ModSoft

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

Teil II b: Strukturmodellierung: Klasssen,…

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

fischer@informatik.hu-berlin.de

Inhalt

Teil I- Einführung

Teil II-Struktur

- Klassen, Assoziationen, Abhängigkeiten
- Interface, Datentypen, Signale, Port,
- Strukturierter Classifier
- Aktive Klassen

a)

- Präzisierungen
- Klassendiagramm, vertiefende Betrachtung

b)

Teil III-Verhalten

a)

- Einfacher Zustandsautomat
- Beispiel: DemonGame
- UML-Modellierungsdefizite

Wechsel Teil III → Teil II

Korrektur: SDL– in UML-Notation (letzte Vorlesung)

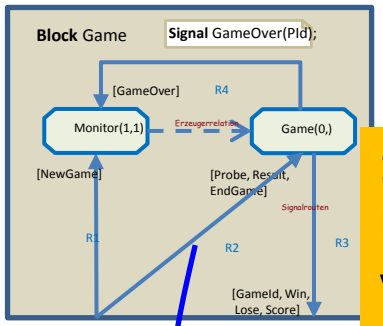
~ SDL

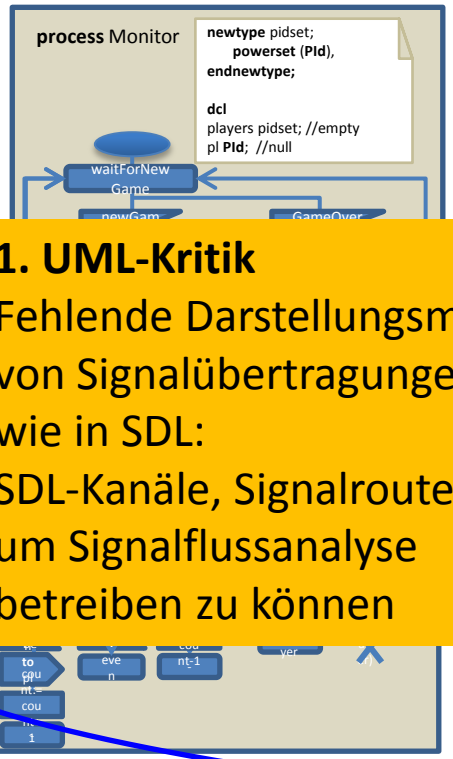
~ UML

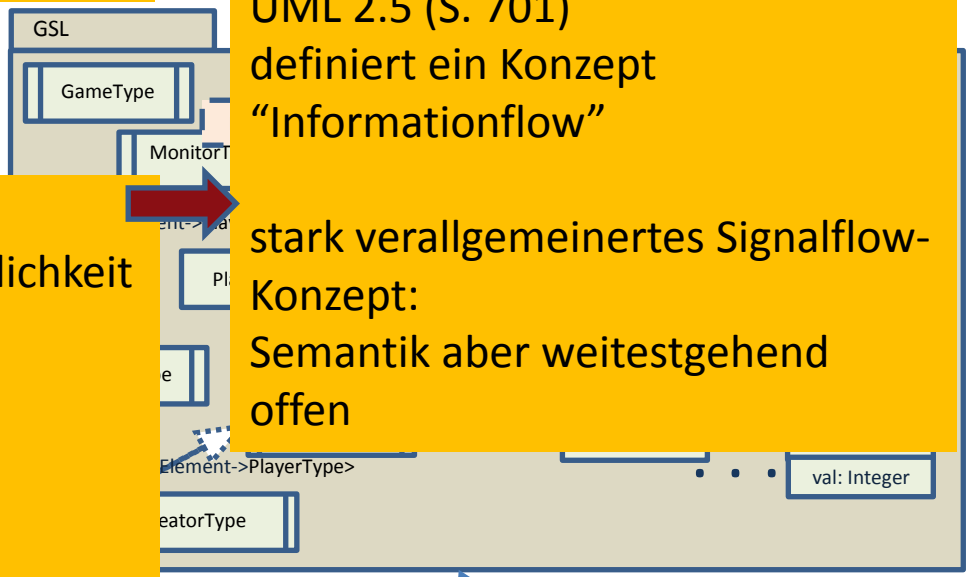
… wird zurückgenommen
UML 2.5 (S. 701)
definiert ein Konzept
“Informationflow”

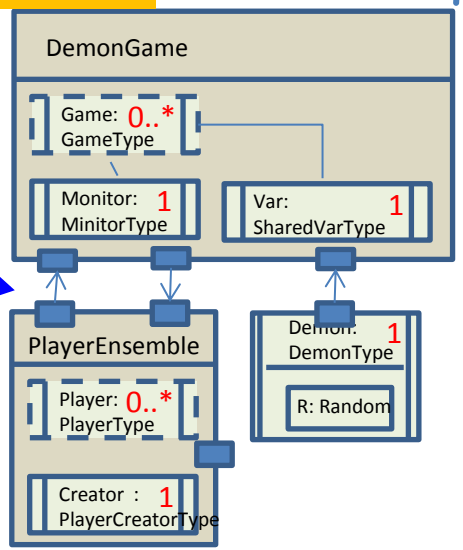
stark verallgemeinertes Signalflow-Konzept:
Semantik aber weitestgehend offen

1. UML-Kritik
Fehlende Darstellungsmöglichkeit von Signalübertragungen wie in SDL:
SDL-Kanäle, Signalrouten
um Signalflussanalyse betreiben zu können

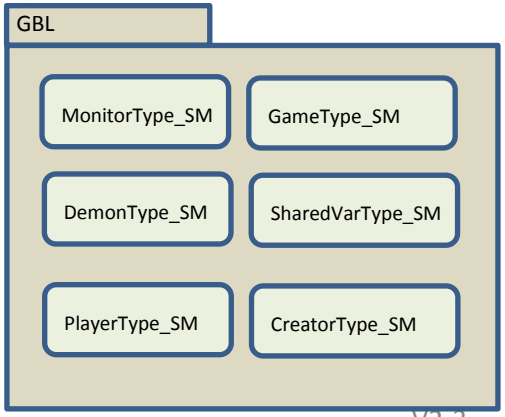




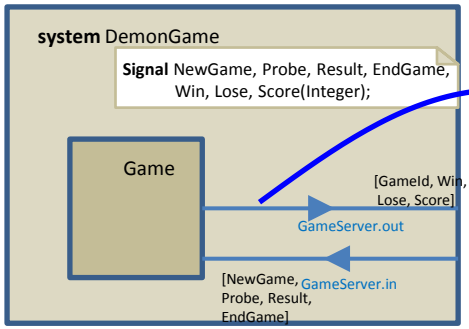




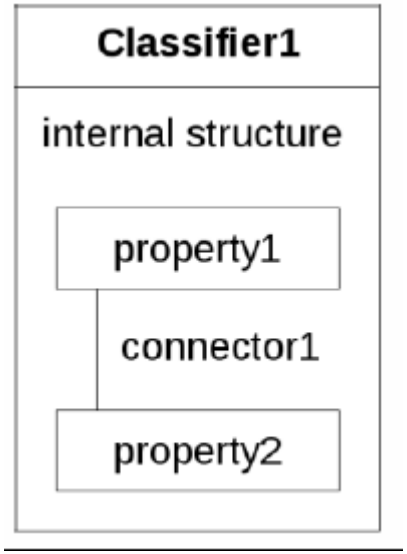
GameBehaviourLib



DemonGame



Problem



unklar:
Signalflüsse zwischen Parts

~

Verbindung zu Ports

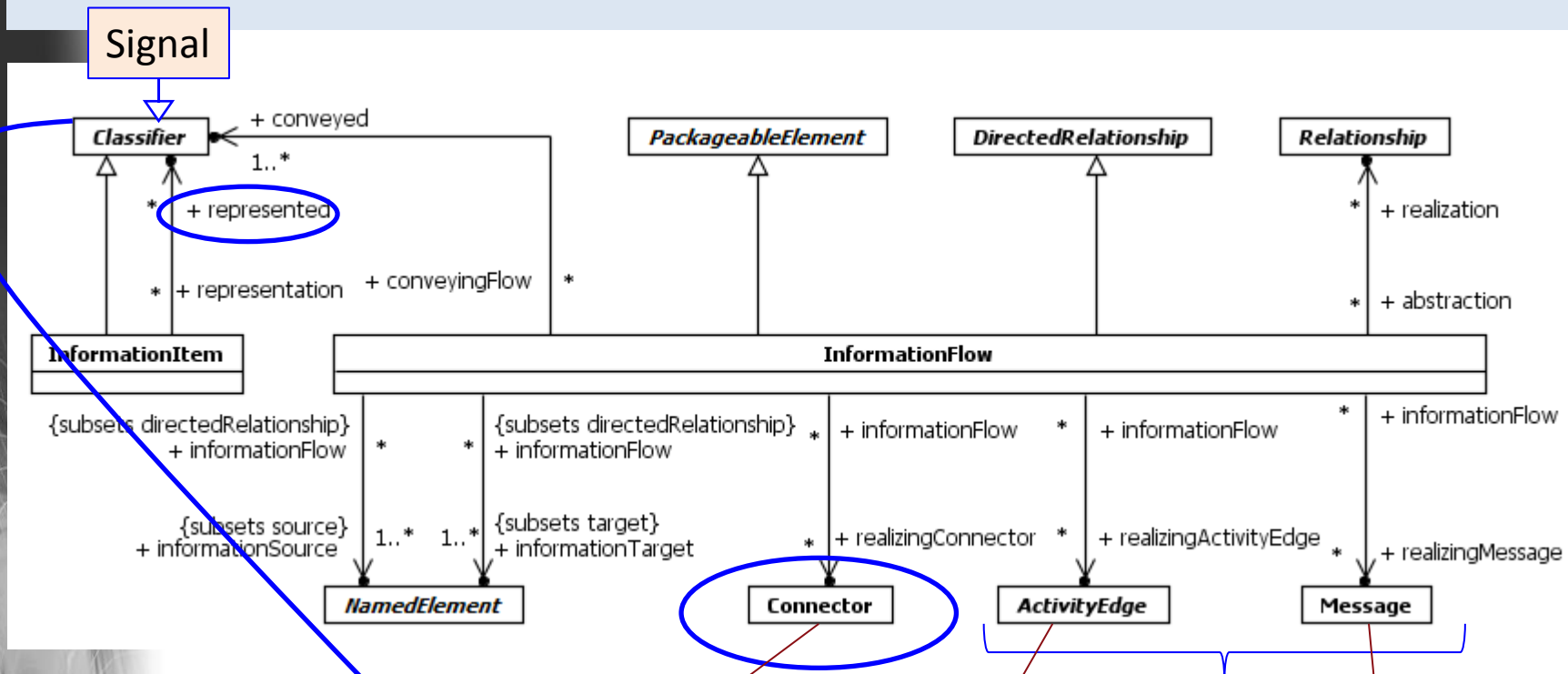
~

Verbindung
Port-zu- Port

InformationFlow im UML-Metamodell

- The **InformationFlows** package supports exchange of information between system entities at high levels of abstraction.
- **InformationFlows** may be useful during top-down model development, representing aspects of models not yet fully specified, and for recording less detailed, heuristic representations of more complex model areas.
- In these ways, **InformationFlows** can help to clarify and document overall understanding of the intent of large or complicated models.
- **InformationFlows** describe circulation of information through a system in a general manner.
- They do not specify the nature of the information, mechanisms by which it is conveyed, sequences of exchange, or any control conditions.
- During more detailed modeling, representation and realization links may be added to specify which model elements implement an **InformationFlow** and to show how information is conveyed.
- Similarly, **InformationItems** can be used to represent the information that flows along InformationFlows even before details of their realization have been designed.

InformationFlow im UML-Metamodell

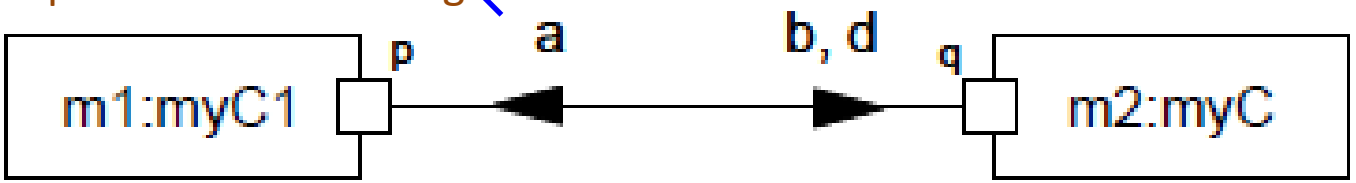


bereits in früheren UML-Versionen

Bedeutung für Aktivitätsdiagramme

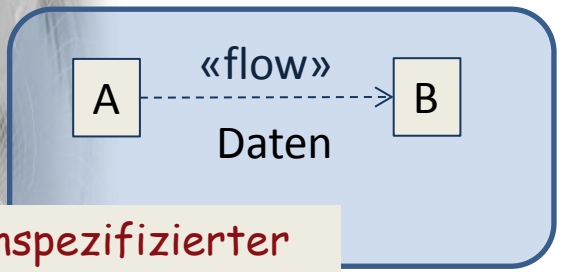
Bedeutung für SequenzDiagramme

Bedeutung für KompositionsstrukturDiagramme

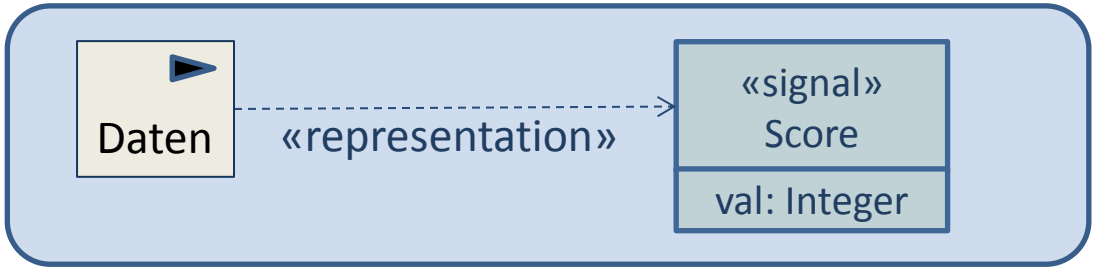


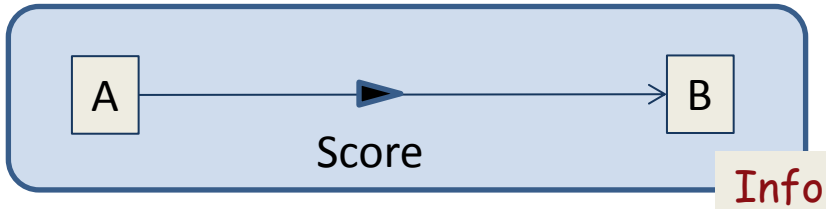
InformationFlow im UML-Metamodell

- Quelle und Senke von Informationsflüssen können Klassen sein (aber auch Akteure, Knoten, Anwendungsfälle, …)
- Als Kanäle kommen Assoziationen, Beziehungen, Konnektoren (!!!), Aktivitätskanten, … in Frage
- Die ausgetauschten Daten können abstrakte Informationselemente (ohne nähere Spezifikation von Eigenschaften oder Vererbungsmöglichkeiten) oder Classifier-Ausprägungen jeglicher Art sein: Klassen, Signale, … Verhalten



Unspezifizierter Informationsfluss





Informationsfluss über einen Kanal

Einführung weiterer Linientypen (Schlüsselwort-Angaben)

Suche in der Tabelle definierter UML-Schlüsselworte …

Keyword	Metamodel Element	Semantics	Notation Placement
abstraction	Abstraction	Abstraction	dashed-line label
access	PackageImport	visibility <> VisibilityKind::public	dashed-line label
activity	Activity	Activity	box header
actor	Actor	Actor	box header
apply	ProfileApplication	ProfileApplication	dashed-line label
artifact	Artifact	Artifact	box header
centralBuffer	CentralBufferNode	CentralBufferNode	box header
bind	TemplateBinding	TemplateBinding	dashed-line label
collaboration	Collaboration	Collaboration	box header
component	Component	Component	box header
datastore	DataStoreNode	DataStoreNode	box header
dataType	DataType	DataType	box header
decisionInput	Behavior	DecisionNode::decisionInput	note label
decisionInputFlow	ObjectFlow	DecisionNode::decisionInputFlow	line label
deploy	Deployment	Deployment	dashed-line label
deployment spec	DeploymentSpecification	DeploymentSpecification	box header
device	Device	Device	box header
element access	ElementImport	visibility <> VisibilityKind::public	dashed-line label
element import	ElementImport	visibility = VisibilityKind::public	dashed-line label
enumeration	Enumeration	Enumeration	box header
executionEnvironment	ExecutionEnvironment	ExecutionEnvironment	box header
extend	Extend	Extend	dashed-line label
extended	Region	extendedRegion->notEmpty()	after name
extended	StateMachine	extendedStateMachine->notEmpty()	after name
external	ActivityPartition	isExternal = true	swimlane header

Einführung weiterer Linientypen (Schlüsselwort-Angaben)

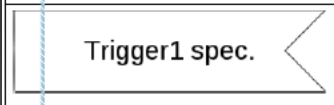
Suche in der Tabelle definierter UML-Schlüsselworte …

final	State	State::isLeaf	After name
flow	InformationFlow	InformationFlow	dashed-line label
import	PackageImport	visibility = VisibilityKind::public	dashed-line label
include	Include	Include	dashed-line label
information	InformationItem	InformationItem	box header
interface	Interface	Interface	box header
iterative	ExpansionRegion	mode = ExpansionKind::iterative	top left corner
localPostcondition	Constraint	Action::localPostcondition	box header
localPrecondition	Constraint	Action::localPrecondition	box header
manifest	Manifestation	Manifestation	dashed-line label
merge	PackageMerge	PackageMerge	dashed-line label
model	Model	Model	box header
multicast	Obj		
multireceive	Obj		
occurrence	Coll		
parallel	Exp		
postcondition	Constraint	Behavior::postcondition	box header
protocol	ProtocolStateMachine	ProtocolStateMachine	after name / box header
precondition	Constraint	Behavior::precondition	box header
primitive	PrimitiveType	PrimitiveType	box header
profile	Profile	Profile	box header
reference	ElementImport	Profile::metaclassReference	dashed-line label
reference	PackageImport	Profile::metamodelReference	dashed-line label
representation	InformationItem::represented	represented	dashed-line label
selection	Behavior	ObjectFlow::selection	note label
signal	Signal	Signal	box header
singleExecution	Activity	isSingleExecution = true	inside box
statemachine	StateMachine	StateMachine	box header
stream	ExpansionRegion	mode = ExpansionKind::stream	top left corner
Stereotype	Stereotype	Stereotype	box header
strict	ProfileApplication	isStrict = true	dashed-line label
structured	StructuredActivityNode	StructuredActivityNode	box header
substitute	Substitution	Substitution	dashed-line label
transformation	Behavior	ObjectFlow::transformation	note label
use	Usage	Usage	dashed-line label

	UMLEdge (InformationFlow) - UMLKeywordLabel (InformationFlow) - UMLShape (InformationFlow, isIcon=true)	20.1.4, B.5 Fig. 20.2, 20.5, 20.6

Senden von Signalen

klar:
Signal-Trigger

Notation	Diagram elements	Ref.
	UMLShape (Trigger, in StateMachine diagrams when isTransitionOriented=true) - UMLNameLabel (Trigger)	14.2.4, B.4.2 Fig. 14.32
	UMLShape (SendSignalAction, in StateMachine diagrams when isTransitionOriented=true) - UMLNameLabel (SendSignalAction)	14.2.4, B.4.2 Fig. 14.32

2. Unzulässige UML-Kritik
Fehlende Darstellungsmöglichkeit für das Senden von Signalen

Offen bleibt:
Signal-Adressierung

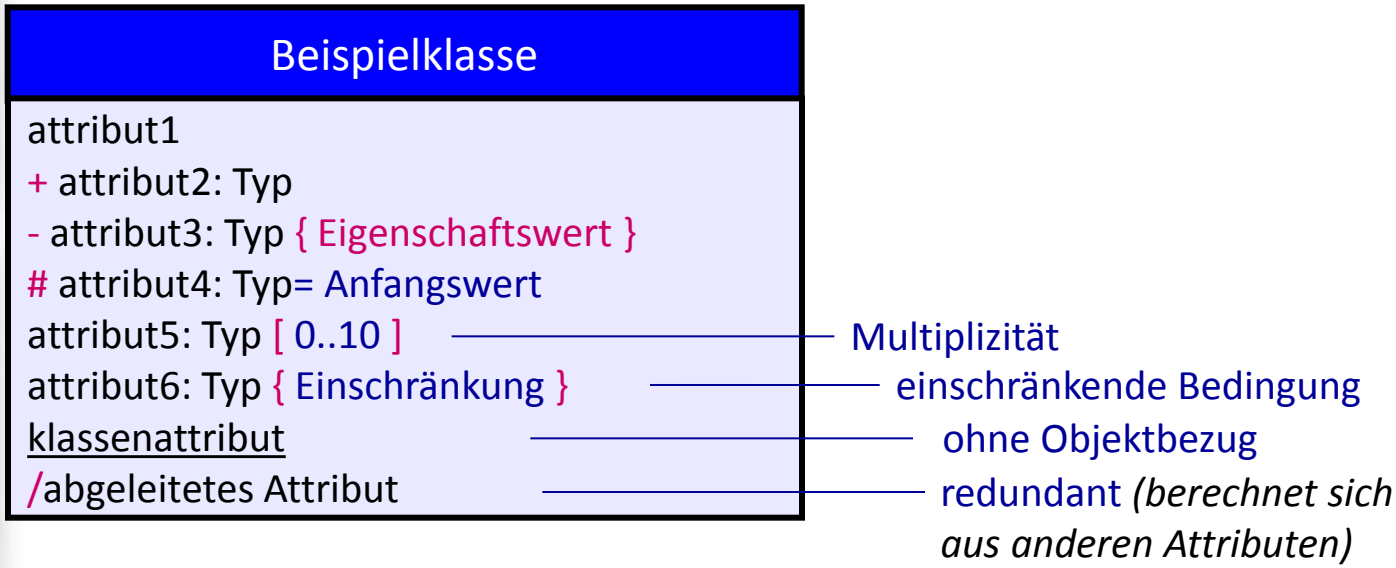
Klassendiagramme - eine vertiefende Betrachtung

1. Attribute, Assoziationen und Links
2. Operationen und Methoden

Einheitliche Betrachtung: Attribute und Rollen

Bestandeile

Name, Sichtbarkeit, Typ, Anfangswert, Multiplizität, Eigenschaft, Einschränkung



Falls Attribut vom Typ Classifier und Multipliziät > 1
charakterisieren Eigenschaftswerte
unique, ordered, seq, bag
die Werte (Kollektion von Objekten) näher

In diesem Fall
Attribute sind äquivalent beschreibbar als navigierbare Assoziationen mit Rolleneigenschaften

Grundsätzliches zu Einschränkungen

- jedem Modellelement dürfen beliebig viele Einschränkungen zugeordnet werden
- vordefinierte Standardeinschränkungen
  (aus dem Metamodell)
- nutzerdefinierte Einschränkungen
- Beschreibung:
  - OCL, natürliche Sprache, beliebige Notation
- Syntax
  - {…} oder
  - { <name der Einschränkung>: … }

Zusammenhang von Attribut und Assoziation (Wdh.)

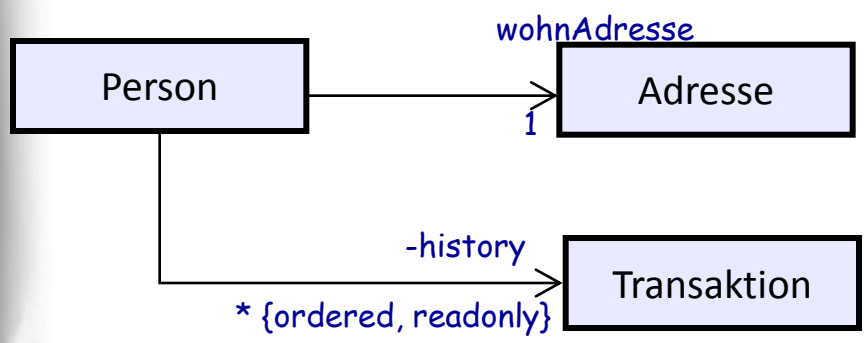
- UML 2 kennt **keinen** semantischen Unterschied zwischen
  - Attributen einer Klasse und
  - navigierbaren Assoziationsenden (Rollen),

  } *abstraktes UML-Konzept:* Property

- Ist das Ende einer Assoziation als **navigierbar** gekennzeichnet, so besitzt die navigierende Klasse ein Attribut mit dem Namen dieses Endes (Rollenname)

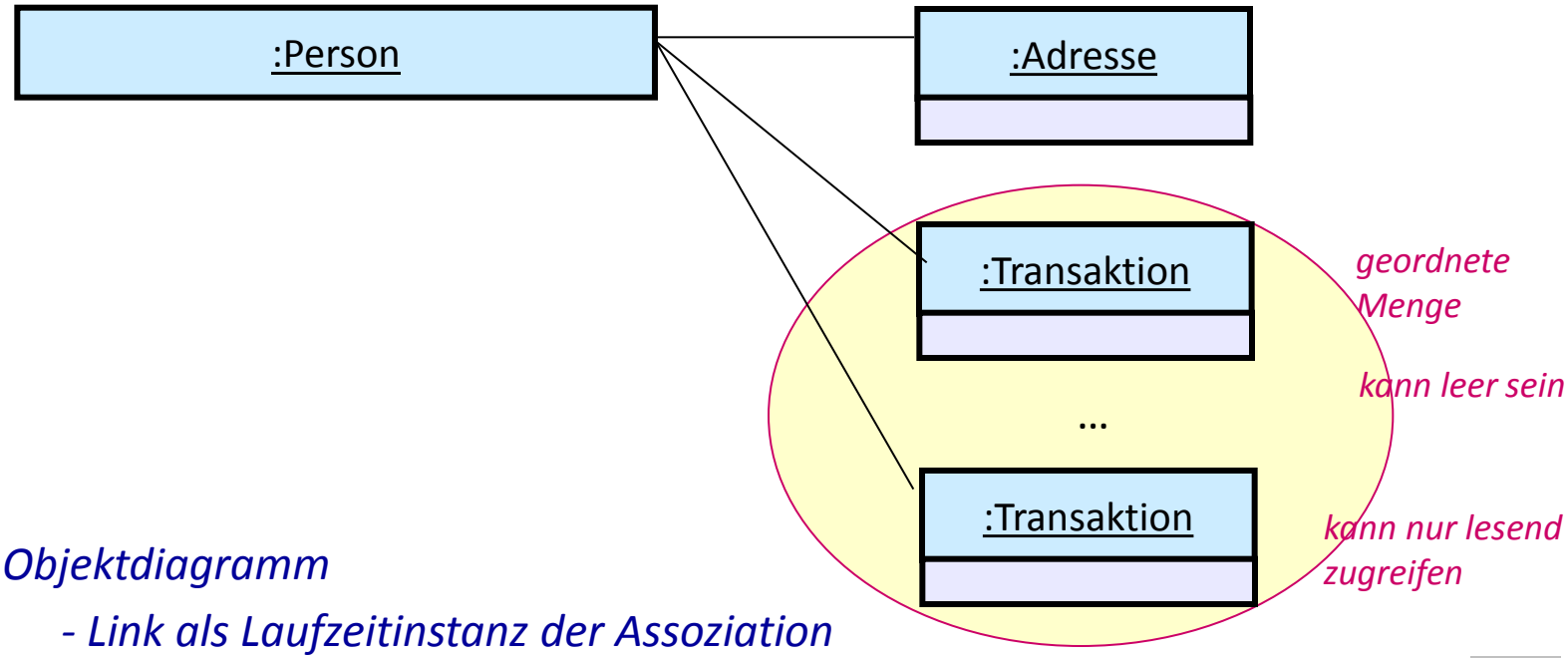
- Ist das Ende einer Assoziation als **nicht navigierbar** gekennzeichnet, so besitzt die Assoziation (Instanz einer Metaklasse) selbst eine Ausprägung mit dem Namen des Endes
  (so als Repräsentation im Repository)

Zusammenhang: Attribut und Assoziationsende



Assoziationsende mit Rollenname

Klassendiagramm



geordnete Menge

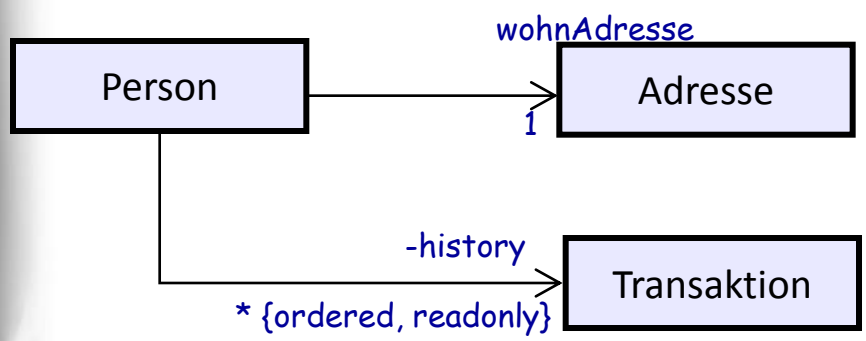
kann leer sein

kann nur lesend zugreifen

Objektdiagramm

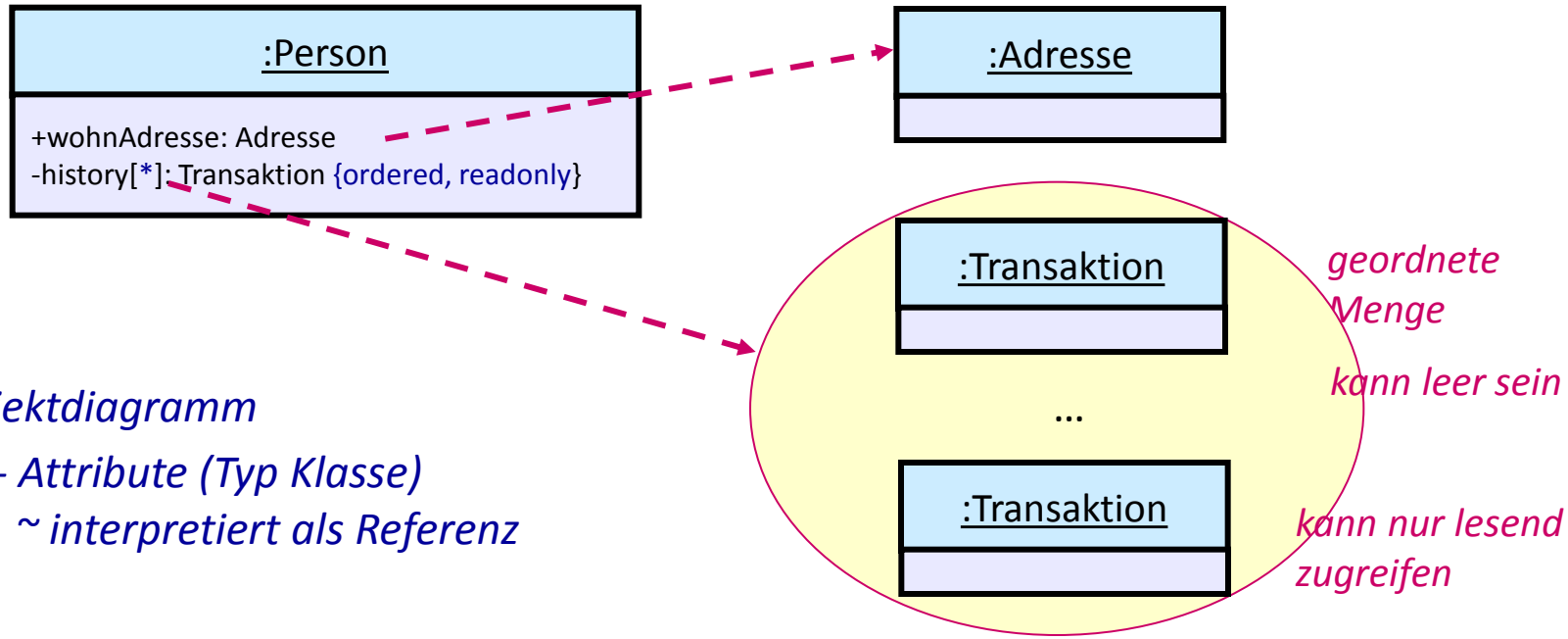
- Link als Laufzeitinstanz der Assoziation

Zusammenhang: Attribut und Assoziationsende



Assoziationsende mit Rollenname

Klassendiagramm



geordnete Menge

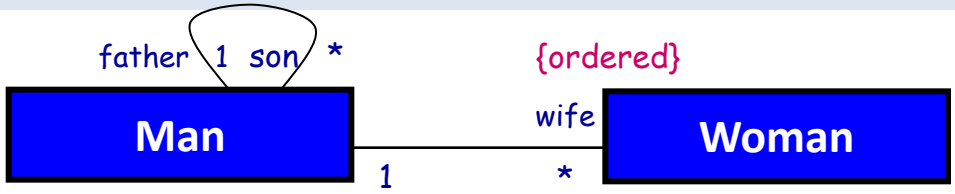
kann leer sein

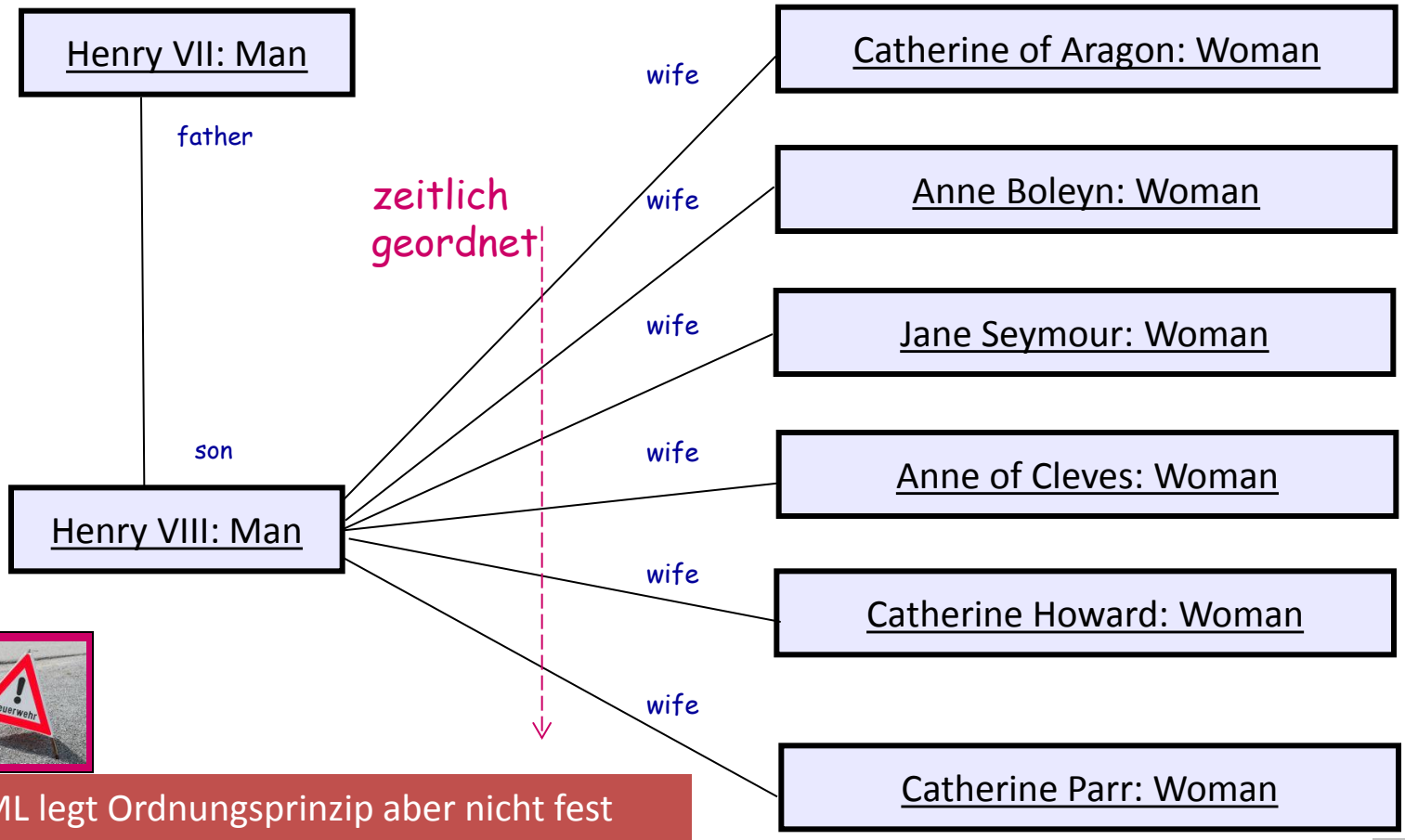
kann nur lesend zugreifen

Objektdiagramm

- *Attribute (Typ Klasse)*
  - *~ interpretiert als Referenz*

Geordnete Links





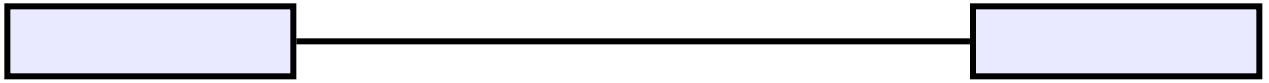
UML legt Ordnungsprinzip aber nicht fest

Navigierbarkeit (Wdh.)

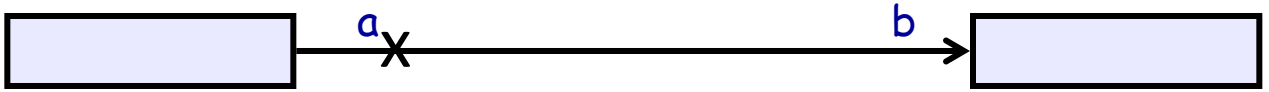
- zwei navigierbare Enden

- zwei nicht navigierbare Enden

- Navigierbarkeit nicht ausgeschlossen



- Kombinationen



Eigenschaftswerte von Attributen / Rollen

Liste möglicher Eigenschaften

z.T. mit Voreinstellung und kombinierbar

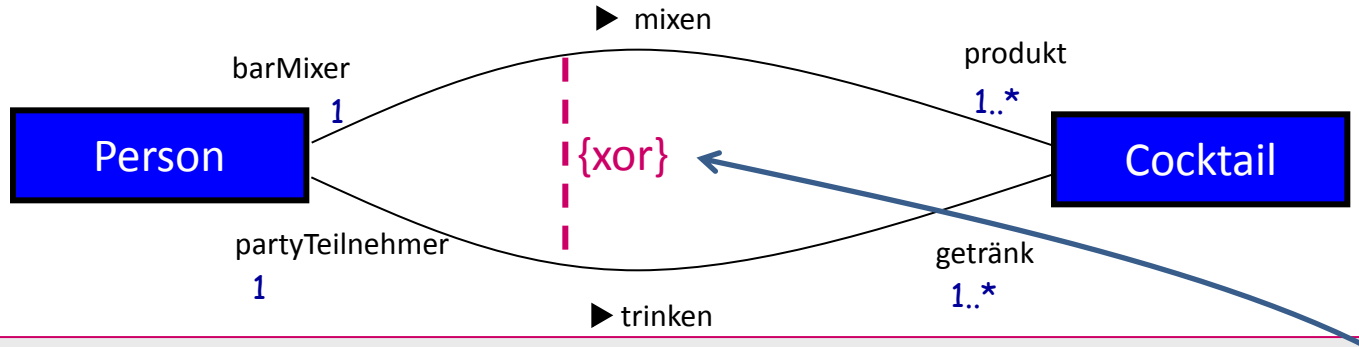
– [/] (abgeleitetes Attribut)	:	Verwendungsabweichung (Präfix-Notation)
– [readonly | unrestricted]	:	readonly oder nicht → CONST
– [composite | shared | none]	:	zusammengesetzt (Typ von Aggregation)
– (redefines *Attr*)*	:	redefiniert das ererbte Attribut *Attr*
– (subsets *Attr*)*	:	ist Teilmenge des ererbten *Attr*
– [union]	:	bildet Vereinigung
– [unique | nonunique]	:	einziges oder mehrfaches Vorkommen
– [ordered | unordered]	:	geordnet oder ungeordnet
– seq	:	Folge/Liste
– bag	:	Multimenge

heutige Vorlesung

… - Voreinstellung

Rollen, Multipliziäten, Leserichtung

damit kann leider auch ein gemixter Cocktail niemals getrunken werden

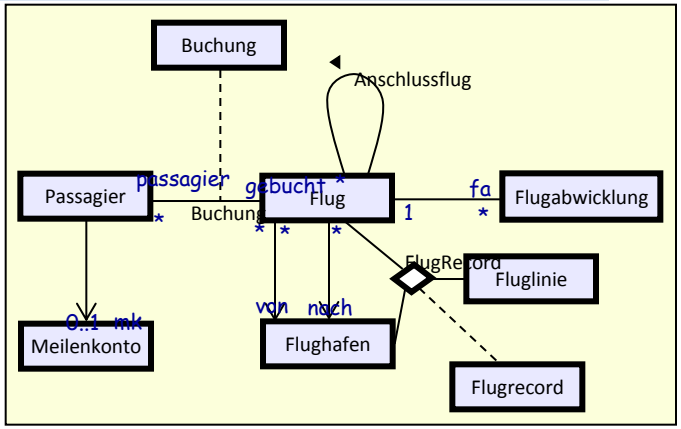


- Ein barMixer (max. ein Person-Objekt) mixt endlich viele Cocktails (mind. einen) zu. Diese bilden eine Kollektion produkt. Diese Kollektion von Cocktail-Objekten ist nicht leer.
- Ein beliebiger Cocktail (als produkt-Element) wird genau von einer Person in der Rolle des barMixers zubereitet.
- Jede Person in der Rolle eines partyTeilnehmers trinkt endlich viele Cocktails (mindestens einen)
- Jeder getrunkene Cocktail wird von genau einer Person getrunken

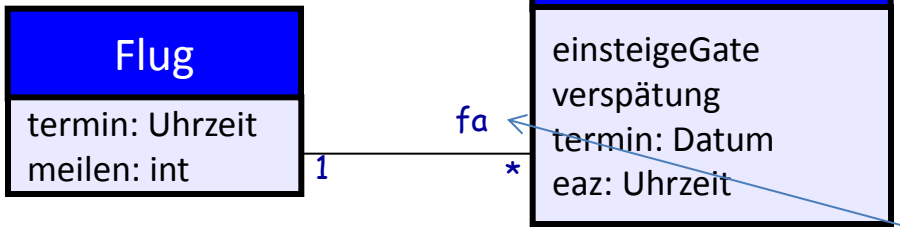
Offene nützliche Präzisierungen:

- Es sollten nicht mehr Cocktails getrunken werden können, wie vom barMixer zubereitet.
- Kann die Person in der Rolle des Barmixers auch die eines Gastes annehmen ?
- Anzahl von Partyteilnehmern

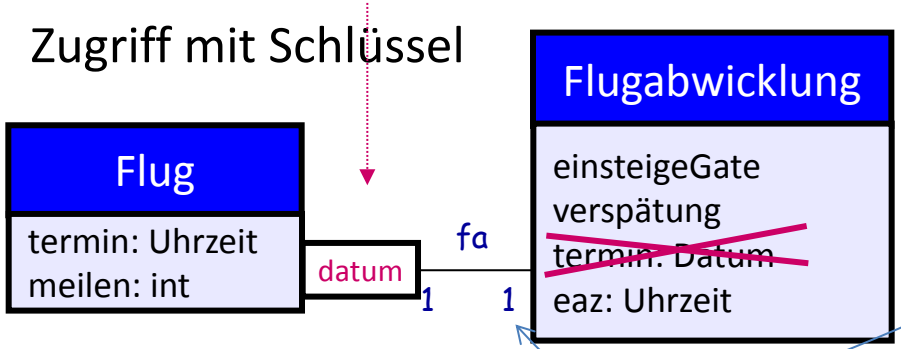
Qualifizierte Assoziationen



- Zugriff ohne Schlüssel



- Zugriff mit Schlüssel



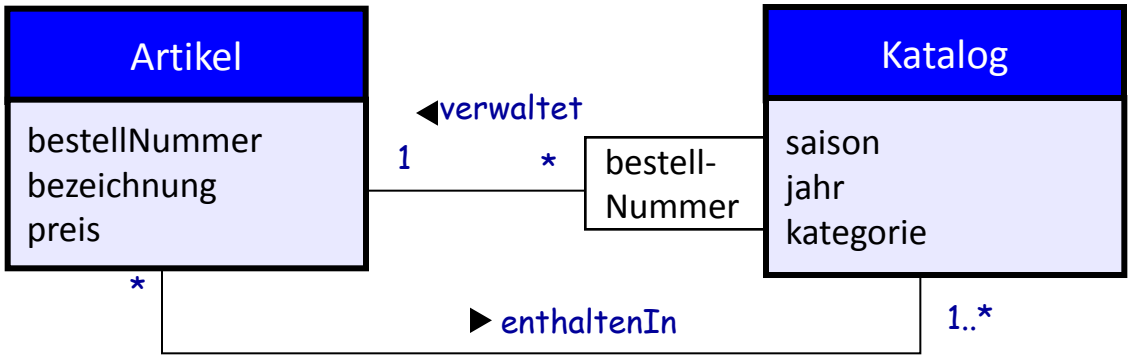
Die referenzierte Objektmenge wird durch qualifizierende Attribute in Partionen unterteilt

Festlegung der Anzahl von Elementen je Partition

…wird eingesetzt, um die Vielfachheit einer Assoziation zu reduzieren

Beispiel: unterschiedliche Aussagen

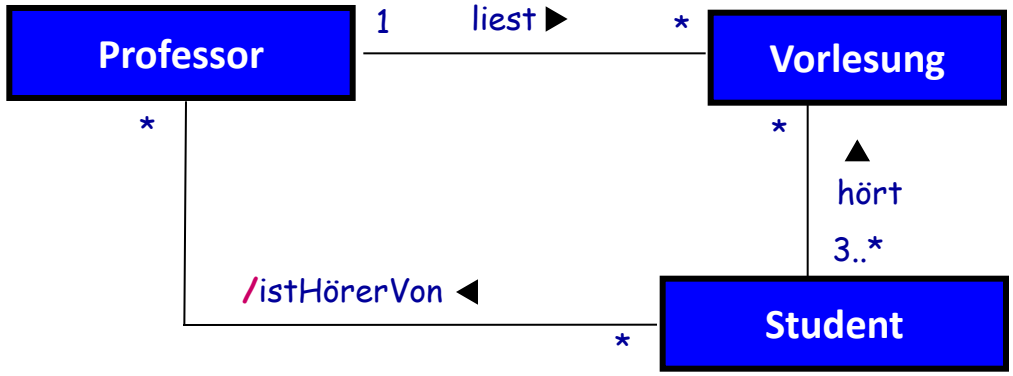
- innerhalb eines Kataloges bezeichnet eine Bestellnummer genau einen Artikel



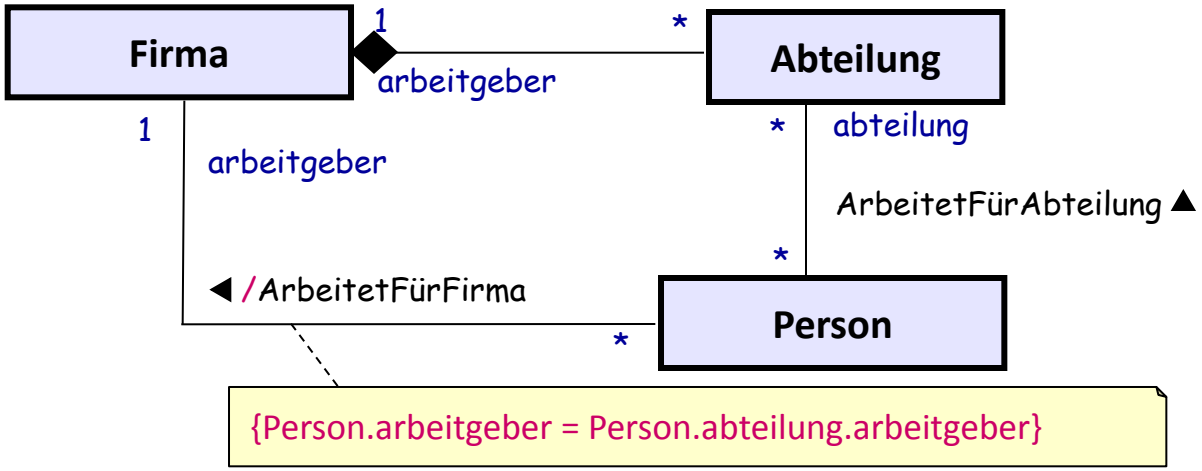
- ein Katalog enthält verschiedene Artikel
  ein Artikel kann in verschiedenen Katalogen enthalten sein

Abgeleitete Assoziationen

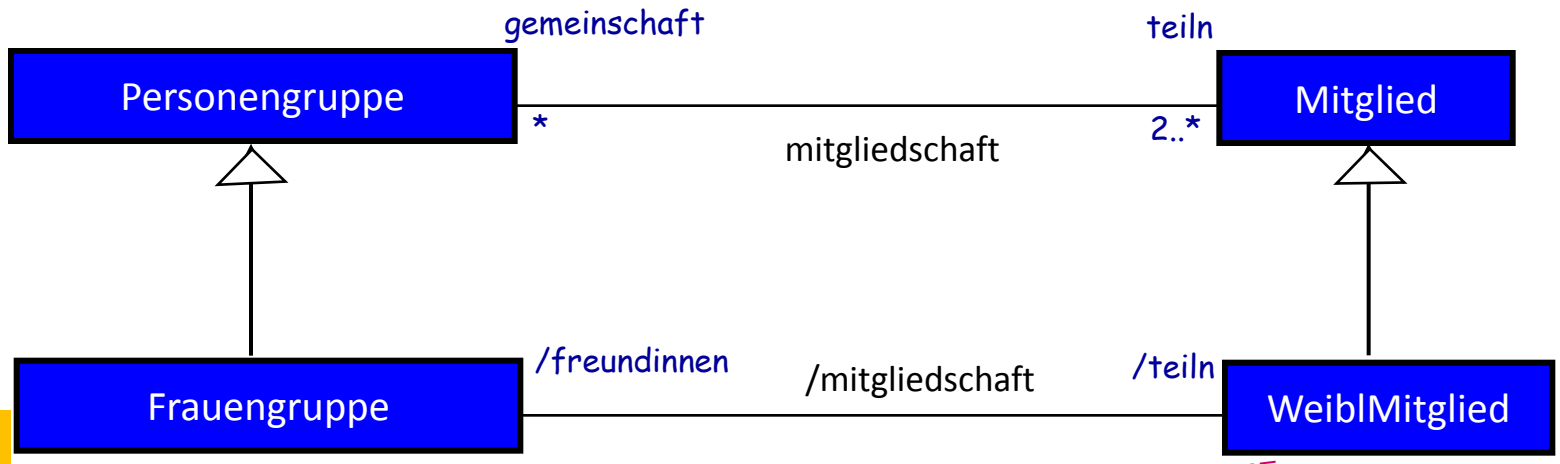
- Eine Assoziation heißt **abgeleitet** (derived association),
  wenn die Abhängigkeiten bereits durch andere Assoziationen beschrieben worden sind
  (es kommen keine neuen Informationen hinzu,
  erhöhen aber u.U. die Klarheit des Entwurfs



Weiteres Beispiel



/-Eigenschaft im Zusammenhang mit Spezialisierung



OCL-Regel: kann das formal ausdrücken

{ **context** WeiblMitglied:
freundinnen ergibt sich aus Elementen von gemeinschaft,
wobei gilt, dass diese Objekte ebenfalls von der Klasse WeiblMitglied sind}

- Assoziationen werden vererbt (und müssen nicht erneut aufgeführt werden)
  gilt auch für Rollennamen (Attribut-Dualität)

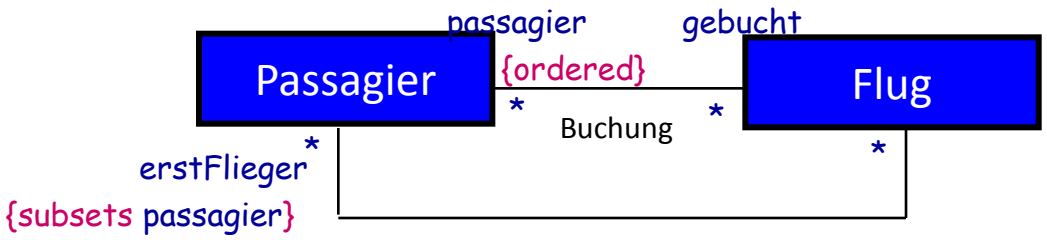
- falls jedoch erneute Angabe, dann **muss** /-Kennung erfolgen;
  Rollennamen dürfen dabei neu vergeben werden (ebenfalls mit /-Kennung)
  Sie sind zusätzliche Attribute, deren Werte sich berechnen
  (**keine Wertzuweisung** möglich!!!)

Teilmengen von Objektbeziehungen

- Assoziationsende mit **spezifischer** Attribut-/Rolleneigenschaftsangabe zur Präzisierung der Semantik von Assoziationen

Teilmenge:

– {subsets <Attribut> | <Rollenname> }
Teilmenge von Objektbeziehungen

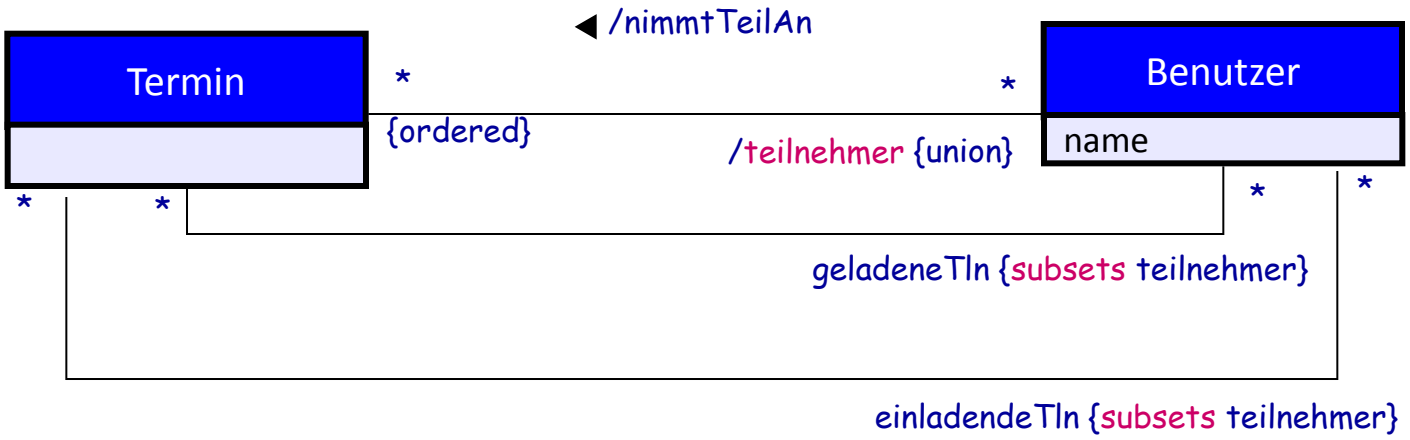


Wert des Attributes erstFlieger ist eine **Teilmenge** des Attributwertes passagier

Kollektion von
Passagier-Objekten
(mit Erstflieger-Eigenschaft)

geordnete Kollektion von
Passagier-Objekten,
die ein und denselben Flug gebucht haben

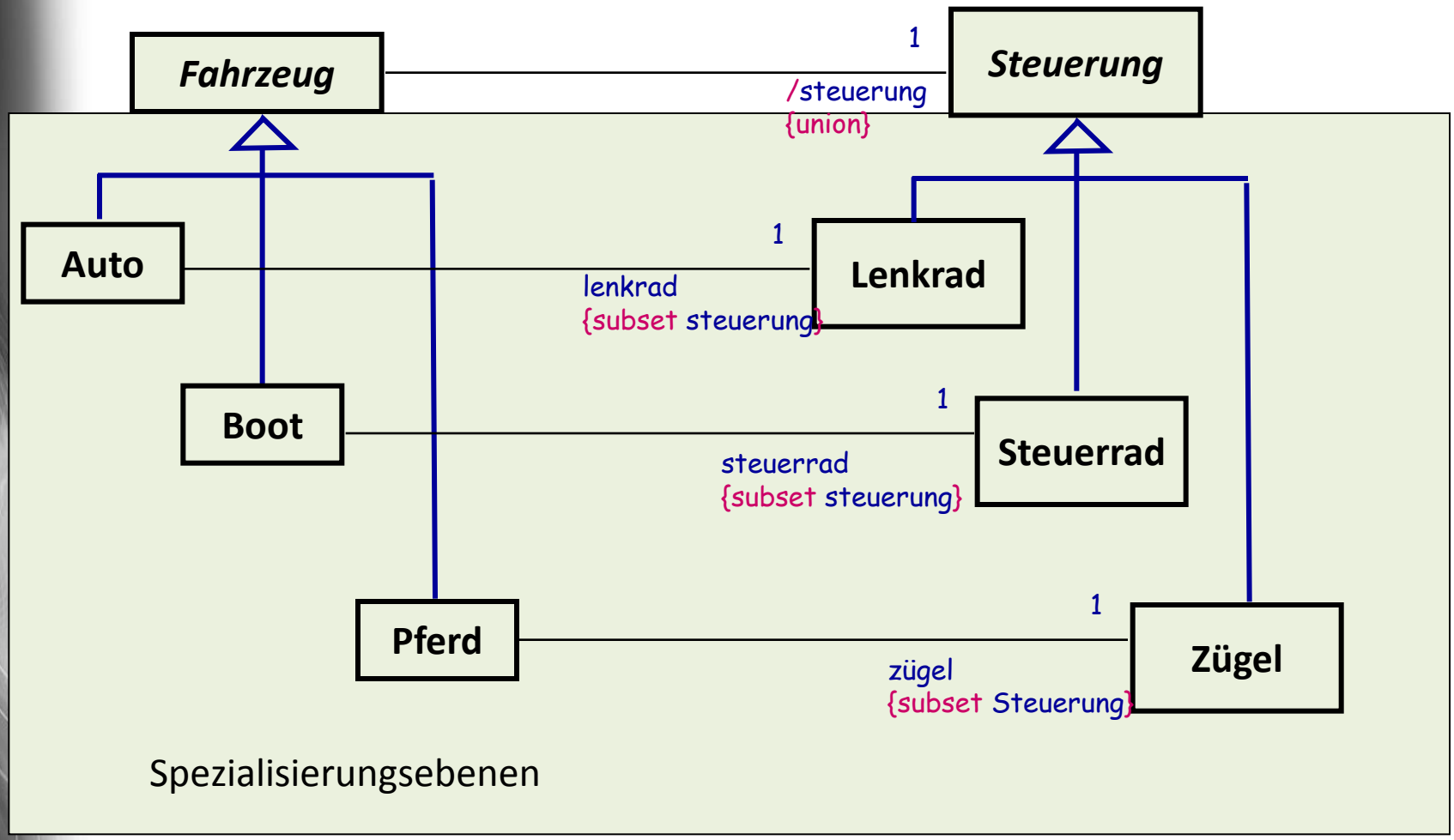
Vereinigung von Teilmengen von Objektbeziehungen



- teilnehmer ist als **berechenbare Vereinigung** aller Objektmengen bestimmt, die als **Teilmengen** von teilnehmer spezifiziert worden sind
- zu den Teilnehmern eines Termins zählt sowohl die Menge der einladenen (einladendeTln) als auch die der eingeladenen (geladeneTln) Teilnehmer
- **Vereinigungs-** und einzelne **Teilmenge**n-Beziehungen müssen nicht im gleichen Diagramm spezifiziert werden

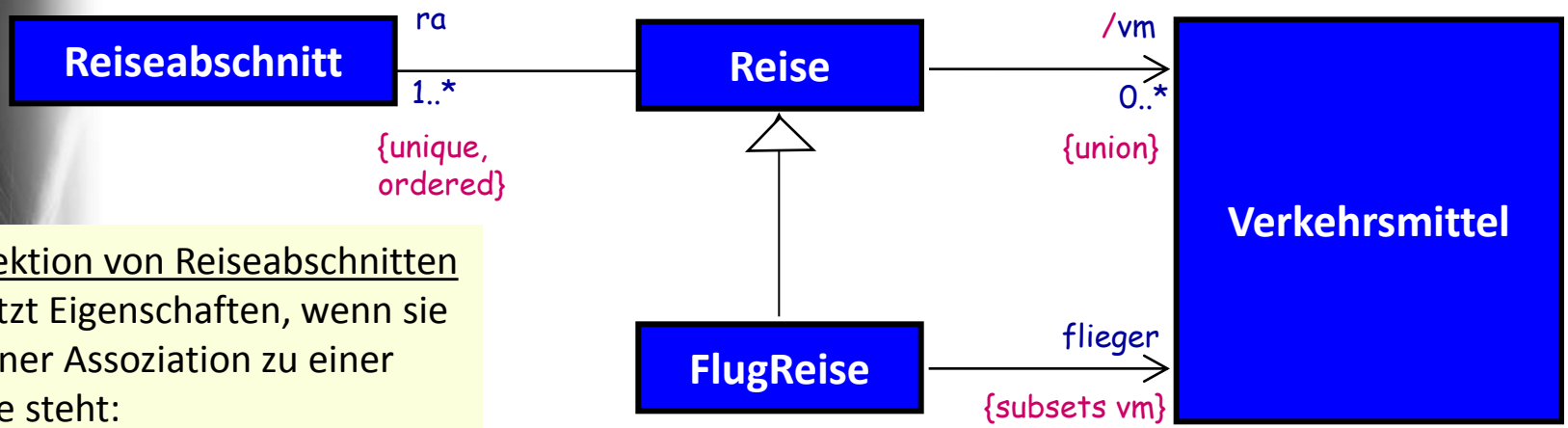
Das UML-Metamodell macht davon intensiv Gebrauch

Teil- und Vereinigungsmengen von Objektbeziehungen



Beispiel: mehrere Eigenschaften von Assoziationsenden

Man kann davon ausgehen, dass es noch weitere Reise-Spezialisierungen und vm-Teilmengen gibt



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

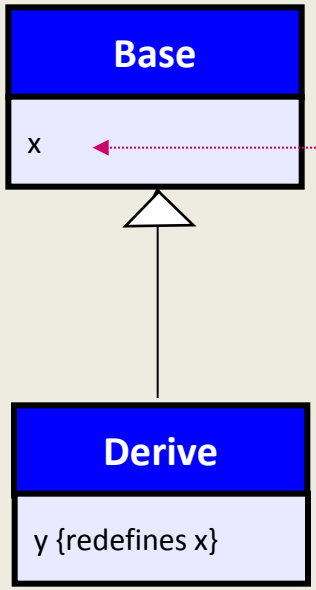
Kollektion von Reiseabschnitten besitzt Eigenschaften, wenn sie in einer Assoziation zu einer Reise steht:
- nicht leer
- echte Menge
- es gibt eine Ordnung

Es wird eine Teilmenge von Verkehrsmitteln ausgezeichnet
- kann leer sein
- unter *flieger* ist diese Menge einem Reise-Objekt bekannt,
  wenn es sich dabei um eine Flugreise-Objekt handelt
- ist eine Reise eine Flugreise,
  so sind alle Verkehrsmittel über *flieger* erreichbar
- *vm* ist eine abgeleitete Größe

Redefinition von Attributen

- **Merkmale** (Features) eines Classifiers (z.B. Klasse) sind:
  - Attribute, Operationen, navigierbare Assoziationsenden

- Im Kontext einer **Generalisierung/Spezialisierung** können alle geerbten Merkmale redefiniert werden



- Typ,
- Eindeutigkeit,
- Ordnung
- Multiplizität,
- weitere Eigenschaften

werden bei Vererbung
- **ohne** Redefinition von x übernommen
- **mit** Redefinition von x ersetzt

dennoch Konsistenzregeln:
(1) **Typ Y** von y muss konsistent zum **Typ X** von x sein: atleast X (gleich oder Ableitung von X)
(2) **Multiplizitätsintervall** von y muss in dem Intervall von x enthalten sein
(3) x **derived** ist, falls y redefines werden soll

- Name und Sichtbarkeit von y können von den Vereinbarungen von x beliebig abweichen (Name x könnte z.B. wieder neu verwendet werden)

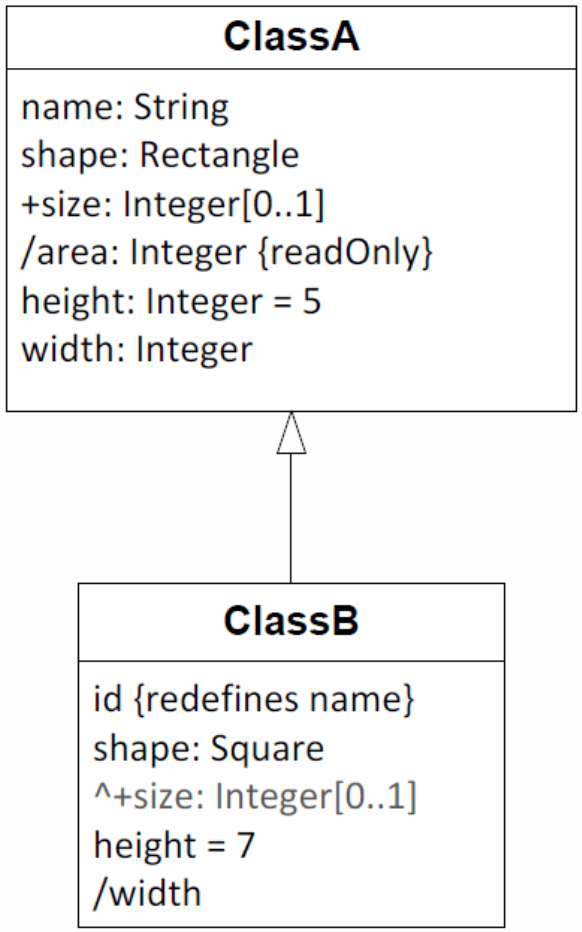
Redefinition (allgemein)

- Notation hängt von der jeweiligen Entity-Art ab
  - *Behaviour*
  - *Classifier*
  - *Operation*
  - *Property* *(wird jetzt behandelt)*
  - *State Machine*
  - *Template*

Konsistenzregeln für die anderen Entity-Arten unterliegen semantischen Variationspunkten, da sie von der jeweils gewählten Action-Semantic abhängen

Aussage früherer UML-Standards-
Muss nochmal geprüft werden

Beispiel aus dem UML-Standard

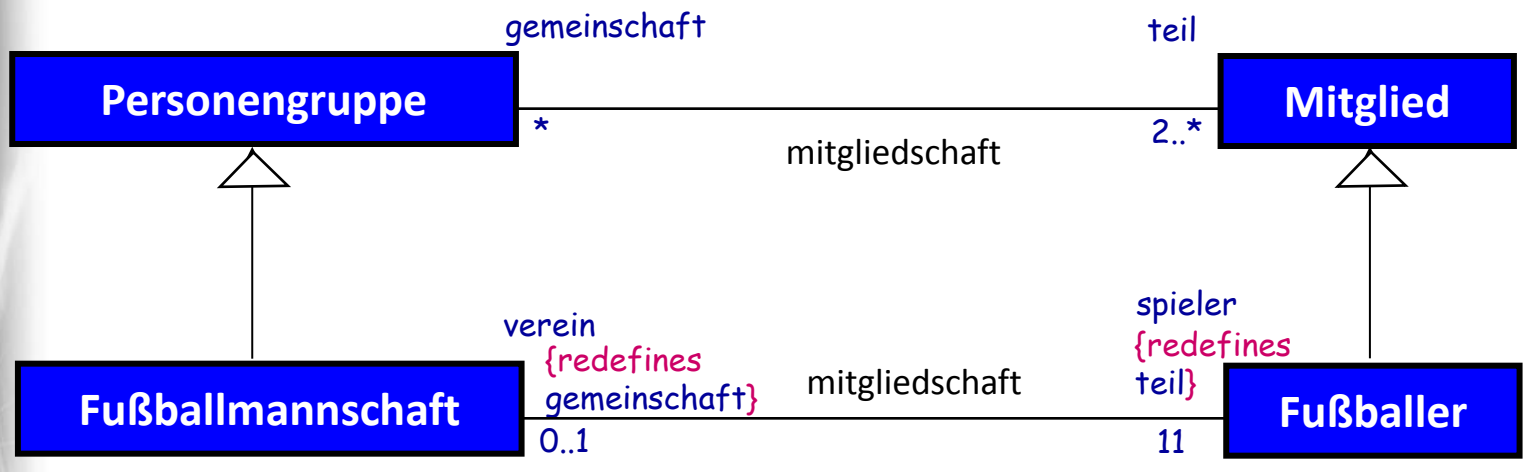


- ClassA::name is an attribute with type String.
- ClassA::shape is an attribute with type Rectangle.
- ClassA::size is a public attribute of type Integer with multiplicity 0..1.
- ClassA::area is a derived attribute with type Integer. It is marked as read-only.
- ClassA::height is an attribute of type Integer with a default initial value of 5.
- ClassA::width is an attribute of type Integer.

- ClassB::id is an attribute that redefines ClassA::name.
- ClassB::shape is an attribute that redefines ClassA::shape. It has type Square, a specialization of Rectangle.
- ClassB shows size as an attribute inherited from ClassA, as signified by the prepended caret symbol (see 9.2.4).
- ClassB::height is an attribute that redefines ClassA::height. It has a default of 7 for ClassB instances that overrides the ClassA default of 5.
- ClassB::width is a derived attribute that redefines ClassA::width, which is not derived.

Offene Frage: kann man Attribute gleichen Namens in Basis und Ableitungsklasse angeben, ohne dass sie implizit redefiniert werden?

Redefinition navigierbarer Assoziationsenden



Klassendiagramme - eine vertiefende Betrachtung

1. Attribute, Assoziationen und Links
2. Operationen und Methoden

Operationscharakteristika

Allgemeine Charakteristika

- Signatur ✓
- Ausnahme (*exceptions*) ✓
- Sichtbarkeit ✓
- Methode (*method*) ✓
- Signal (*signal*)

Operationseigenschaften

- Abstraktheit (*abstract*) ✓
- Static (*static*) ✓
- Ausführung (*concurrency*)
- Laufzeit-Bedingungen (*constraints*)
- Redefinition (*redefined*)
- Finalisierung (*leaf*) ✓
- Seiteneffekt (*query*) ✓

UML-Bekenntnis zu Programmausnahmen nicht präzise

1. Festlegung der Action Semantic ist ein Variationspunkt

2. OO-Sprachen haben z.T. unterschiedliche Positionen

   C++, Java, Python: Signatur ohne Exception
   SDL: Signatur mit Exception

3. Ausnahmen in UML bleiben bei Redefinition und Überladungsregeln unberücksichtigt

   Frage: Wodurch unterscheiden sich zwei Operationssignaturen in UML?

4. Ausnahmen als optionaler Bestandteil einer Operationssignatur kann nur vermutet werden

[**exception** nameList]

?

name ([paramList]) [: returnType]

Operationssignatur

Operationscharakteristika

Allgemeine Charakteristika

- Signatur ✓
- Ausnahme (*exceptions*) ✓
- Sichtbarkeit ✓
- Methode (*method*) ✓
- Signal (*signal*)

Operationseigenschaften

- Abstraktheit (*abstract*) ✓
- Static (*static*) ✓
- Ausführung (*concurrency*)
- Laufzeit-Bedingungen (*constraints*)
- Redefinition (*redefined*)
- Finalisierung (*leaf*) ✓
- Seiteneffekt (*query*) ✓

UML-Bekenntnis zu Programmausnahmen nicht präzise

1. Festlegung der Action Semantic ist ein Variationspunkt

2. OO-Sprachen haben z.T. unterschiedliche Positionen

   C++, Java, Python: Signatur ohne Exception
   SDL: Signatur mit Exception

3. Ausnahmen in UML bleiben bei Redefinition und Überladungsregeln unberücksichtigt

   Frage: Wodurch unterscheiden sich zwei Operationssignaturen in UML?

4. Ausnahmen als optionaler Bestandteil einer Operationssignatur kann nur vermutet werden

[«stereotype»]
[visibility]
name ([paramList]) [: returnType]
[{ propertyStringList }]

[**exception** nameList]

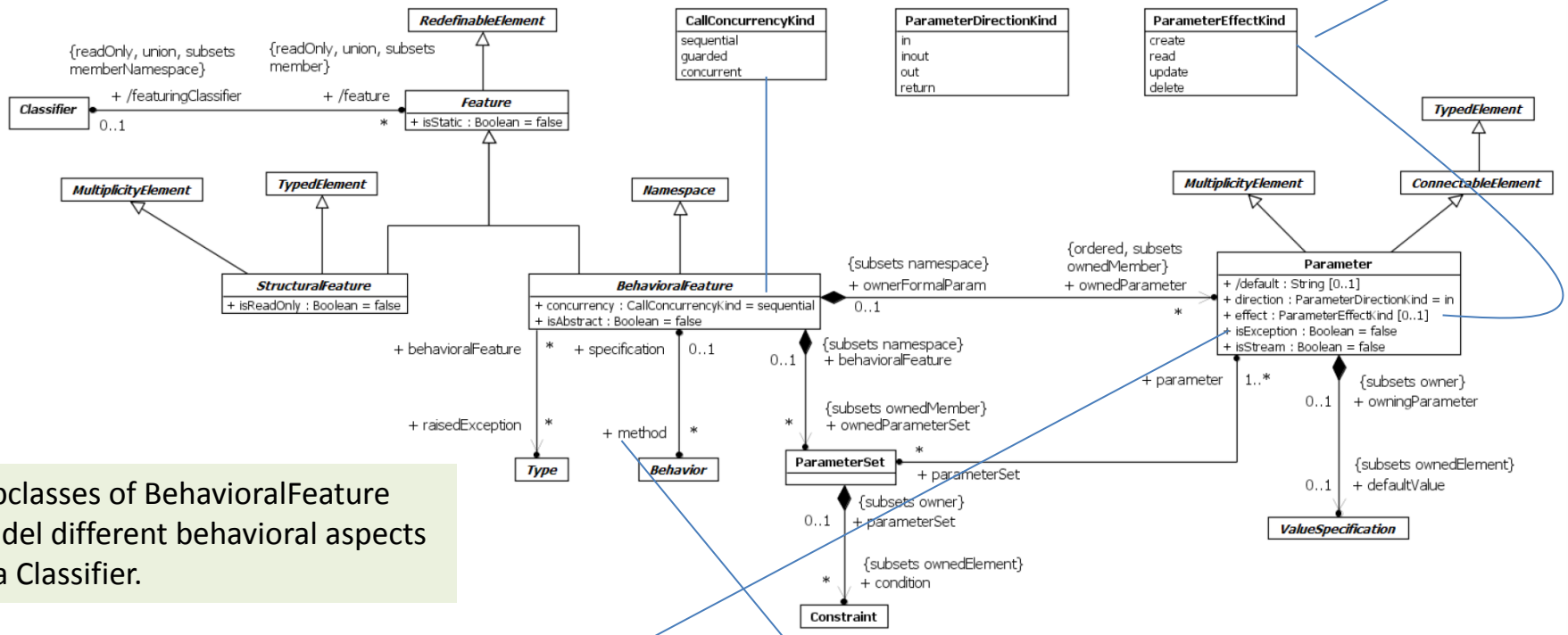
?

Operationssignatur

The **effect property** may be used to specify what happens to **objects** passed in or out of a Parameter. It

Overloading ???:

A **ParameterSet** owned by a BehavioralFeature is an element that provides **alternative sets of inputs or outputs that** the Behaviors that implements that BehavioralFeature may use.



Subclasses of BehavioralFeature model different behavioral aspects of a Classifier.

Figure 9.9 Fea

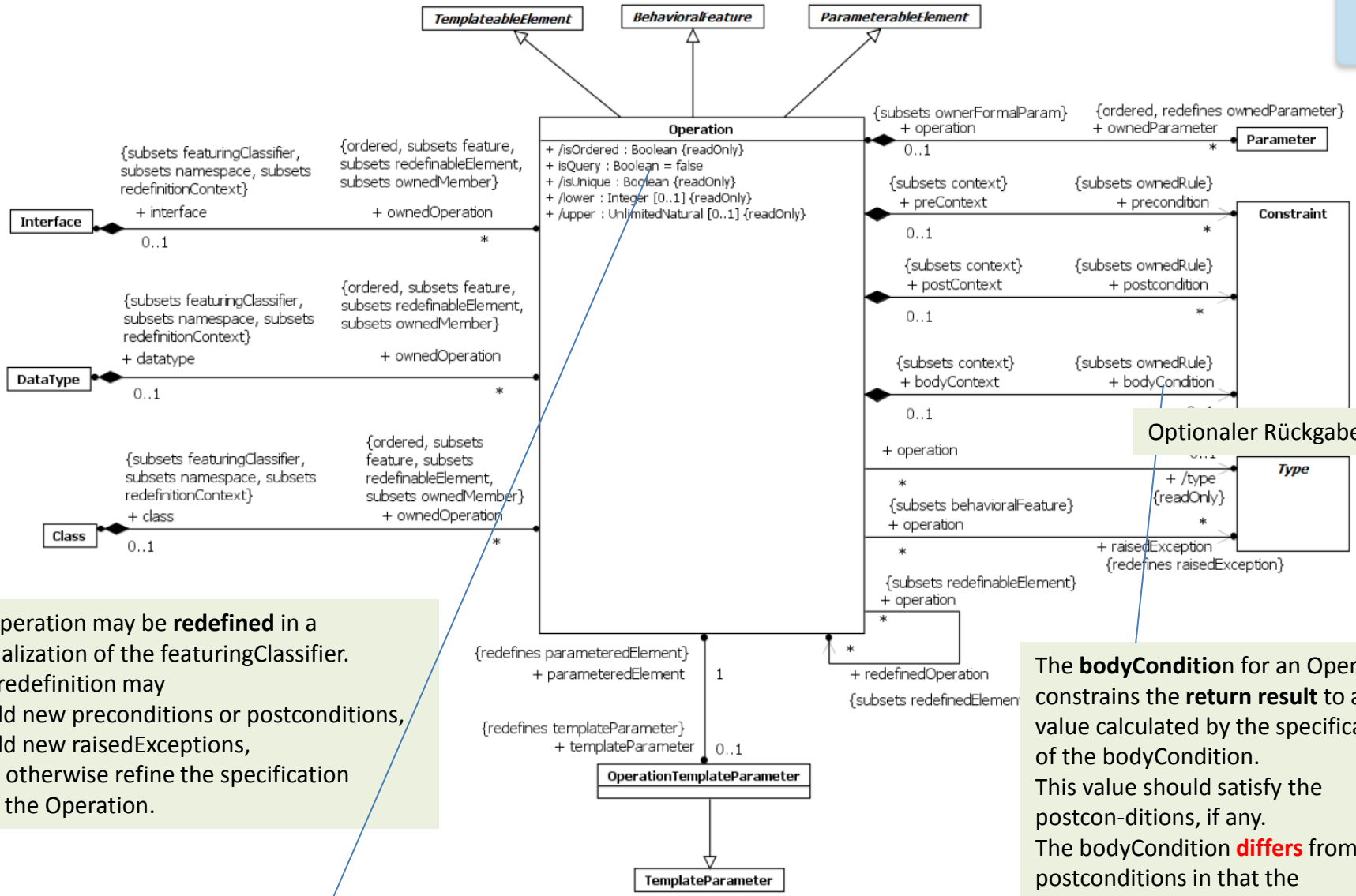
A BehavioralFeature may raise an **exception** during its invocation. Possible exception types may be specified by attaching them to the BehavioralFeature using the **raisedException** association.

One way to define the behavioral response of a BehavioralFeature is to specify one or more Behaviors as **methods** that implement the BehavioralFeature.
An invocation of the BehavioralFeature then results in the execution of one of the associated methods

Effect-Property

create	Objects passed out of executions of the behavior as values of the parameter do not exist before those executions start.
read	Objects that are values of the parameter have values of their properties, or links in which they participate, or their classifiers retrieved during executions of the behavior.
update	Objects that are values of the parameter have values of their properties, or links in which they participate, or their classification changed during executions of the behavior.
delete	Objects that are values of the parameter do not exist after executions of the behavior are finished.

The **effect property** may be used to specify what happens to **objects** passed in or out of a Parameter. It



An Operation may be **redefined** in a specialization of the featuringClassifier. This redefinition may

- add new preconditions or postconditions,
- add new raisedExceptions,
- or otherwise refine the specification of the Operation.

If the **isQuery** property is true, an invocation of the Operation shall not modify the state of the instance or any other element in the model.

Optionaler Rückgabetyp

The **bodyCondition** for an Operation constrains the **return result** to a value calculated by the specification of the bodyCondition.
This value should satisfy the postcon-ditions, if any.
The bodyCondition **differs** from postconditions in that the bodyCondition may be overridden when an Operation is redefined, whereas postconditions may only be added during redefinition.