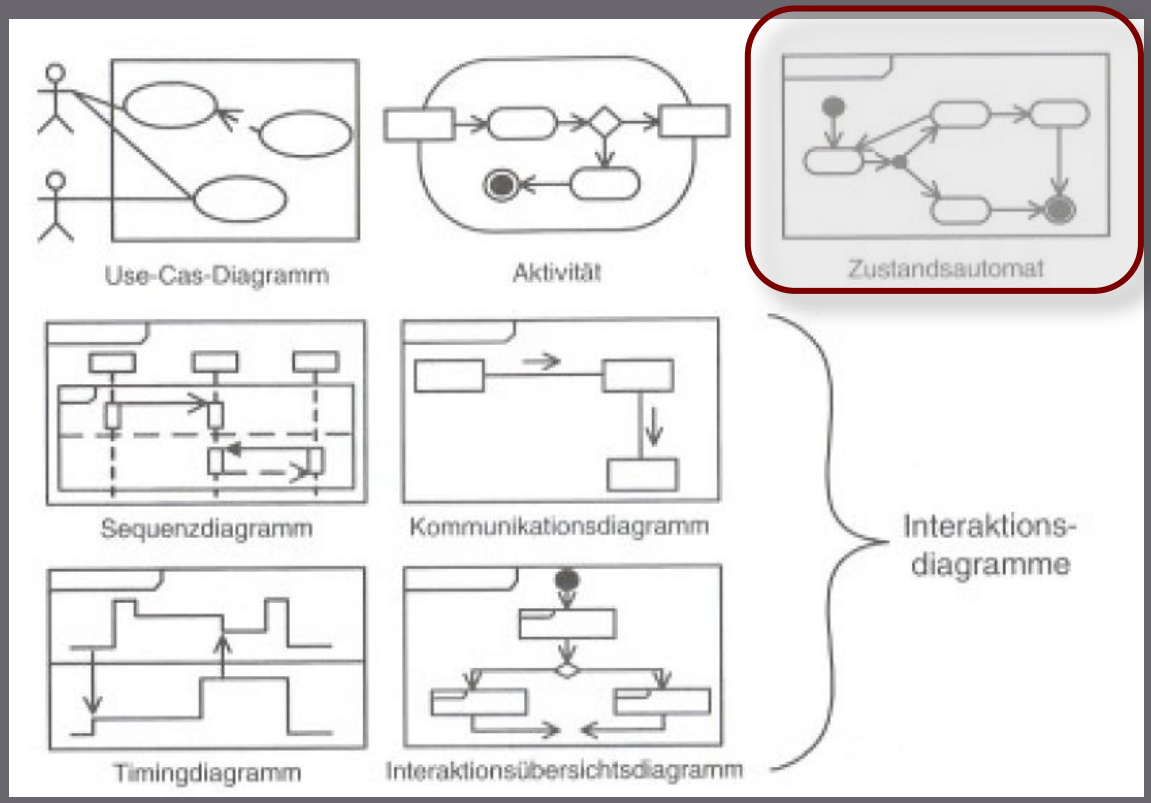
UML erlaubt aber auch beliebigen Mix in einem Diagramm (Tool-abhängig)



ACHTUNG: Abweichung
UML-Standard: Anwendungsfalldiagramm als Verhaltensdiagramm)

Zustandsautomaten, Allgemeines

- breitestes Anwendungsspektrum



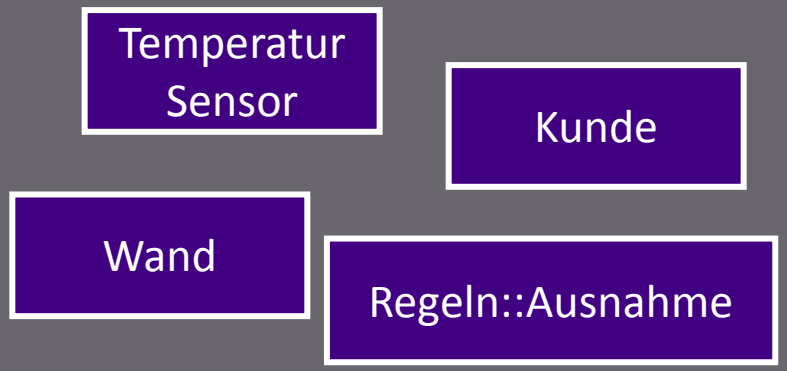
Inhalt

- **Teil A**
  Aspekte von Modellierung und Simulation dynamischer Systeme
- **Teil B**
  Die Modellierungssprache UML
- **Teil C**
  Die ausführbare Modellierungssprache SLX
- **Teil D**
  Modellierung von Lieferketten

- **B.1**
  Wozu UML im Kontext der Computersimulation?
- **B.2**
  UML-Teilsprachen, Sprachkonzepte
- **B.3**
  Klassendiagramme
- **B.4**
  Verhaltensbeschreibung
- **B.5**
  OCL

Klassendiagramm

- … zeigt eine Menge von
  - Klassen
  - Interfaces
  - Kollaborationen (collaborations) und
  - deren Beziehungen untereinander
- Beziehungen:
  - Abhängigkeit,
  - Generalisierung und
  - verschiedene Assoziationen
- hat (wie jedes Diagramm) einen Namen
- kann (wie jedes Diagramm) Anmerkungen und Constraints enthalten

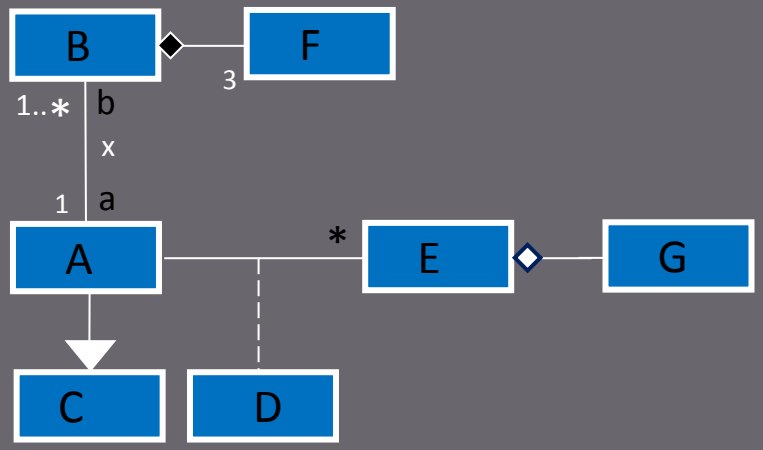


Paketname::Klassenname

Namenskonvention:
- beliebige Zeichenfolge (ohne Doppelpunkt)
- Empfehlung: Substantiv, Beginn mit Großbuchstabe
- 1-deutig in einem Kontext

Paketname::Elementname

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 G-Objekt (Aggregation)
  G-Objekt ist Teil von A (könnte aber gleichzeitig Teil von etwas anderem sein)
- Jedes B-Objekt besitzt 3 F-Objekte als (Komposition)
  F-Objekt kann (max.) nur Teil vom B-Objekt sein

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(**Real**)

op (**Integer**, **Boolean**)

- 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

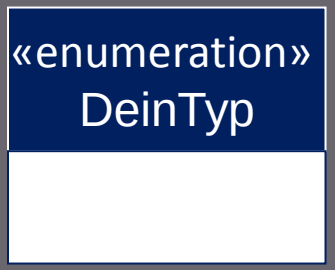
Typen von Klassenattributen

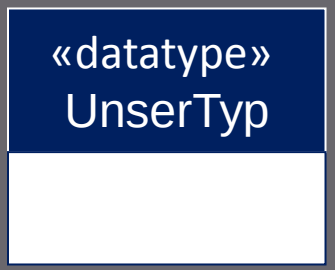
- Typen
  - Klassen
  - Datentypen
    - Primitive Datentypen
      - Boolean
      - Integer
      - UnlimitedNatural
      - String
      - (Real)
    - Aufzählungstypen
    - nutzerdefinierte Typen

Werte mit Identität

Werte ohne Identität

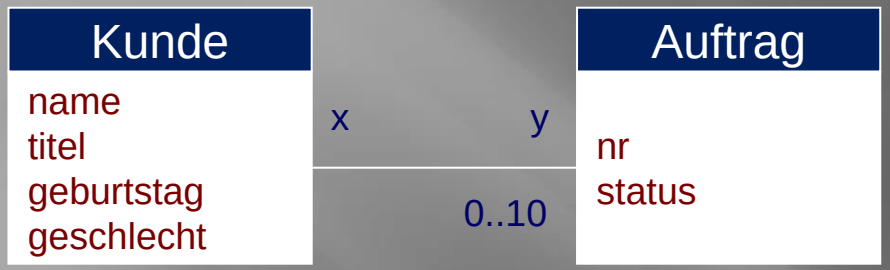
Keine innere Struktur
mit Operationen
ohne Seiteneffekte





innere Struktur
mit Operationen
mit u. ohne Seiteneffekte

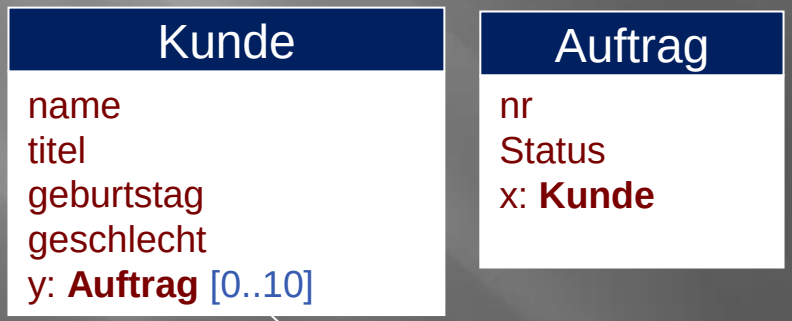
Dualität von Attributen und Assoziationsenden



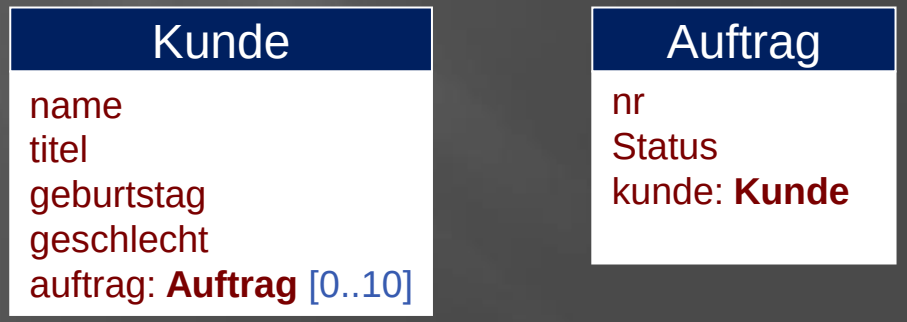
Lesart:
jedem Kunde-Objekt ist eine Kollektion y von Aufträgen zugeordnet

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

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



Kardinalität als Eigenschaften von Attributen/Assoziationsenden



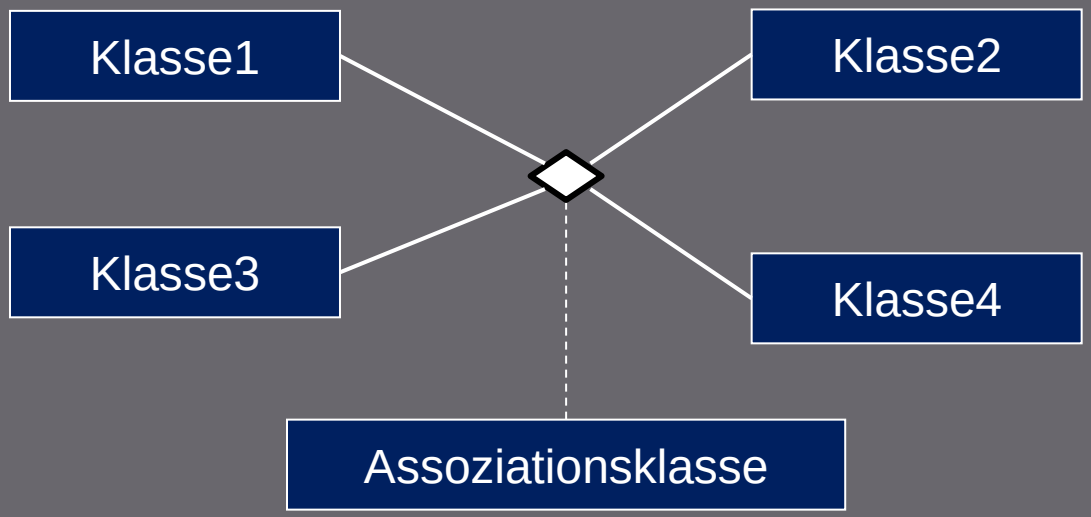
falls Assoziationsenden nicht benannt worden sind (also wenn Angabe y fehlt)

Beziehungen zwischen Klassen bzw. Objektmengen

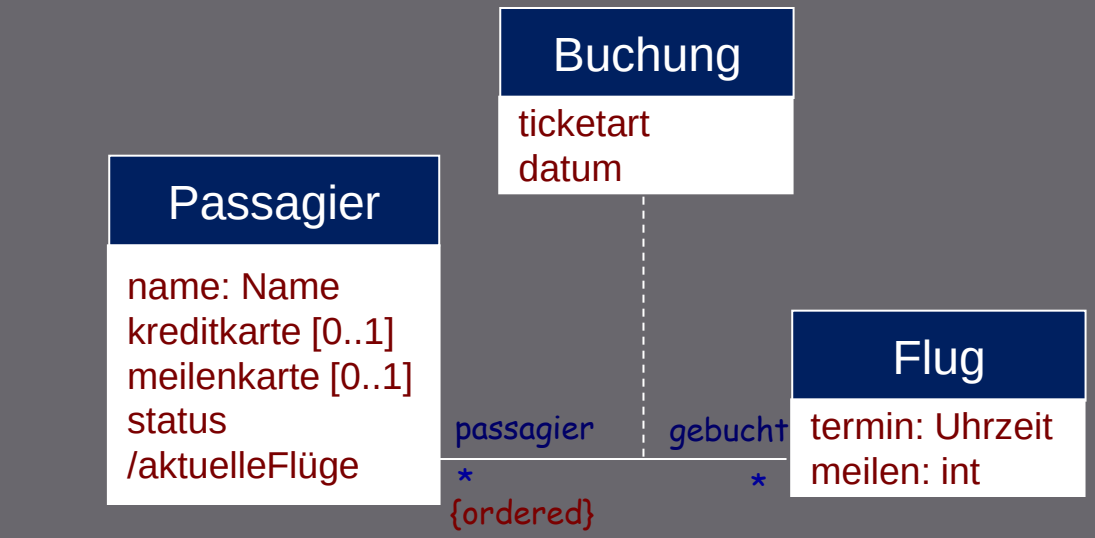
Assoziation

- beschreibt als Relation die gemeinsame Struktur und Semantik von Mengen von Objektverbindungen der beteiligten Klassen
- Objektverbindungen (Links) sind Instanzen einer Assoziation (eineAusprägung von Classifier)
- Eine Assoziation kann einen Namen haben
  (beschreibt worin oder weshalb eine Beziehung besteht)
- Die Enden tragen immer einen Namen
  (beschreiben die Rollen, die die Objekte der Klassen am jeweiligen Ende spielen)
  Ist kein Name angegeben, wird der kleingeschriebene Klassenname angenommen
- Für jede Rolle kann eine Reihe zusätzlicher Angaben gemacht werden:
  - Multiplizität (Standardwert=1),
  - Sichtbarkeit,
  - Eigenschaften
- Dualität von Assoziationsende und Attribut (jedes Rollenende könnte ein Attribut der gegenüberliegende Klasse oder Attribut einer evtl. Assoziationsklasse darstellen)

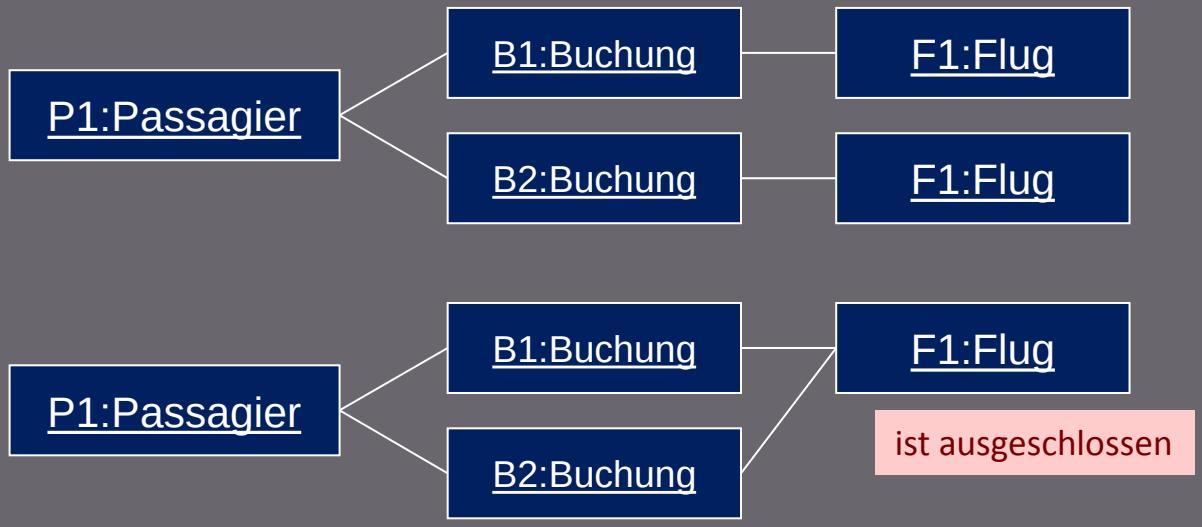
Mehrstellige Assoziation



Assoziationsklasse



jedes Verbindungspaar ist einmalig



ist ausgeschlossen

Liste möglicher Eigenschaftsangaben

… für Assoziationsenden aber 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)

default

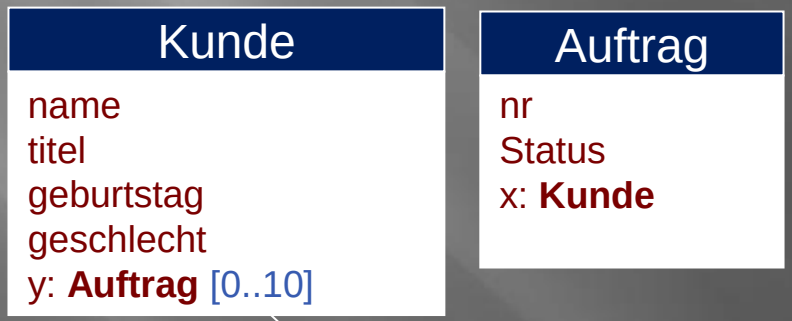
bedingt kombinierbar

Dualität von Attributen und Assoziationsenden

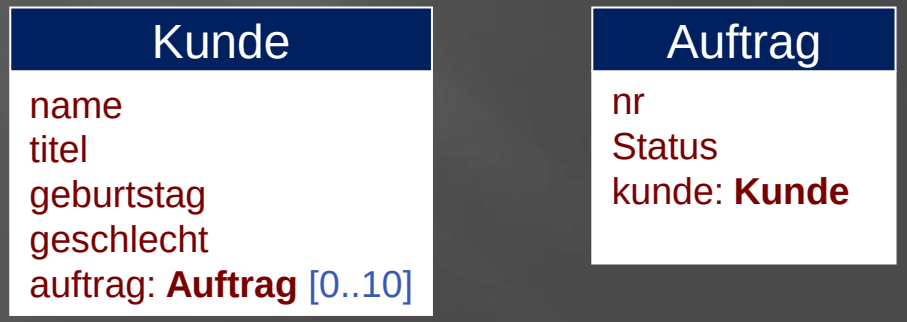
Lesart:
jedem Kunde-Objekt ist
eine Kollektion y von Aufträgen zugeordnet

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*

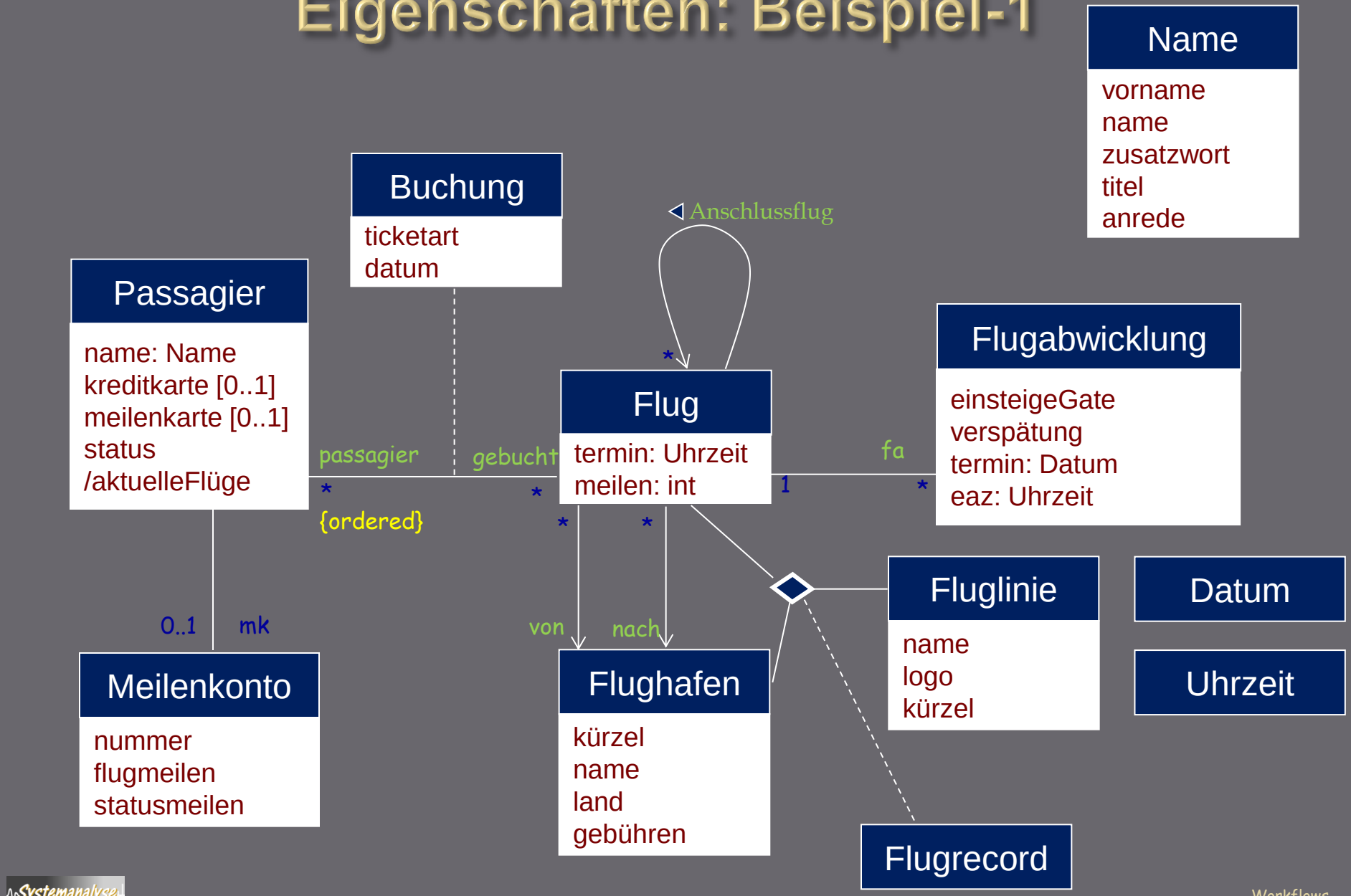


Kardinalität
als Eigenschaften von
Attributen/Assoziationsenden



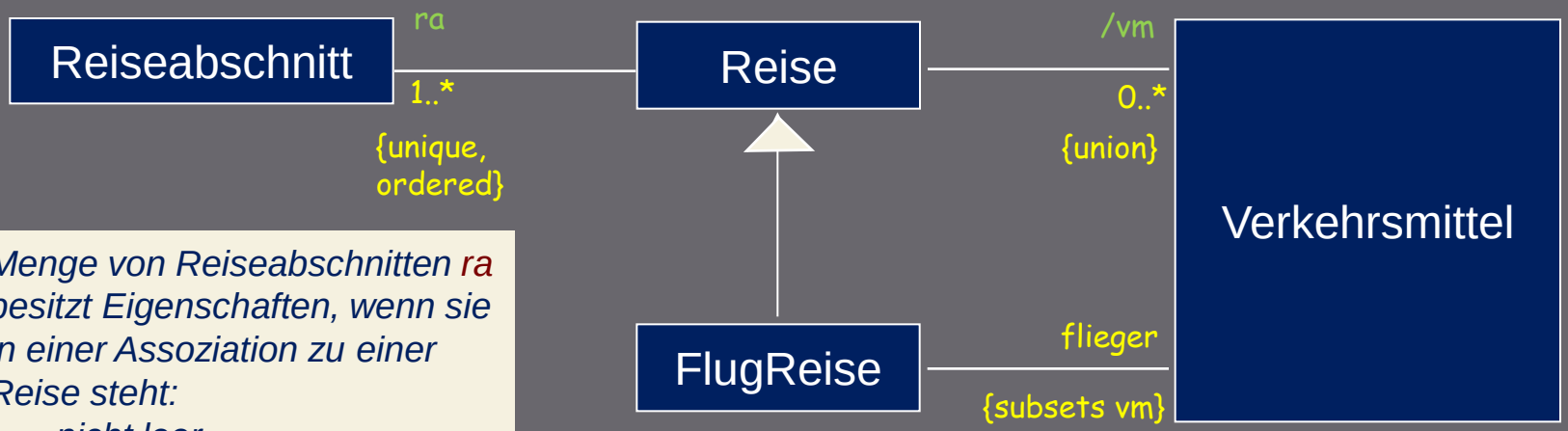
falls Assoziationsenden nicht benannt worden sind

Eigenschaften: Beispiel-1



Eigenschaften: Beispiel-2

Menge von Verkehrsmitteln
kann leer sein
unter vm ist diese Menge einem Reise-Objekt bekannt



Menge von Reiseabschnitten ra besitzt Eigenschaften, wenn sie in einer Assoziation zu einer Reise steht:

- *nicht leer*
- *echte Menge*
- *es gibt eine Ordnung*

Es wird eine Teilmenge von Verkehrsmitteln ausgezeichnet

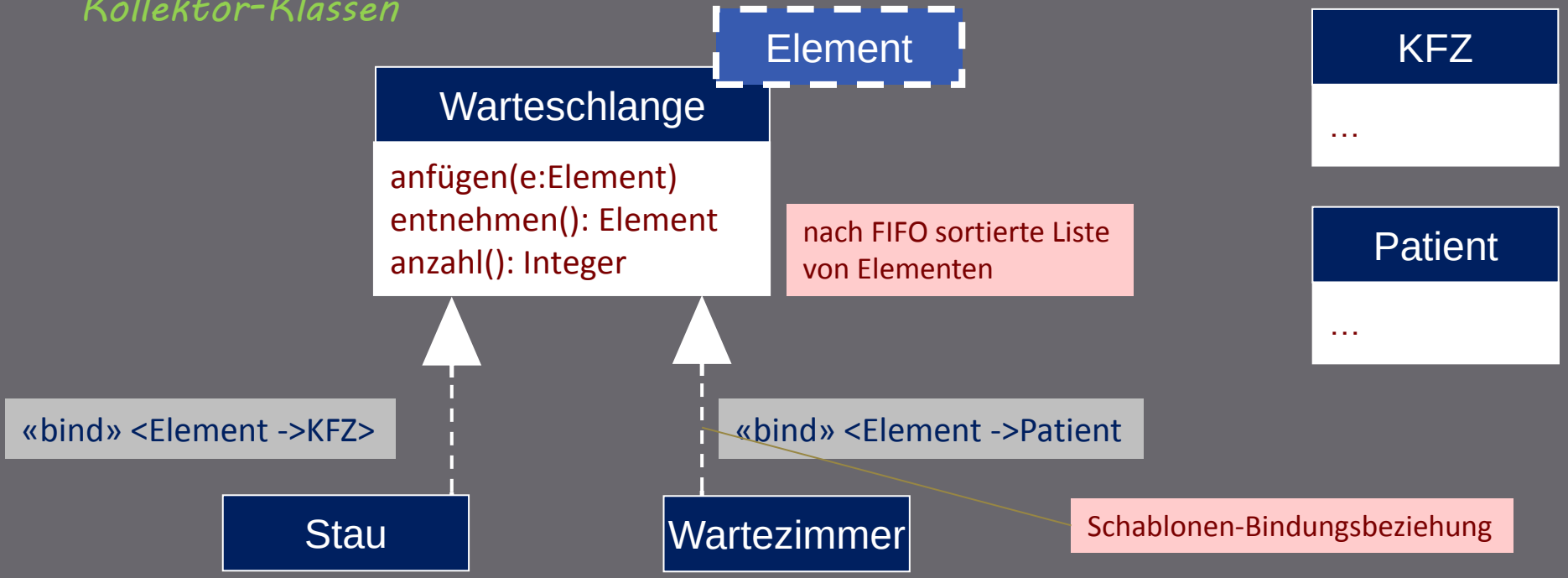
unter flieger ist diese Menge einem Reise-Objekt bekannt, wenn es sich dabei um ein Flugreise-Objekt handelt

ist eine Reise eine Flugreise, so sind alle Verkehrsmittel über flieger erreichbar
vm ist eine abgeleitete Größe

Parametrisierte Klassen / Templates

typischer Anwendungsfall sind Kollektor-Klassen

bereits existierende Klassen



Resultat der Bindung: homogene Listen von KFZ-Objekten bzw. Patient-Objekten

st: Stau

wz: Wartezimer

FRAGE: Wie lässt sich eine heterogene Liste definieren ?

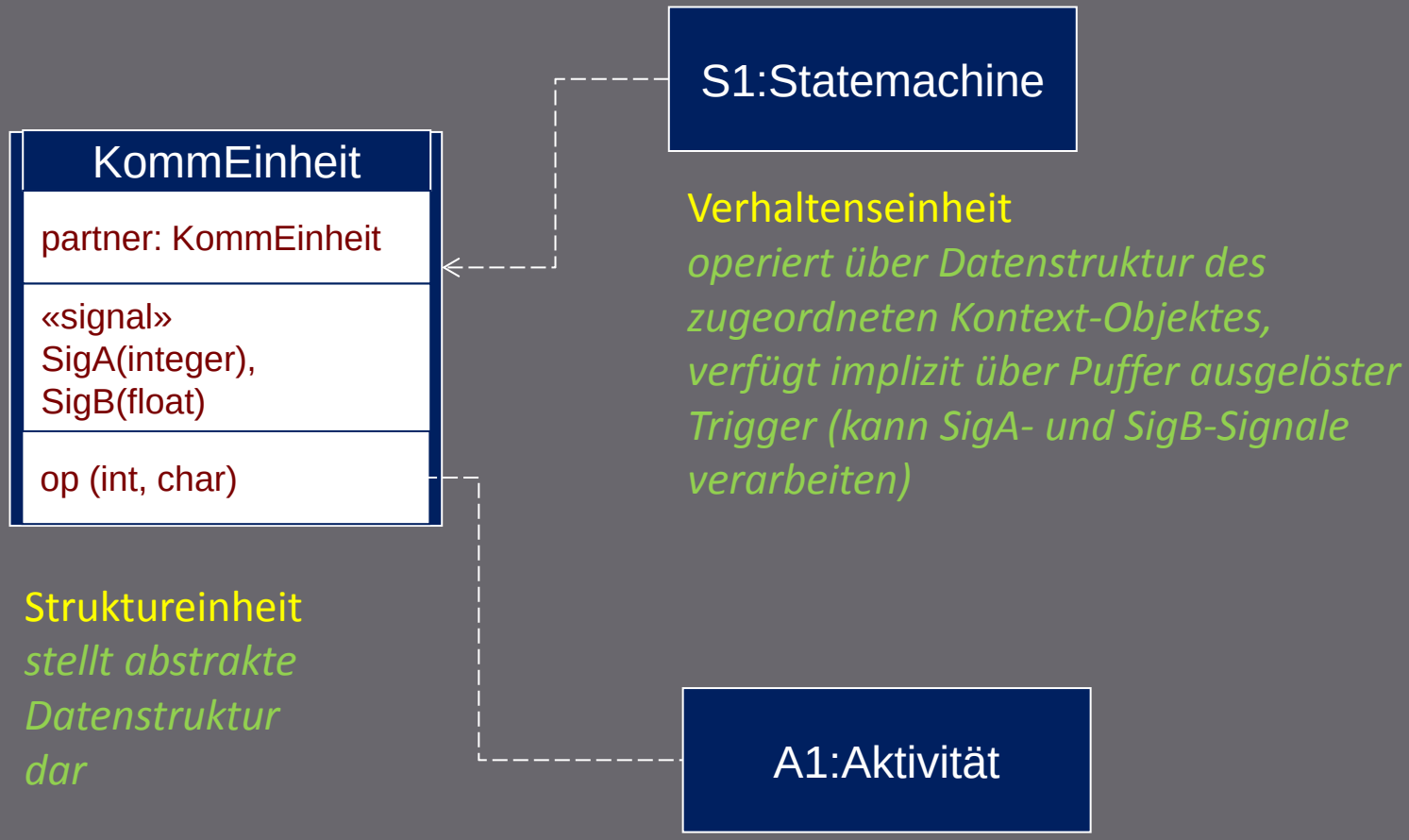
Aktive Klassen

aktive Klasse

- … ist eine **Klasse**,
  bei deren Instanziierung als direkte Konsequenz das für die Klasse beschriebene **Verhalten** startet und nicht aufhört,
  bis entweder
  - das Verhalten zu Ende geht (**interner Stop**) oder
  - das Objektverhalten von einem anderen terminiert wird (**externer Abbruch**)

- Zustandsautomaten bieten sich als Verhaltensbeschreibung an

Klassen – Aktivitäten - Zustandsmaschinen



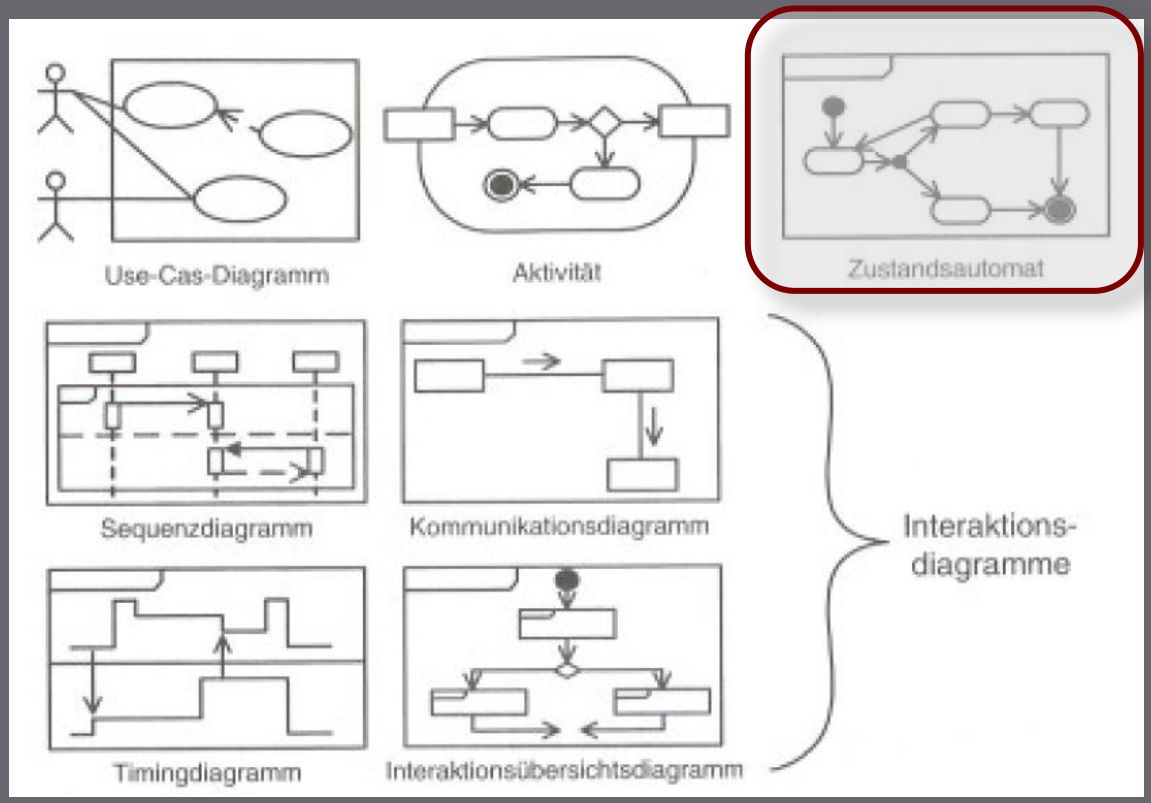
Verhaltenseinheit
operiert über Datenstruktur des zugeordneten Kontext-Objektes, verfügt implizit über Puffer ausgelöster Trigger (kann SigA- und SigB-Signale verarbeiten)

Struktureinheit
stellt abstrakte Datenstruktur dar

Verhaltenseinheit
beschreibt Verhalten von op, operiert über Datenstruktur des zugeordneten Kontext-Objektes

Zustandsautomaten, Allgemeines

- breitestes Anwendungsspektrum



Factory-Template

- häufiges Muster

Generator als Akteur

erzeugt ständig mit zeitlichem Abstand Objekte

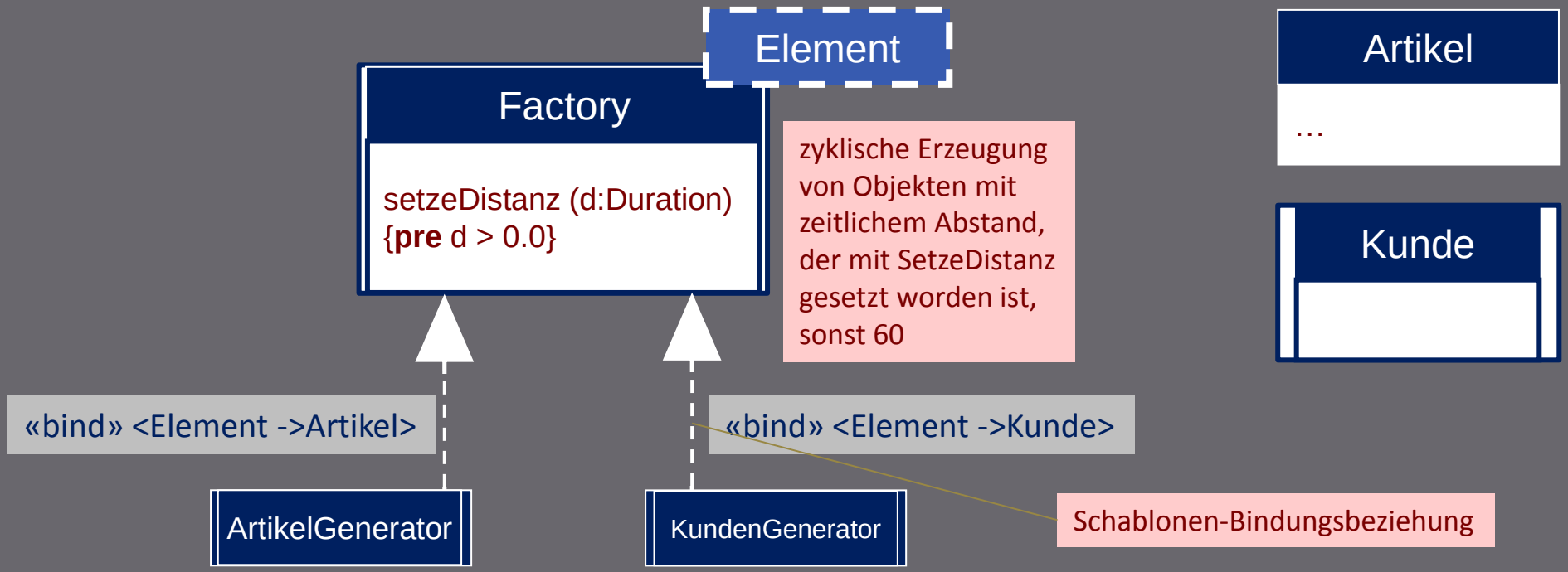
z.B.: Versandartikel (als Objekte passiver Klassen)

oder

Kunden mit eigenem Lebenslauf (als Objekte aktiver Klassen)

Parametrisierte aktive Klassen

bereits existierende Klassen



FRAGE: Wie unterscheiden sich die erzeugten Ströme von Objekten aktiver und passiver Klassen?

kg: KundenGenerator

FRAGE: Wie komme ich zu zwei unabhängigen Kundenströmen?

Inhalt

- **Teil A**
  Aspekte von Modellierung und Simulation dynamischer Systeme
- **Teil B**
  Die Modellierungssprache UML
- **Teil C**
  Die ausführbare Modellierungssprache SLX
- **Teil D**
  Modellierung von Lieferketten

- **B.1**
  Wozu UML im Kontext der Computersimulation?
- **B.2**
  UML-Teilsprachen, Sprachkonzepte
- **B.3**
  Klassendiagramme
- **B.4**
  Verhaltensbeschreibung
- **B.5**
  OCL

Erweiterter Endlicher Zustandsautomat

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

A) Erweiterung eines Endlichen Zustandsautomaten
- lokale Variablen
- Lokale Timer
- Empfangs-Pool von Nachrichten (asynchrone Komm.)
- Trigger (Zustandsübergangsauslöser)
- Potentielle Empfangsmenge (Nachrichten/Remote-Op-Rufe)
- Guards (als Bedingungen über lokale/globale Zustandsgrößen)

B) Zustandserweiterung
- Eintrittsaktionen
- Do-Aktivität
- Austrittsaktionen

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

Im Kontext einer **aktiven Klasse**