ModSoft

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

Teil II: Strukturmodellierung

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

fischer@informatik.hu-berlin.de

bisher

1. Klassen und Assoziationen (zweiter Eindruck)
2. Interface
3. Datentyp, Signal, Signalempfangsspezifikation
4. Klassen und Verhaltensbeschreibungen (erster Eindruck)

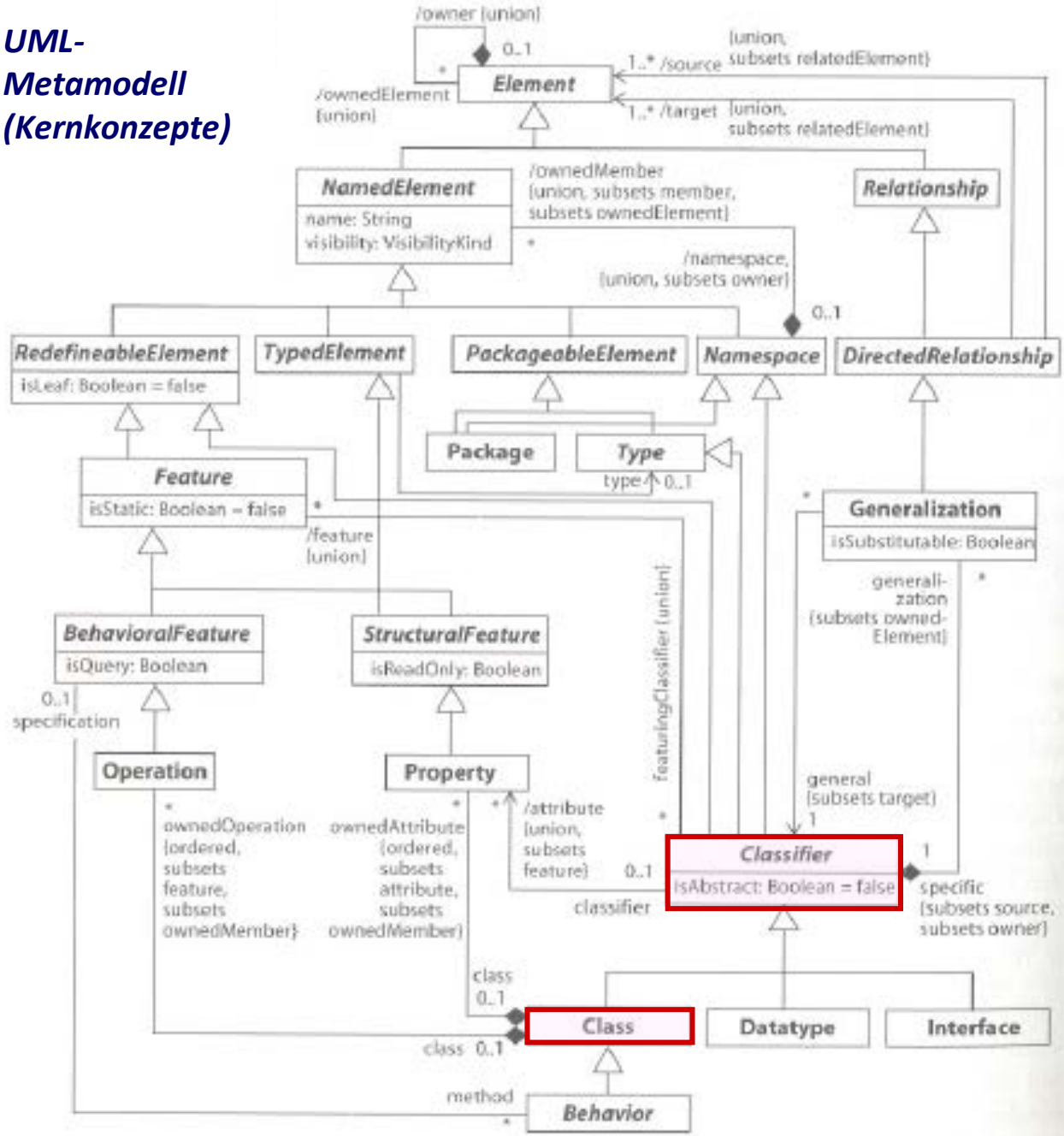
M3
MOF 2.0

M2
UML 2.0

M1
Benutzer-modell

M0
Objekte der Realität

UML-Metamodell (Kernkonzepte)



Klasse (Zusammenfassung)

- … ist eine Instanz der Metaklasse class
  als konkrete Spezialisierung der Metaklasse Classifier
- … wird als Rechteck dargestellt, Name als Substantiv
- … verkörpert einen Typ
  (andere Beispiele Datentyp, Signal, Interface, Behaviour(Verhalten), …)
- … enthält die Beschreibung von Struktur u. Verhalten von Objekten,
  die sie erzeugt oder
  die mit ihr erzeugt werden können
- … setzt sich aus
  - Attributen, Operationen, Signalempfangsspezifikationen, lokale Elemente und Invarianten zusammen
- Verhalten der Objekte wird durch Signale beeinflusst,
  die den Objekten bekannt sind (verstanden werden)
  - möglich sind: Signalempfänge (Nachbildung von Operationsrufen),
  - zu jeder “verständlichen” Nachricht gehört eine passende Operation
- Nicht-verstandene Nachrichten
  werden ignoriert bzw. lösen Ausnahmen aus
- Zusicherungen (als Invarianten formuliert)
  bestimmen einzuhaltende Regeln bei der Ausführung von Operationen oder allgemein für die Einhaltung von Objekteigenschaften

Inhalt

1. Klassen und Assoziationen (zweiter Eindruck)
2. Interface
    a. Konzept und Anwendung
    b. Vertiefung von Abhängigkeitsbeziehung
    c. Vertiefung von Erweiterungsbeziehungen
       Spezialisierung - Stereotypisierung
3. Datentyp, Signal, Signalempfangsspezifikation
4. Klassen und Verhaltensbeschreibungen (erster Eindruck)

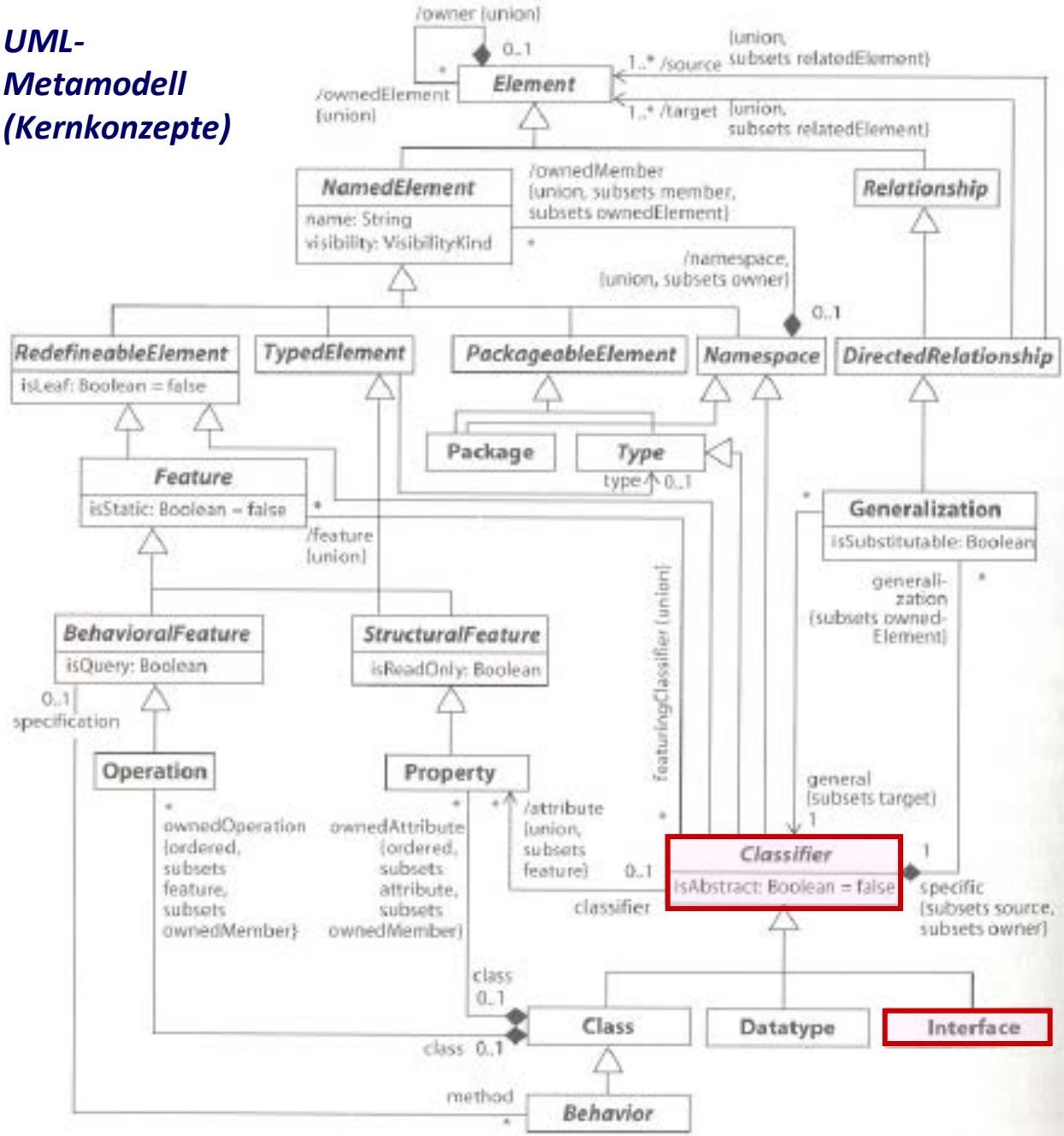
M3
MOF 2.0

M2
UML 2.0

M1
Benutzer-modell

M0
Objekte der Realität

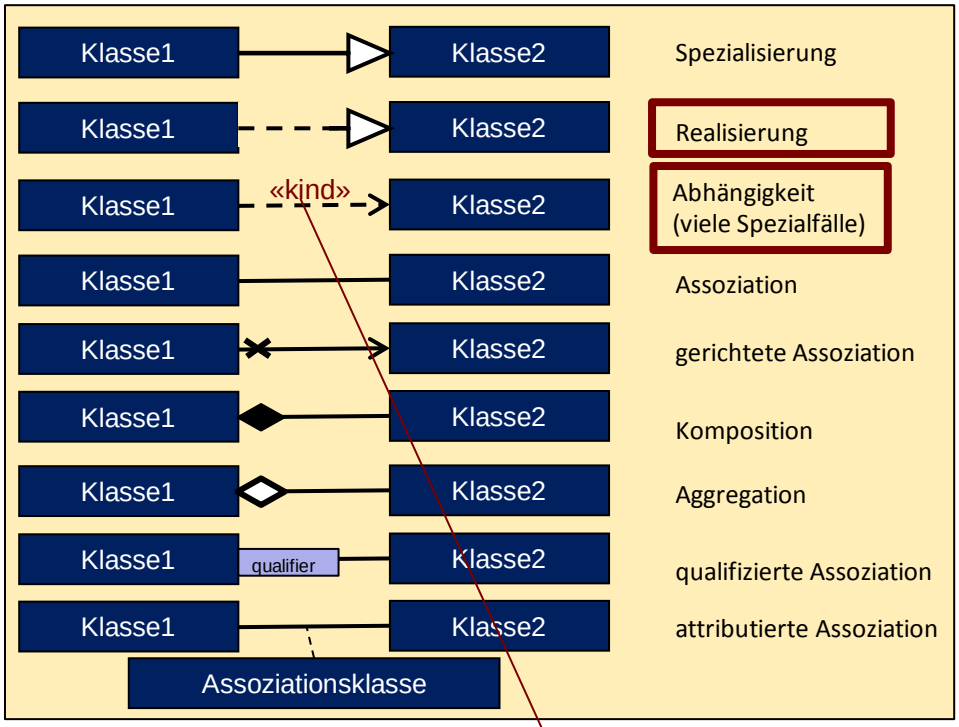
UML-Metamodell (Kernkonzepte)

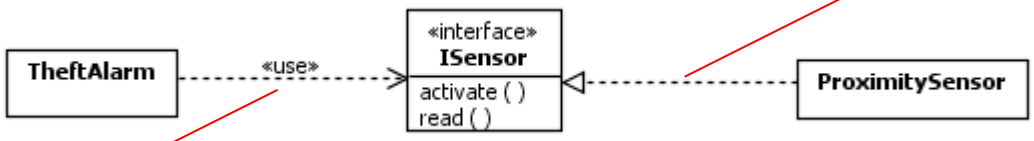


Interface

- **Schnittstellen** beschreiben
  mit einer Menge von Operationen
  und Attributen
  einen **ausgewählten Teil**
  der **extern sichtbaren** Merkmale
  von zugeordneten Modellelementen
  (wie Klassen, Komponenten und Ports)

- **Modellelemente** können Schnittstellen
  - realisieren (provided)
    oder
  - benutzen (required)





Realisierung

Abhängigkeitsbeziehung

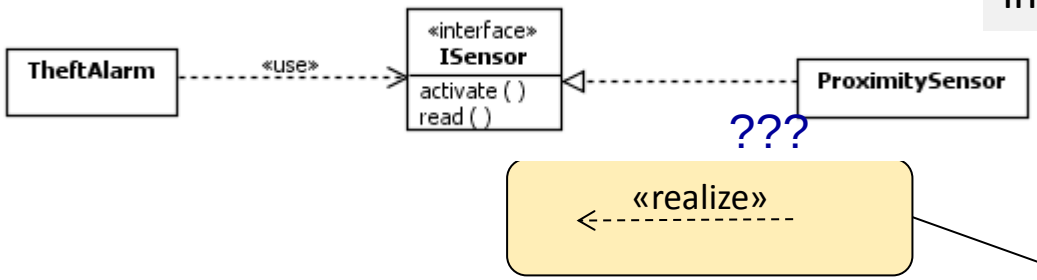
«abstraction»
«trace»
«refine»
«derive»
«use»
«create»
«instantiate»
«call»

~~«realize»~~

~~«permit»~~

Warum so unterschiedlich?

Suche nach einer Antwort im UML-Metamodell



«realize»

1. Frage

Schlüsselworte

«abstraction»

«trace»

«refine»

«derive»

«use»

«create»

«instantiate»

«call»

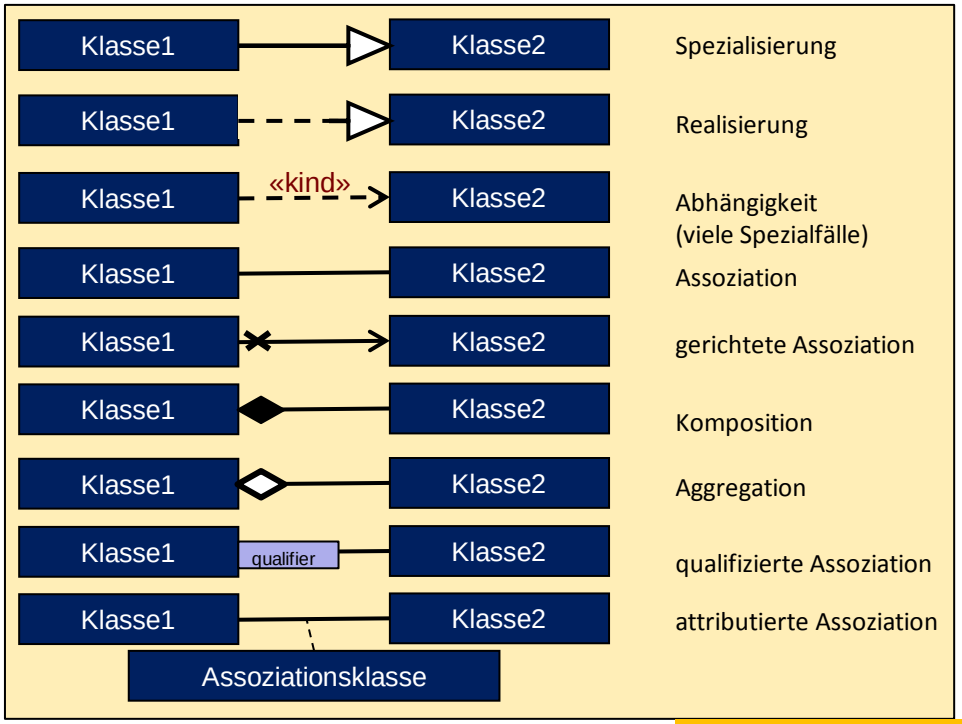
Stereotypen

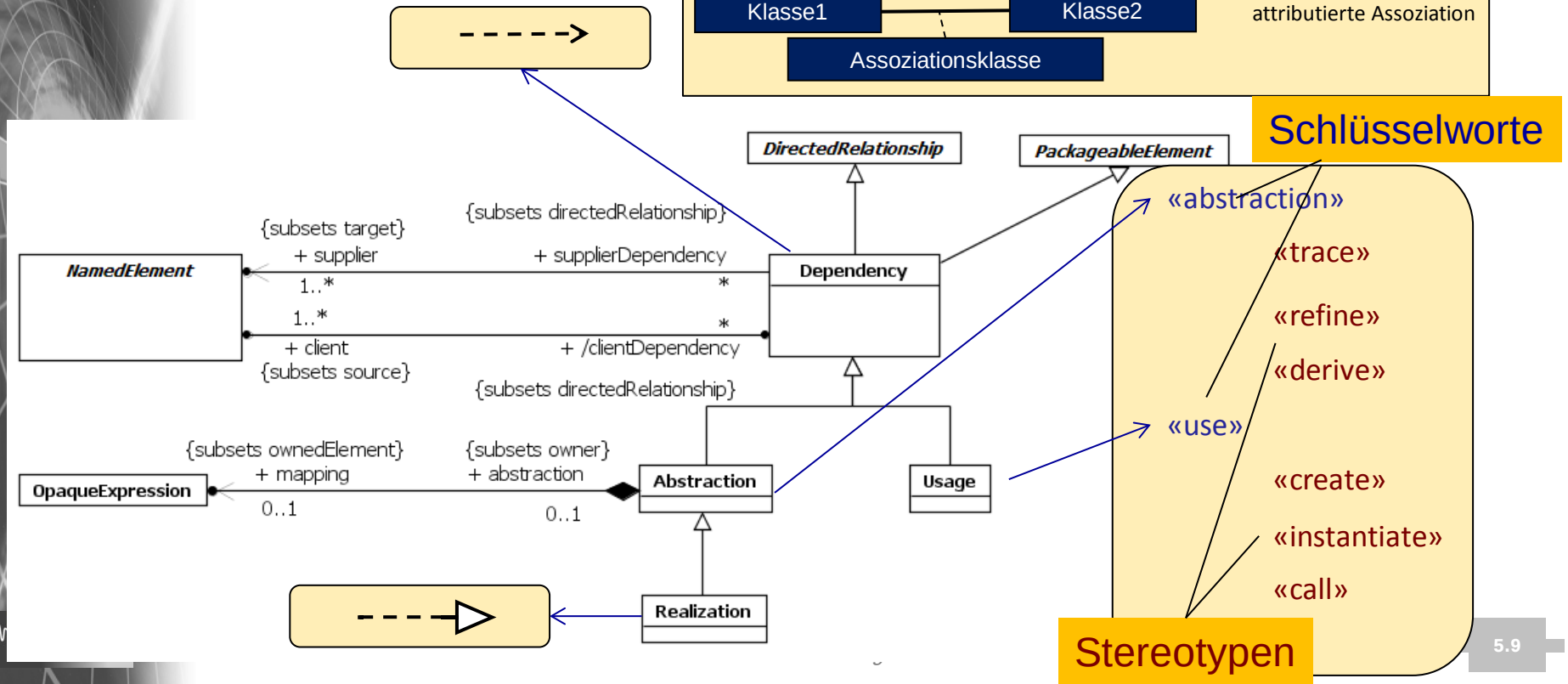
2. Frage

Beziehungen

Dependency ist eine konkrete Klasse
→
Pfeil ohne Zusatzangabe ist erlaubt

(beschreibt allg. eine Abhängigkeit)





Verwendung von Schlüsselworten in der UML-Sprachdefinition

- … erlaubt in der konkreten Syntax eine **semantische Variation** gleicher Grundsymbole

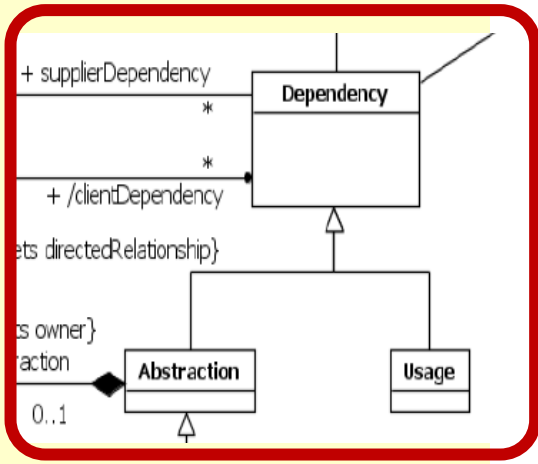
Beispiel-1

Classifier und **Class**

→ Rechteck

andere Metaklassen als Spezialisierungen von **Classifier**

→ Rechteck mit speziellem Schlüsselwort



Beispiel-2

Dependency (Abhängigkeit)

→ gestrichelter offener Pfeil

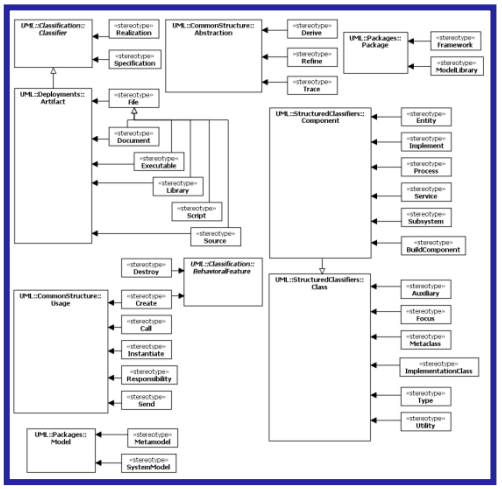
zwei Metaklassen als direkte Spezialisierungen von **Dependency**

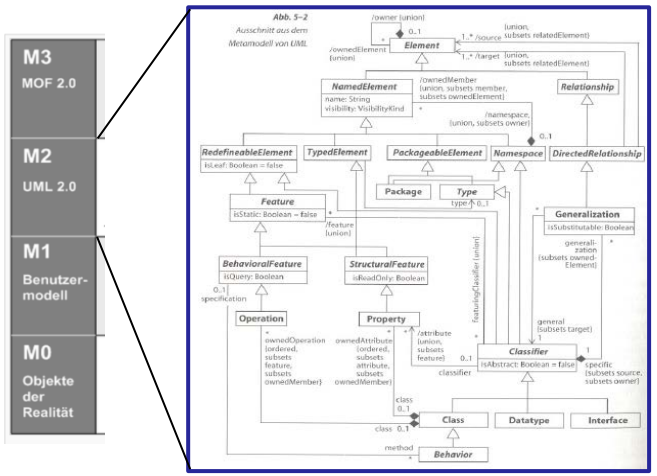
→ gestrichelter offener Pfeil mit speziellem Schlüsselwort:
abstraction, use

Stereotype und Profile

~ inherenter Erweiterungsmechnismus im UML-Metamodell

- mit **Stereotypen** ist es möglich,
  sämtliche **Metaklassen** in einer restriktiven Art und Weise hinsichtlich ihrer Funktionalität und Notation (konkrete Syntax) auf der Sprachdefinitionsebene (M2) **einschränkend/erweiternd** festzulegen

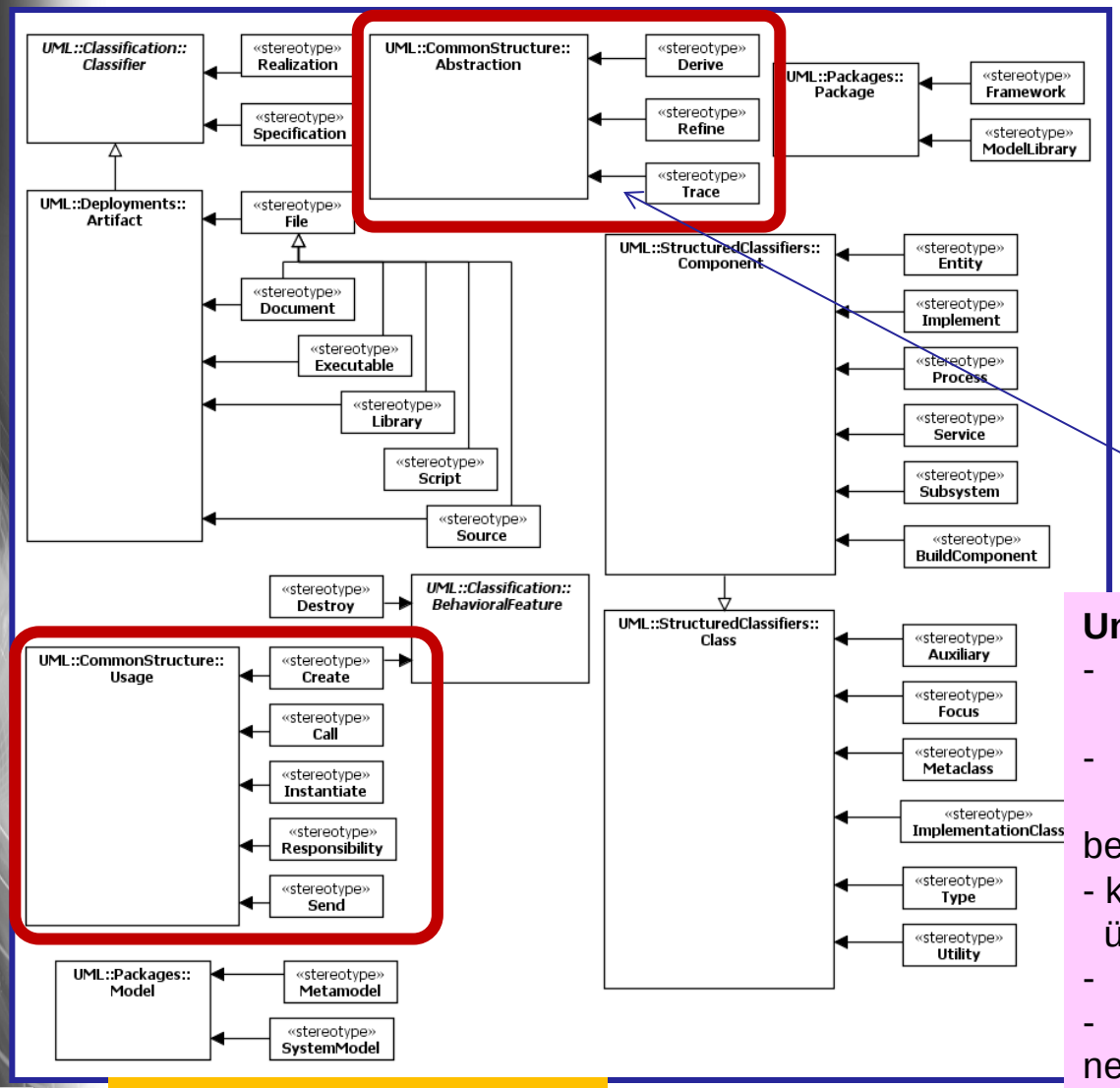


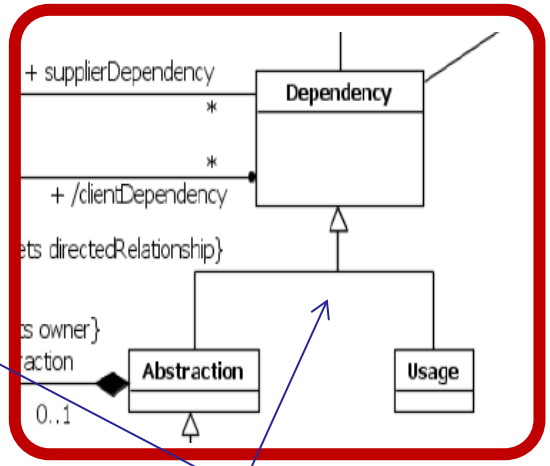


- Sämtliche dieser Art von Erweiterungen können für bestimmte Ziele /Anwendungsdomäne gruppiert werden: → Profile

- Unter Anwendung dieser Profile können die Stereotypen (wie andere Metaklassen) als Konzepte zur domänenspezifischen Modellierung auf der M1-Ebene benutzt werden

Verwendung von Standardstereotypen in der UML-Sprachdefinition



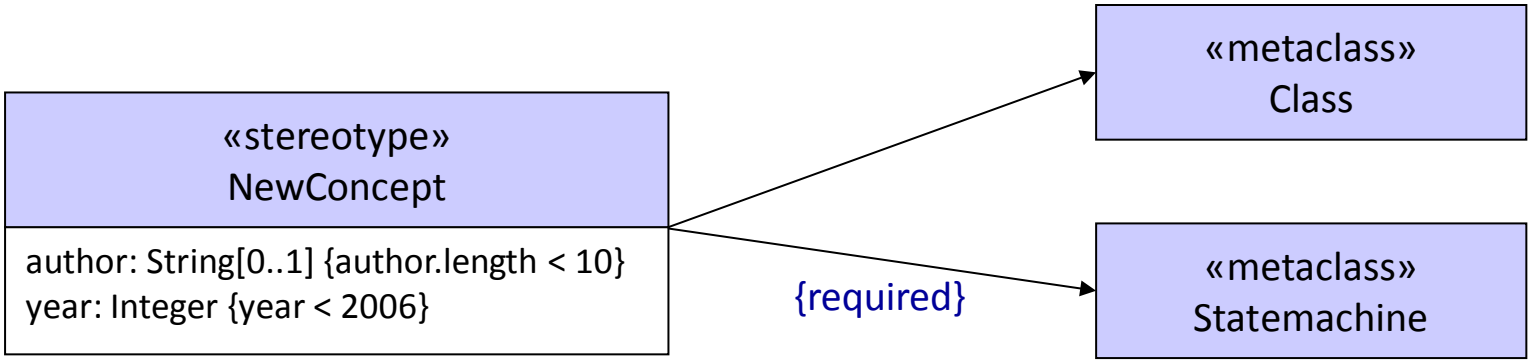


Unterschied von
- Spezialisierung einer Metaklasse und
- Stereotypisierung

bei einer Stereotypisierung dürfen
- keine existierende Eigenschaften überschrieben werden und
- keine neuen verallgemeinernden Metaklassen und Beziehungen neu hinzukommen

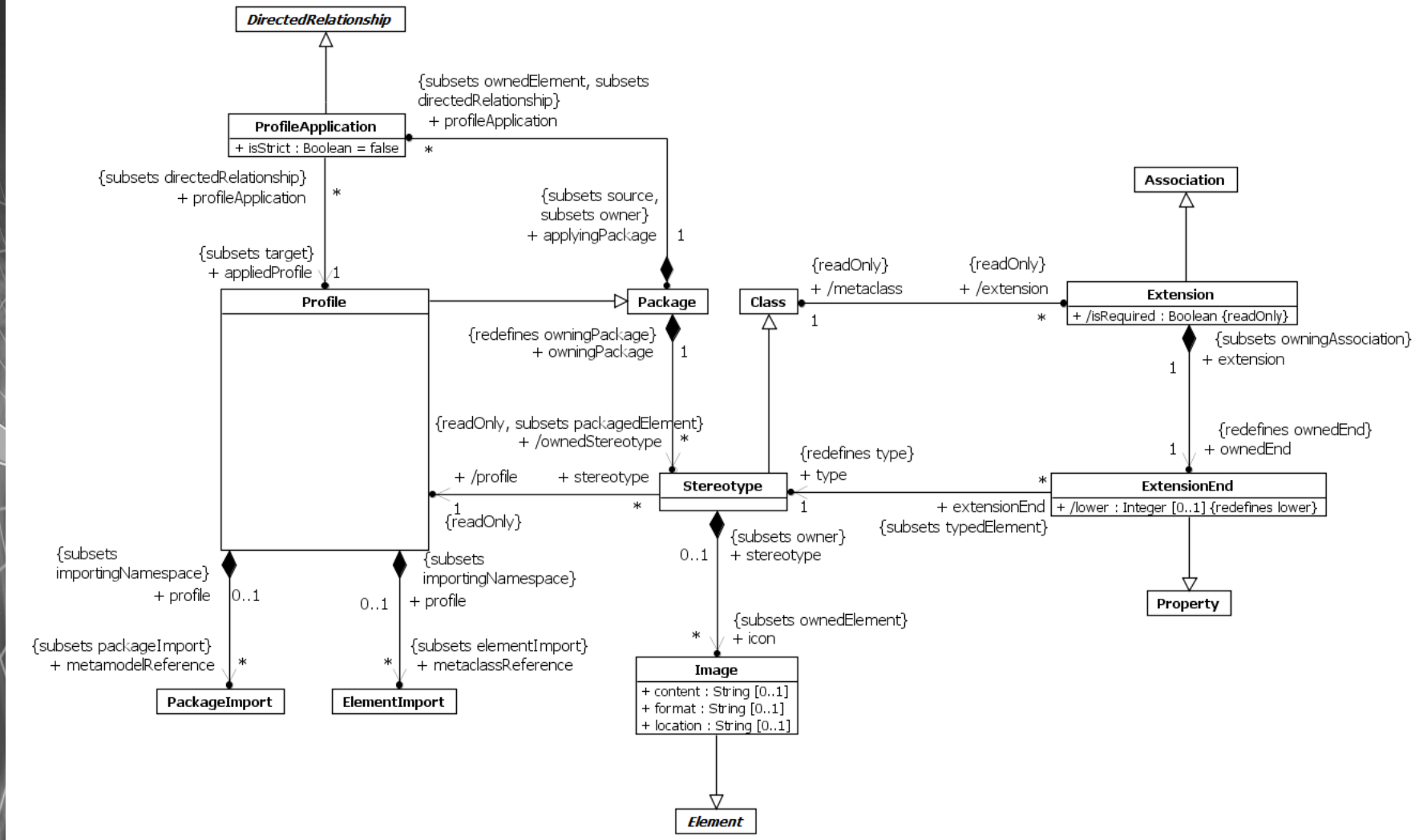
UML-Standard-Profile

Beispiel: Stereotype



- NewConcept erweitert die Metaklassen Class und Statemachine um zwei neue Attribute und zwei Einschränkungen
- für Klassen ist diese Erweiterung optional, d.h. es stehen auf der M1-Ebene Class und NewConcept zur Verfügung
- für Zustandsmaschinen steht nur das Konzept NewConcept definiert werden

Profile: Definition im Metamodell



Fazit

- Die Notwendigkeit einige Konzepte des UML-Metamodells
  - per Metaklassenspezialisierung zu definieren
    und andere
  - durch Stereotypes

  kann noch nicht endgültig erklärt werden

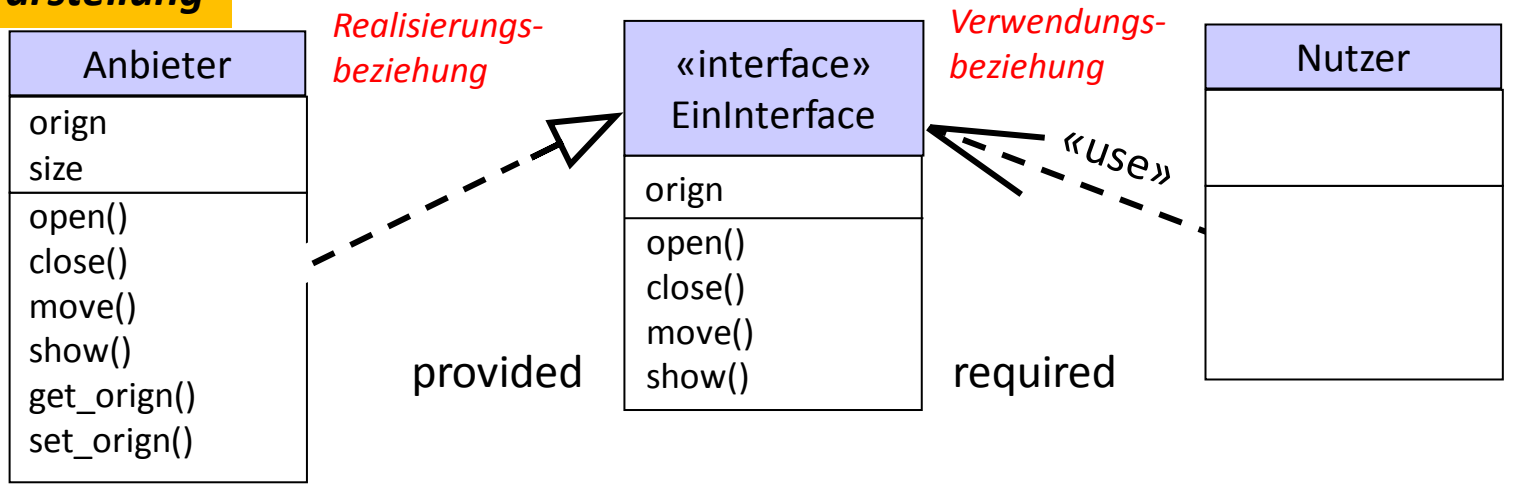
- (Schwache) Vermutung 1:

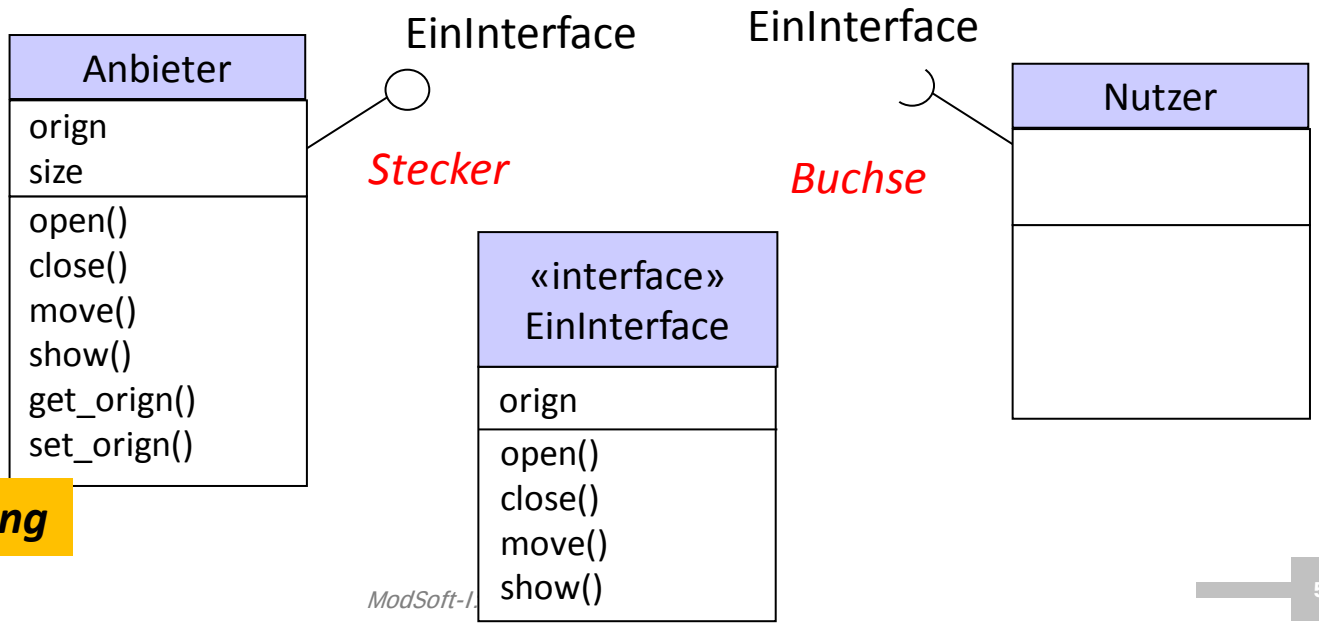
  Man kann durch Anwendung der Metaklassenspezialisierung nicht die gewünschte Semantik der betreffenden Konzepte

- (Starke) Vermutung 2:
  Die Anwendung des UML-Standardprofils ist nur eine Empfehlung, die nicht zwingend ist.
  D.h., auf alle Konzepte des UML-Standard-Profiles kann man verzichten

Alternative Darstellung: Interface-Nutzung

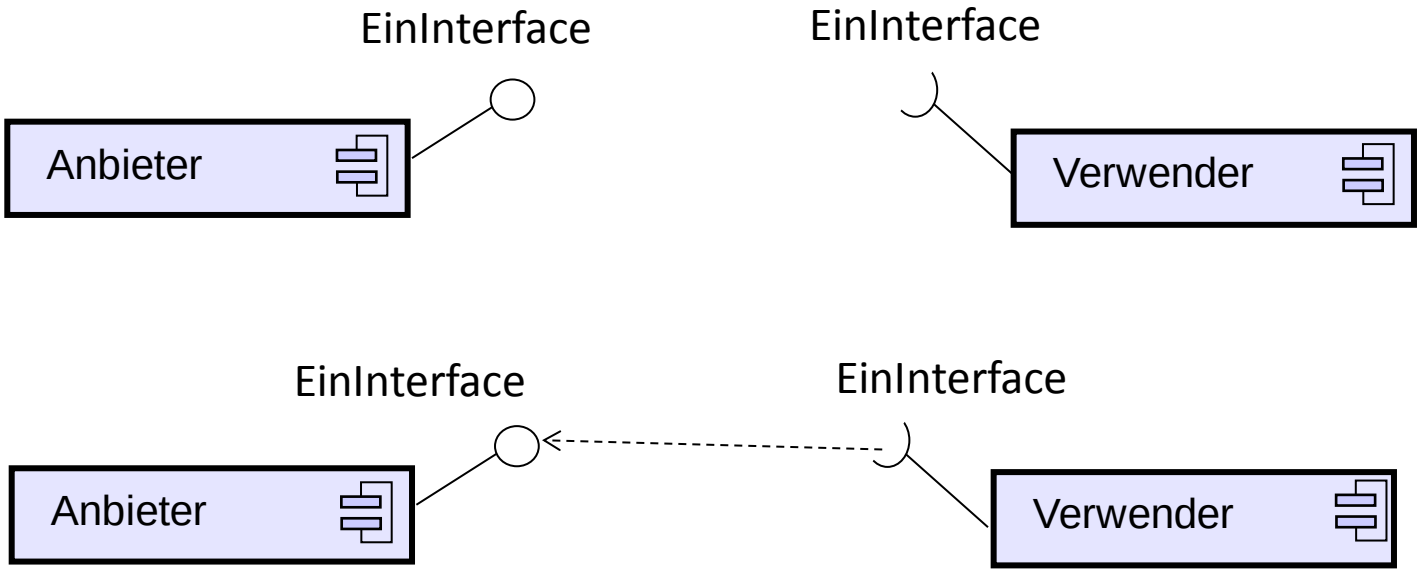
gekoppelte Darstellung





entkoppelte Darstellung

Interface für Komponenten



übliche Darstellung für Komponenteninstallation

Machen Sie sich folgendes klar

- Anbieter mit vielen Verwendern,
  der selbst andere Anbieter verwendet

- die Passfähigkeit von Komponenten auf der Basis identischer Schnittstellen hilft in der Praxis nur bedingt
  Vertrag (Contract) per Schnitsstellen-Name, Signatur von Operationen reicht nicht aus

- Was man benötigt, ist die Zusicherung einer Konformität des Verhaltens der angebotenen Schnittstellen-Operationen

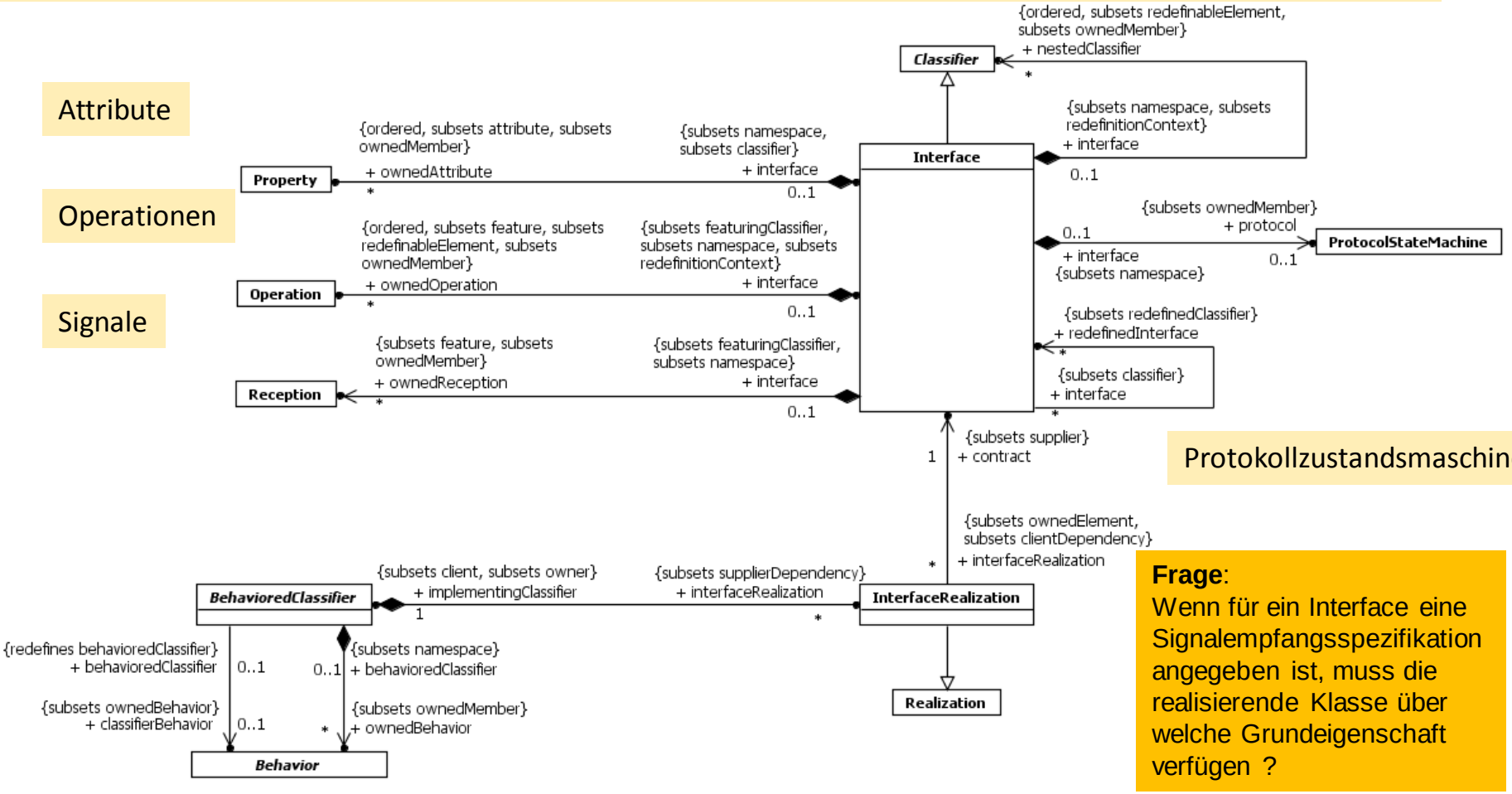
Interface, im UML Metamodell

Interfaces may own a **ProtocolStateMachine** that specifies event sequences and pre/post conditions for the Operations and Receptions described by the Interface.
A **BehavioredClassifier** realizing an Interface shall comply with the **ProtocolStateMachine** owned by the Interface.

Attribute

Operationen

Signale



Protokollzustandsmaschine

Frage:
Wenn für ein Interface eine Signalempfangsspezifikation angegeben ist, muss die realisierende Klasse über welche Grundeigenschaft verfügen ?

An **InterfaceRealization** is a specialized realization relationship between a **BehavioredClassifier** and an **Interface**. This relationship signifies that the realizing BehavioredClassifier conforms to the **contract** specified by the Interface.

Protocol State Machine

- **NOTE.** Because ProtocolStateMachines provide a “black box” view of the behavior of a Classifier, their States may not necessarily correspond to the States of internal behavioral StateMachines.

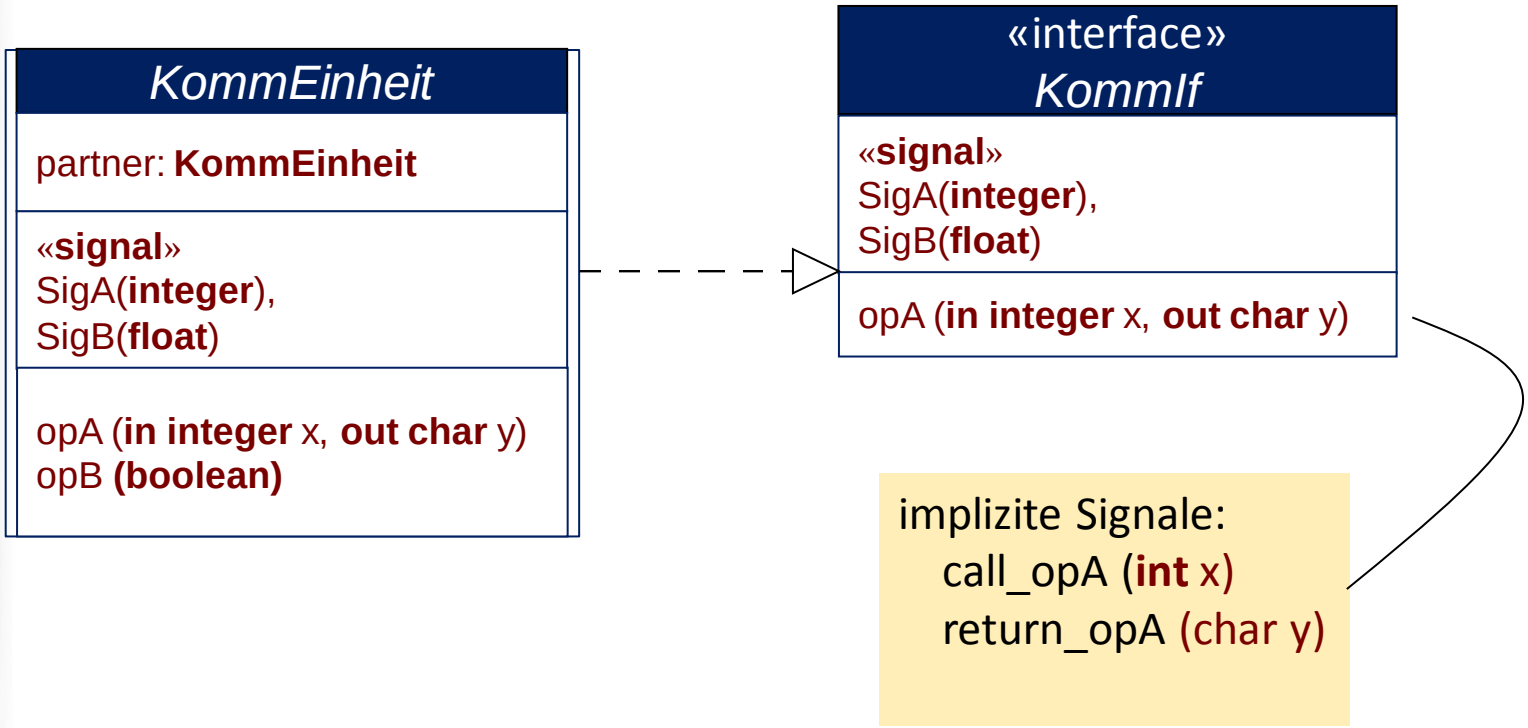
- ProtocolStateMachine interpretation can vary from:

  1. *Declarative* ProtocolStateMachines, which specify the legal Transitions for BehavioralFeature invocations. The effects of a BehavioralFeature invocation is not specified. This type of specification only provides a contract for the user of the context Classifier.

  2. *Executable* (run time) ProtocolStateMachines, which specify all Event occurrences that an object may receive and handle, together with the Transitions that these trigger. In this case, the legal Transitions for BehavioralFeature invocations must match exactly the triggered Transitions or a run-time exception occurs. The invocation results in the execution of the method associated with the invoked BehavioralFeatures.

- The specifications for both interpretations is the same, the only difference being the direct dynamic implication that the latter interpretation provides.
- The more sophisticated forms of modeling encountered in behavioral StateMachines such as compound Transitions, submachine StateMachines, composite States, and concurrent orthogonal Regions, can also be used for ProtocolStateMachines. For example, concurrent Regions make it possible to express protocols where an instance can have several active States simultaneously. Submachine StateMachines and compound transitions can be used for factorizing complex ProtocolStateMachines.
- A Classifier may have several ProtocolStateMachines. This can be used, for example, when a Classifier has multiple parents, each having its own ProtocolStateMachine, and the protocols are orthogonal. …

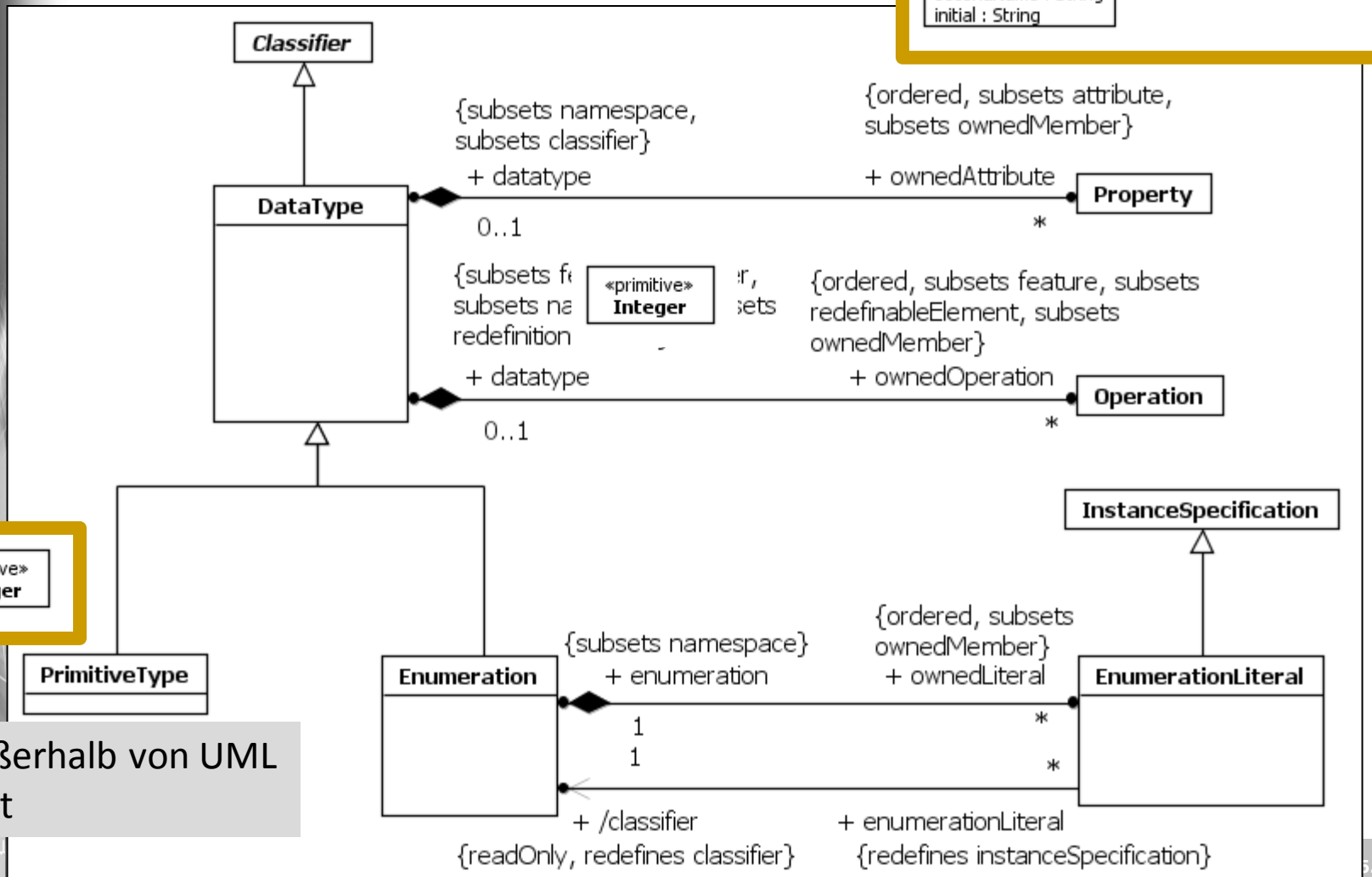
Weitere Frage: Operationsrufe



Inhalt

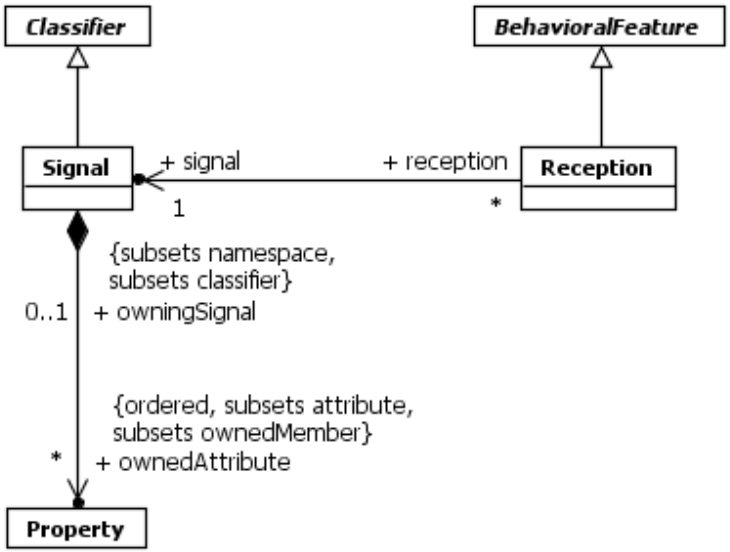
1. Klassen und Assoziationen (zweiter Eindruck)
2. Interface
    a. Konzept und Anwendung
    b. Vertiefung von Abhängigkeitsbeziehung
    c. Vertiefung von Erweiterungsbeziehungen
       Spezialisierung - Stereotypisierung
3. Datentyp, Signal, Signalempfangsspezifikation
4. Klassen und Verhaltensbeschreibungen (erster Eindruck)

Datatype im UML-Metamodell



sind außerhalb von UML definiert

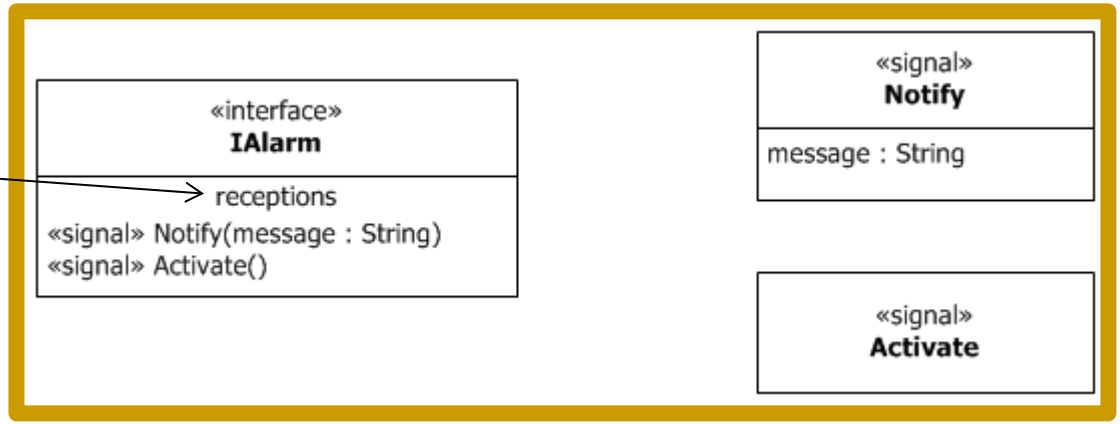
Signal im UML-Metamodell



Operationen und
Assoziationen für
Signale sind unzulässig

Bestätigung im UML-Metamodell
noch offen

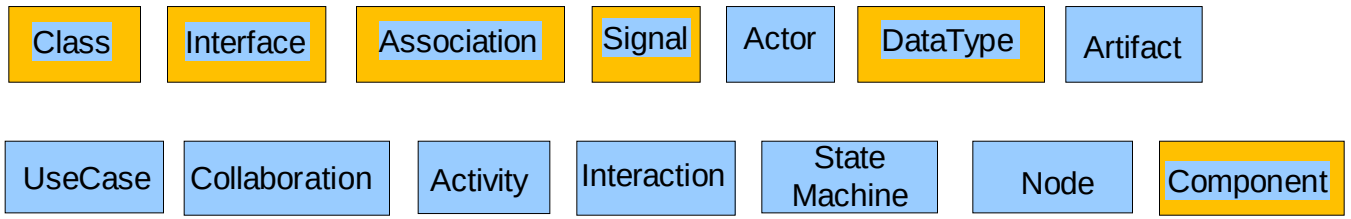
optionale Angabe



Strukturelle Modellelemente: Überblick

- bilden die Struktur eines Modells
  (meist in statischer aber auch dynamischer Art und Weise)

- generalisierende Zusammenfassung als *Classifier*



z.T. mit weiteren Spezialisierungen

Einige wurden bereits besprochen

Inhalt

1. Klassen und Assoziationen (zweiter Eindruck)
2. Interface
3. Datentyp, Signal, Signalempfangsspezifikation
4. Klassen und Verhaltensbeschreibungen (erster Eindruck)

Erweiterter Endlicher Zustandsautomat

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

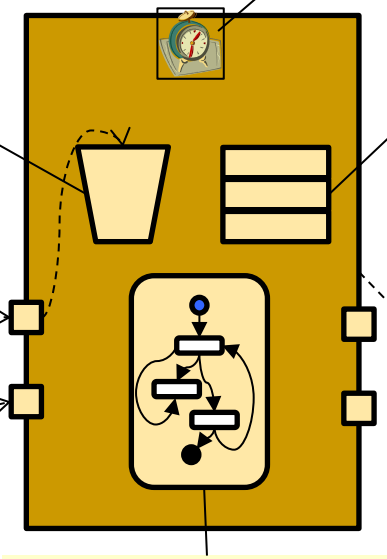
A) Erweiterung eines Endlichen Zustandsautomaten
- Variablen
- Timer
- Pool von Nachrichten (asynchrone Komm.)
- Trigger (Zustandsübergänge)
- Nachrichten/Remote-Op-Rufe
- Guards (als Bedingungen über Zustandsgrößen)

B) Zustandserweiterung
- Eintritts-Aktivität
- Do-Aktivität
- Austritts-Aktivität
- Defer-Aktivität

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

Im Kontext einer aktiven Klasse

Automaten zur Laufzeit



lokaler Timer (Wecker)

Implizites Laufzeitkonzept
Pool ankommender Signale und eingehender Operationsrufe

Objekt einer (aktiven) Klasse

Signale

sigA

Einheit
~ unklar in UML
unsere Sprechweise:
in SDL: Agent
In ODEMx: Ableitung der der ODEMx-Klasse Process

Port
(adressierbare Anschlusspunkte)
mit zugeordnetem **Interface**
(ein- u. ausgehende Signale u. Op-Rufe)

Zustandsmaschine
(Instanz einer Zustands-maschinenklasse)

Zustand eines SDL-Agenten zu einem Zeitpunkt:

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

Achtung (wird hier nicht behandelt):

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