Softwaretechnik
Model Driven Architecture
Metamodellierung

Prof. Dr. Peter Thiemann

Universität Freiburg

17.07.2008

Metamodellierung
Einführung

- Was?
  - meta = über
  - Definiert eine Ontologie von Konzepten für eine Domäne.
  - Definiert das Vokabular und die grammatischen Regeln einer Modellierungssprache.
  - Definiert eine domänenspezifische Sprache (DSL).
- Warum?
  - Spezifikation der Menge der Modelle für eine Domäne.
  - Präzise Definition der Modellierungssprache.
- Wie?
  - Grammatiken und Attributierungen für textbasierte Sprachen.
  - Metamodellierung generalisiert dies auf beliebige Sprachen (*z.B.*, grafische)

Metamodellierung
Verwendungen

- Konstruktion von DSLs
- Validierung von Modellen
  (testen gegen ein Metamodell)
- Model-to-model Transformation
  (definiert auf Grundlage eines Metamodells)
- Model-to-code Transformation
- Werkzeugintegration

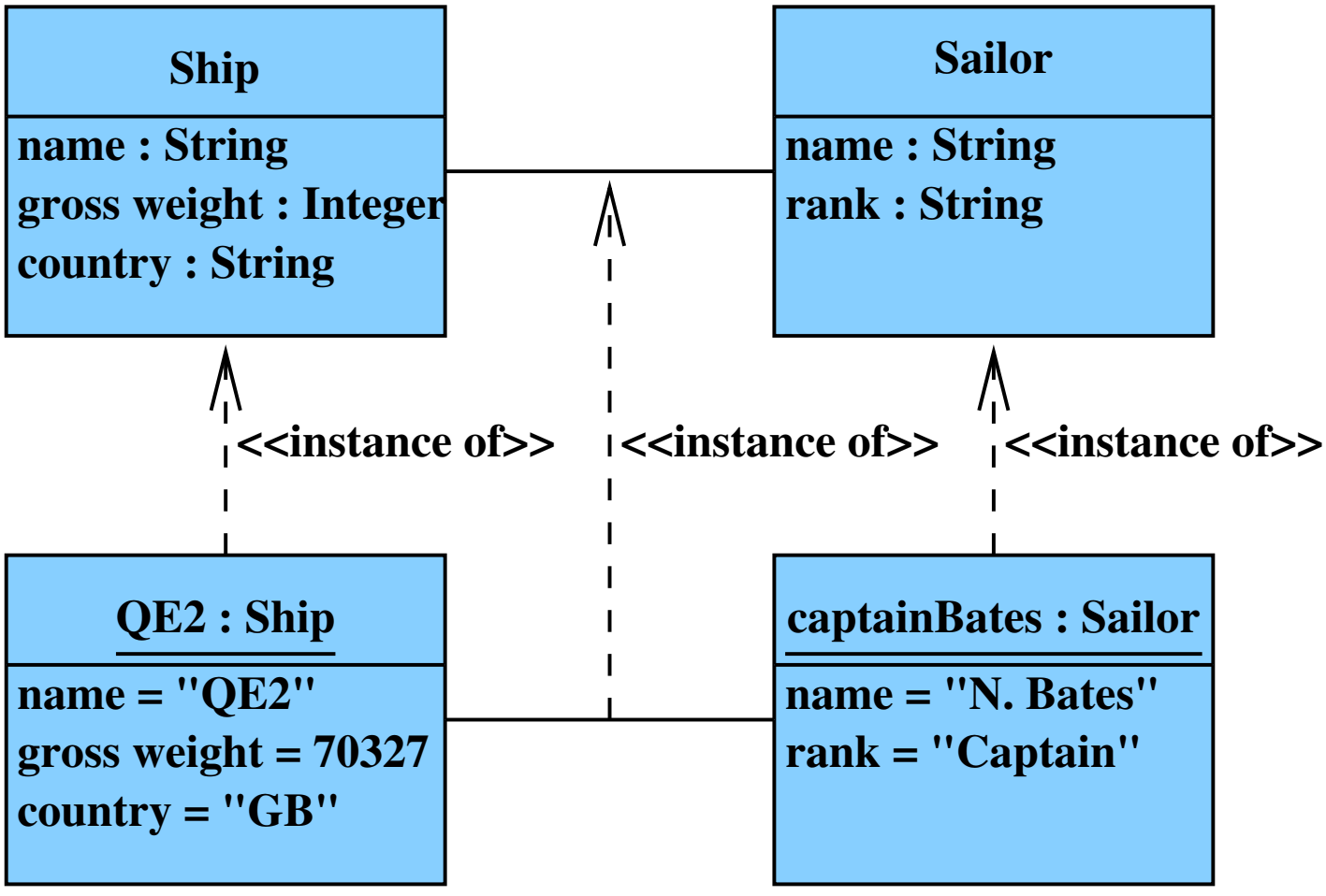
Exkursion: Classifiers und Instanzen

- Classifier-Diagramme dürfen auch Instanzen enthalten
- Beschreibung einer Instanz kann enthalten
  - Namen (optional)
  - Klassifikation durch beliebig viele Klassifikatoren
  - Art der Instanz
    - Instanz einer Klasse: Objekt
    - Instance einer Assoziation: Link
    - usw
  - Optional Spezifikation von Werten

Exkursion: Notation für Instanzen

- Instanzen verwenden die gleiche Notation wie Klassifikatoren
  - Rechteck für die Instanz
  - Namesabteil enthält
    name:*classifier*,*classifier*...
    name:*classifier*
    :*classifier* anonyme Instanz
    : unklassifizierte, anonyme Instanz
  - Attribut im Klassifikator veranlasst einen gleichnamigen Slot mit optionalem Wert
  - Assoziation mit dem Klassifikator veranlasst einen Link zum anderen Ende der Assoziation
    Richtung muss mit Navigierbarkeit verträglich sein

Exkursion: Notation für Instanzen (grafisch)



Terminologie/Syntax

Regeln für Wohlgeformtheit

- Abstrakte Syntax
  nur Struktur, wie werden Sprachkonzepte zusammengesetzt
- Konkrete Syntax
  definiert spezifische Notation
- Typische Verwendung:
  ein Parser bildet konkrete Syntax in abstrakte Syntax ab

Terminologie/Abstrakte Syntax
Beispiel: Arithmetische Ausdrücke

- Abstrakte Syntax

```
Expr = Const String
     | Var String
     | Binop Op Expr Expr
Op   = Add | Sub | Mul | Div

Binop Mul (Const "2")
          (Binop Add (Var "x") (Const "3"))
```

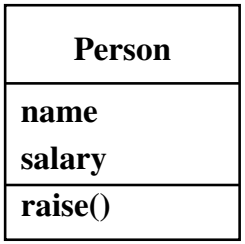
- Konkrete Syntax

$$
\begin{aligned}
E &::= \mathtt{c} \mid \mathtt{x} \mid E\ B\ E \mid (E) \\
B &::= + \mid - \mid * \mid /
\end{aligned}
$$

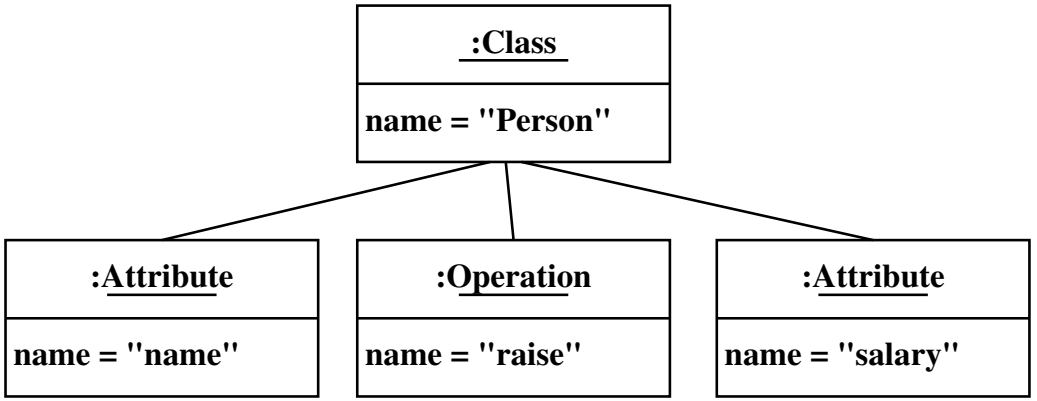
```
2 * (x + 3)
```

Terminologie/Abstrakte Syntax
Beispiel: UML class diagram

- Konkrete Syntax



- Abstrakte Syntax



Terminologie/Statische Semantik

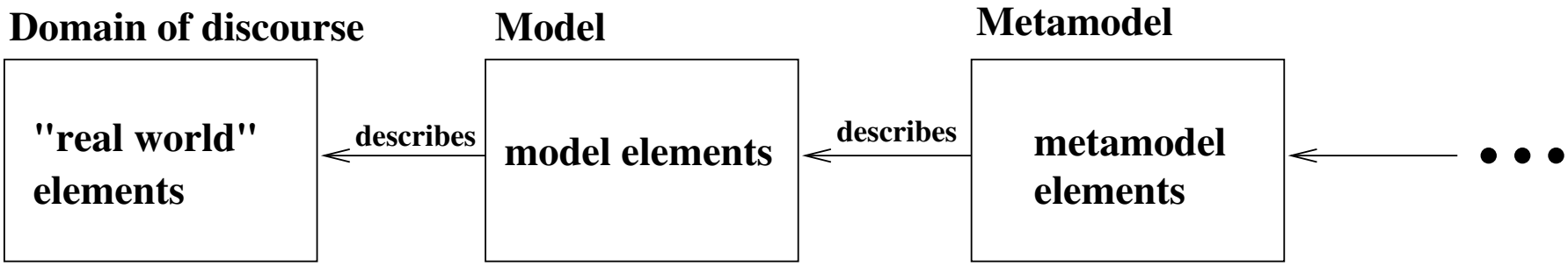
- Statische Semantik definiert Wohlgeformtheitsregeln, die über die Syntax hinausgehen
- Beispiele
  - “Variablen müssen vor ihrer Verwendung definiert werden
  - Typesystem einer Programmiersprache
    `"hello" * 4` ist syntaktisch korrektes Java, wird aber zurückgewiesen
- UML: statische Semantik via OCL Ausdrücke
- Verwendung: Erkennung von Fehlern in der Modellierung bzw Transformation

Terminologie/Domänenspezifische Sprache (DSL)

- Zweck: formale Beschreibung von Schlüsselaspekten einer Domäne
- Metamodell einer DSL definiert abstrakte Syntax und statische Semantik
- Zusätzlich:
  - konkrete Syntax (nah an der Domäne)
  - dynamische Semantik
    - fürs Verständnis
    - für automatische Werkzeuge
- Verschiedene Komplexitätsgrade möglich
  Konfigurationsoptionen mit Gültigkeitsprüfung
  grafische DSL mit domänenspezifischem Editor

Modell und Metamodell

Modell und Metamodell

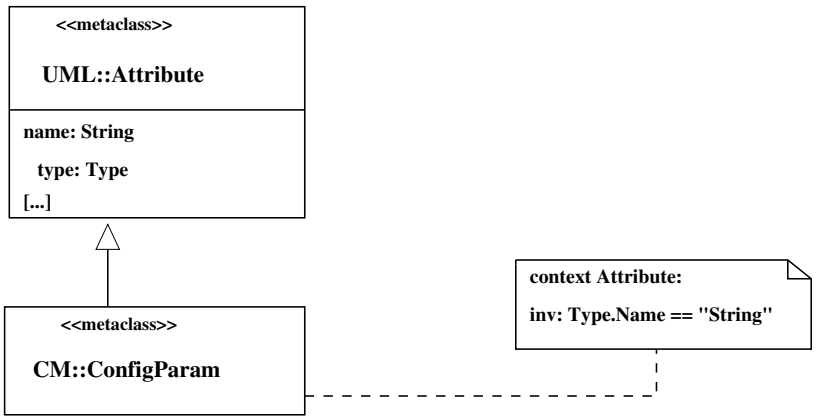


- Einsicht: **Jedes Modell ist Instanz eines Metamodells.**
- Essentiell: *instance-of* Beziehung
- Zu jedem Element muss es ein klassifizierendes Metaelement geben,
  - das die Metadaten enthält und
  - das vom Element erreichbar ist
- Relation Model:Metamodel ist wie Object:Class
- Definition des Metamodell durch ein Meta-metamodell
- $\Rightarrow$ unendlicher Turm von Metamodellen
- $\Rightarrow$ “meta” Relation ist immer relativ zu einer Modellebene

Metamodellierung à la OMG

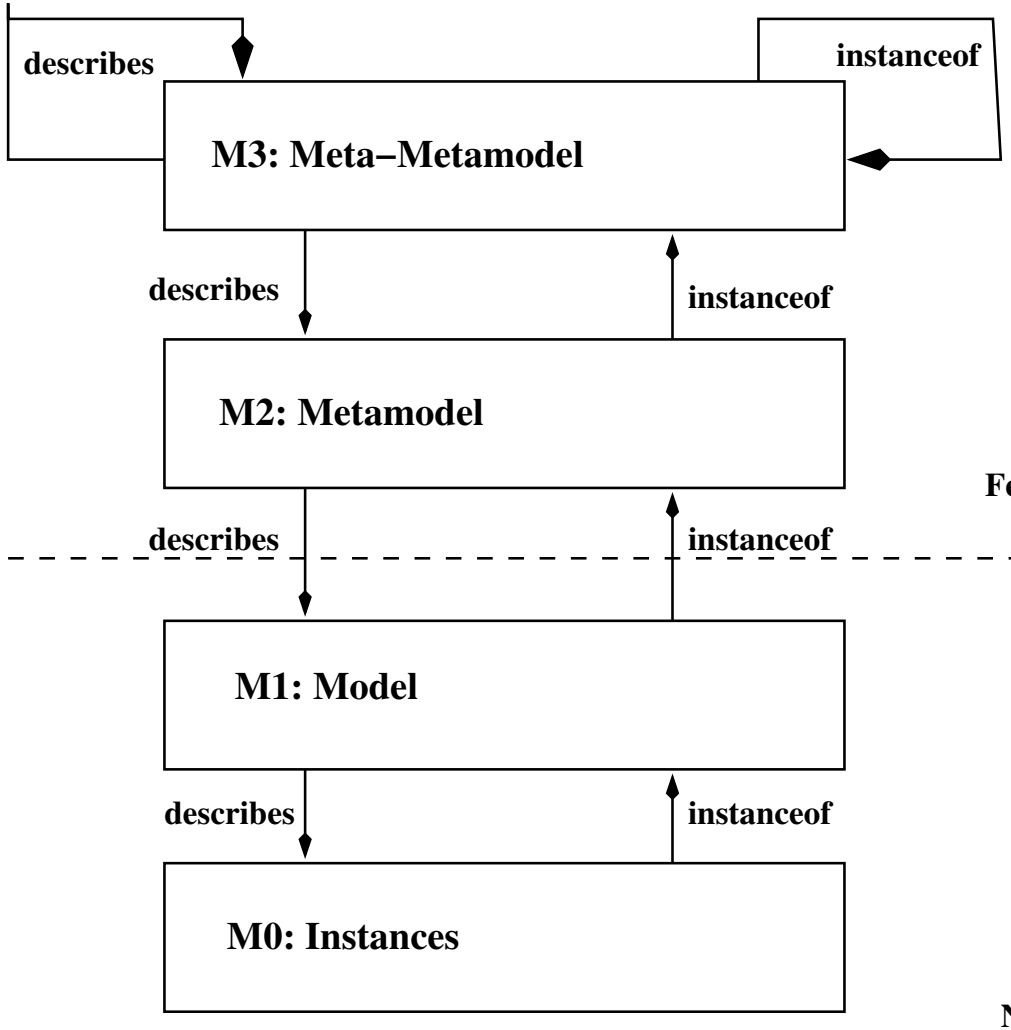
- OMG definiert einen Standard (MOF) für die Metamodellierung
- MOF (Meta-Object Facility) verwendet zur Definition von UML
- Achtung!
  - MOF und UML verwenden die gleiche Syntax (classifier und Instanzdiagramme)
  - MOF verwendet die gleichen Namen für Modellierungselementen wie UML (*e.g.*, Klasse)
- Ansatz
  - Einschränken der unendlich vielen Metaebenen auf vier
  - Die letzte Ebene ist “selbst-beschreibend”

Metamodellierung und OCL



- OCL Constraints sind unabhängig von der Modellierungssprache und der Metaebene
- OCL auf Ebene M$n+1$ restringiert Instanzen auf Ebene Mn

Die vier OMG-Metaebenen



Typ: Classifier
ID: 5346456
Name: Classifier

Typ: Classifier
ID: 764535
Name: Klasse
Features: Attributes, Operations, Ass

Typ: Klasse
ID: 21436456
Name: Person
Attribute: Name, Firstn.
Operations: ...
Association: ...

Typ: Person
ID: 05034503
Name: Doe
Given name: John

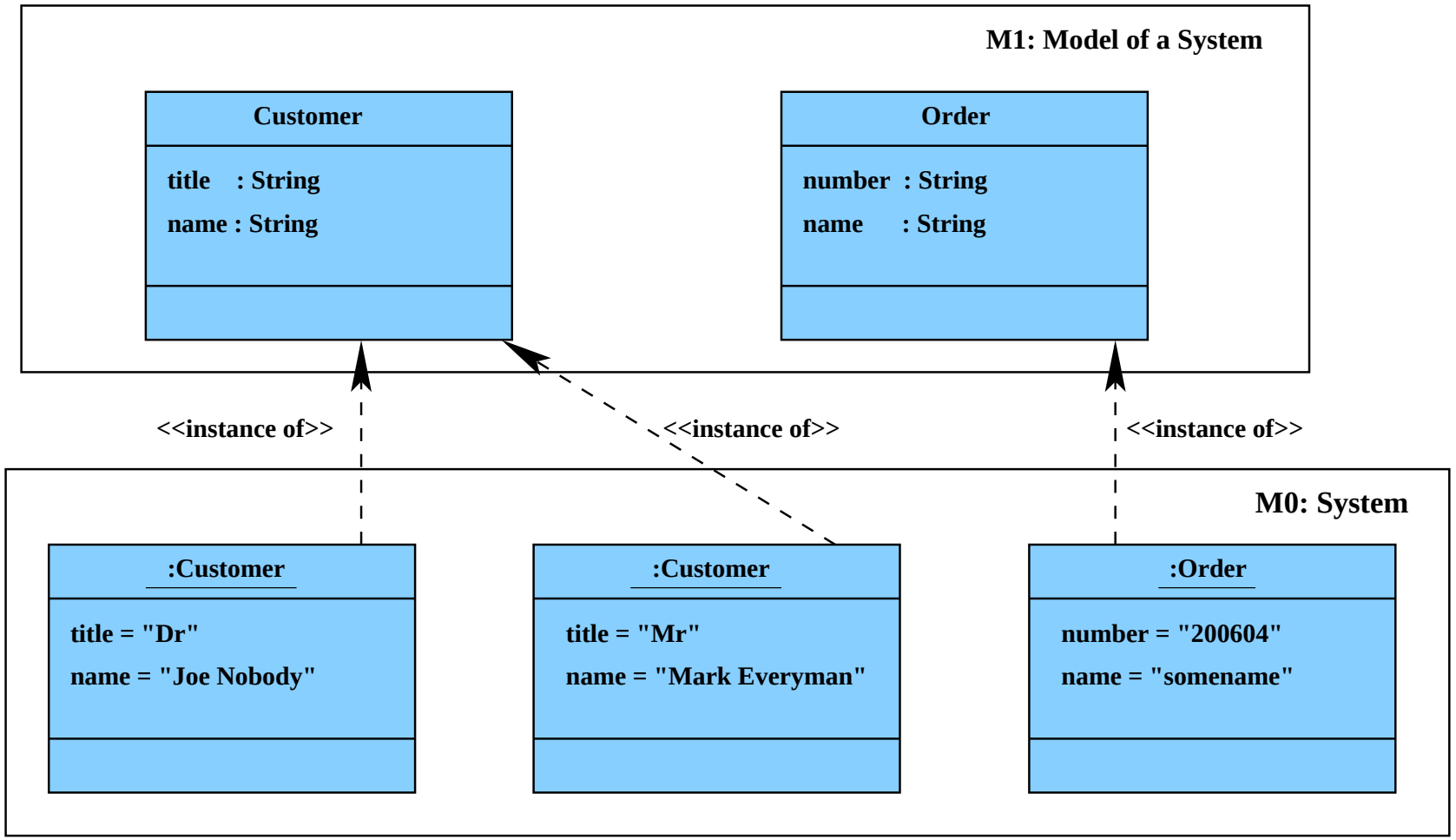
Ebene M0: Instanzen

- Ebene des ablaufenden Systems
- Enthält echte Objekte, *z.B.*, Kunden, Seminare, Bankkonten, mit gefüllten Slots für Attribute usw
- Beispiel: Objektdiagramm

Ebene M1: Modell

- Ebene der Systemmodelle
- Beispiel:
  - UML Modell eines Softwaresystems
  - Klassendiagramm enthät Modellierungselemente: Klassen, Attribute, Operationen, Assoziationen, Generalisierungen, ...
- Elemente von M1 kategorisieren Elemente auf Ebene M0
- Jedes M0-Element ist Instanz eines M1-Elements
- Keine weiteren Instanzen sind auf Ebene M0 zulässig

Verhältnis zwischen M0 und M1

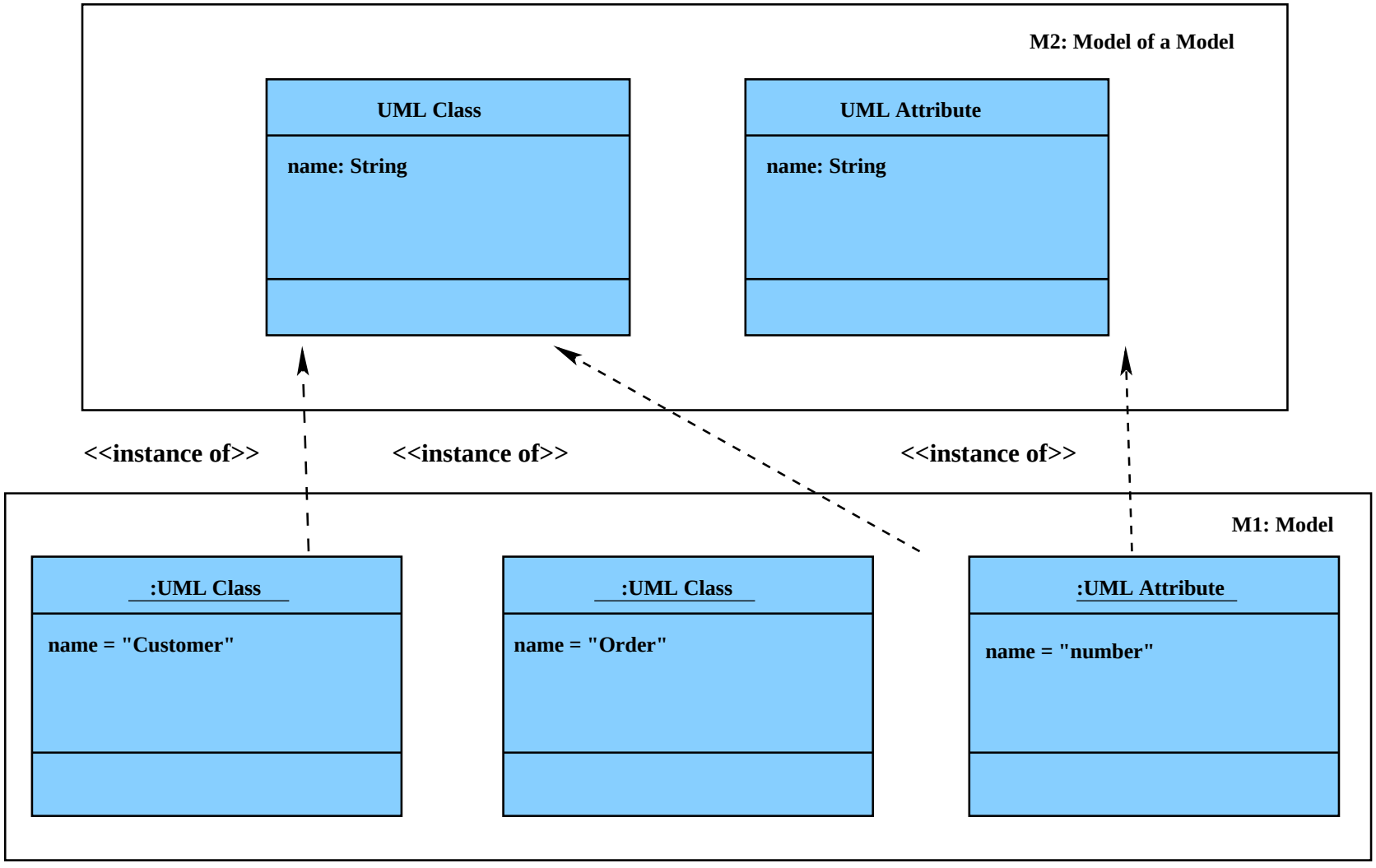


Ebene M2: Metamodell

"Modell fürs Modell"

- Ebene, auf der Modellierungselemente definiert werden
- Konzepte von M2 kategorisieren Instanzen auf Ebene M1
- M2-Elemente kategorisieren M1-Elemente: Klassen, Attribute, Operationen, Assoziationen, Generalisierungen, ...
- Beispiele
  - Jede Klasse in M1 ist Instanz eines Klassen-beschreibenden Elements auf Ebene M2 (d.h., eine *Metaklasse*)
  - Jede Assoziation in M1 ist Instanz eines Assoziations-beschreibenden Elements auf Ebene M2 (eine *Metaassoziation*)
  - und so weiter

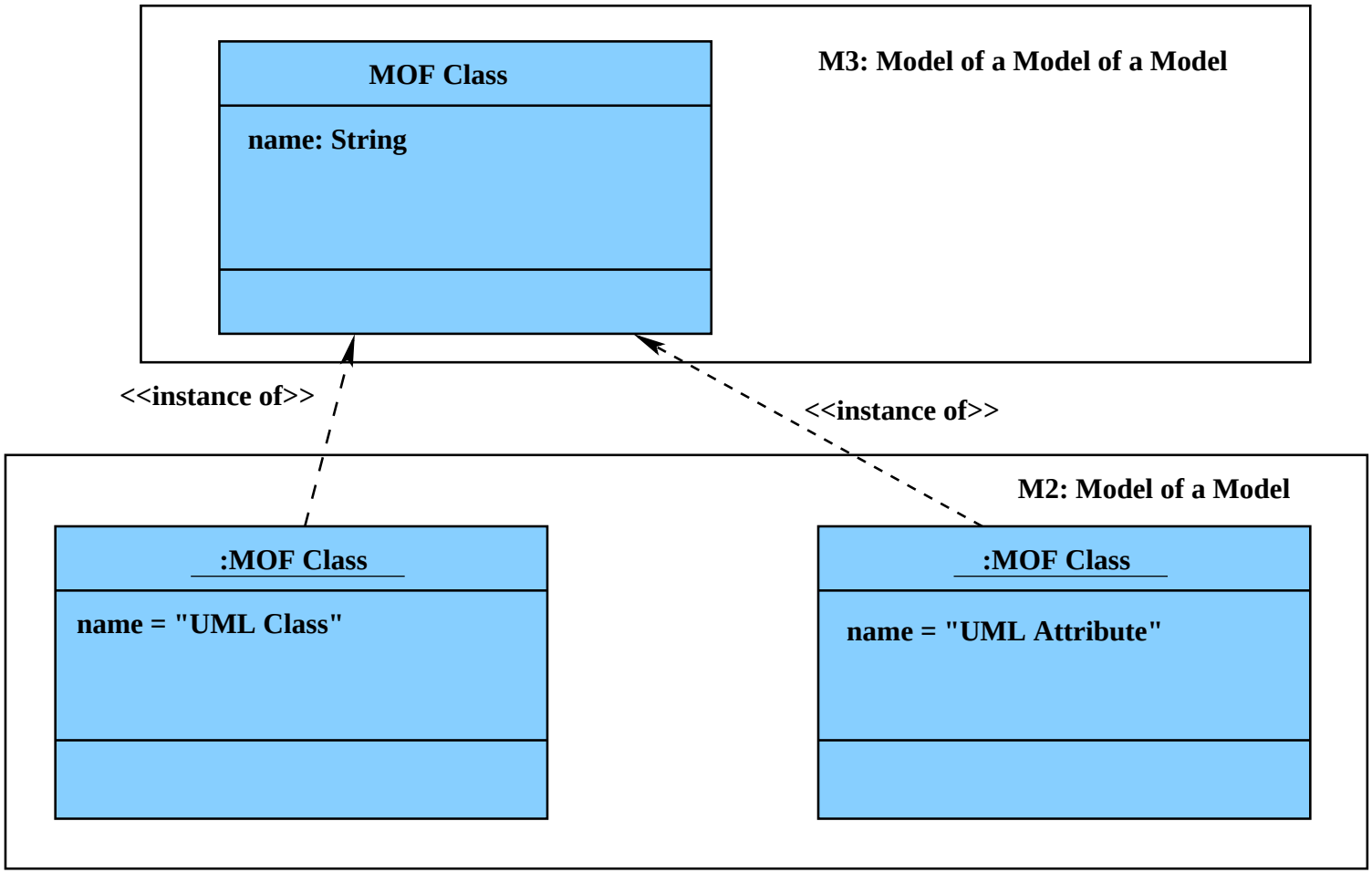
Verhältnis zwischen M1 und M2



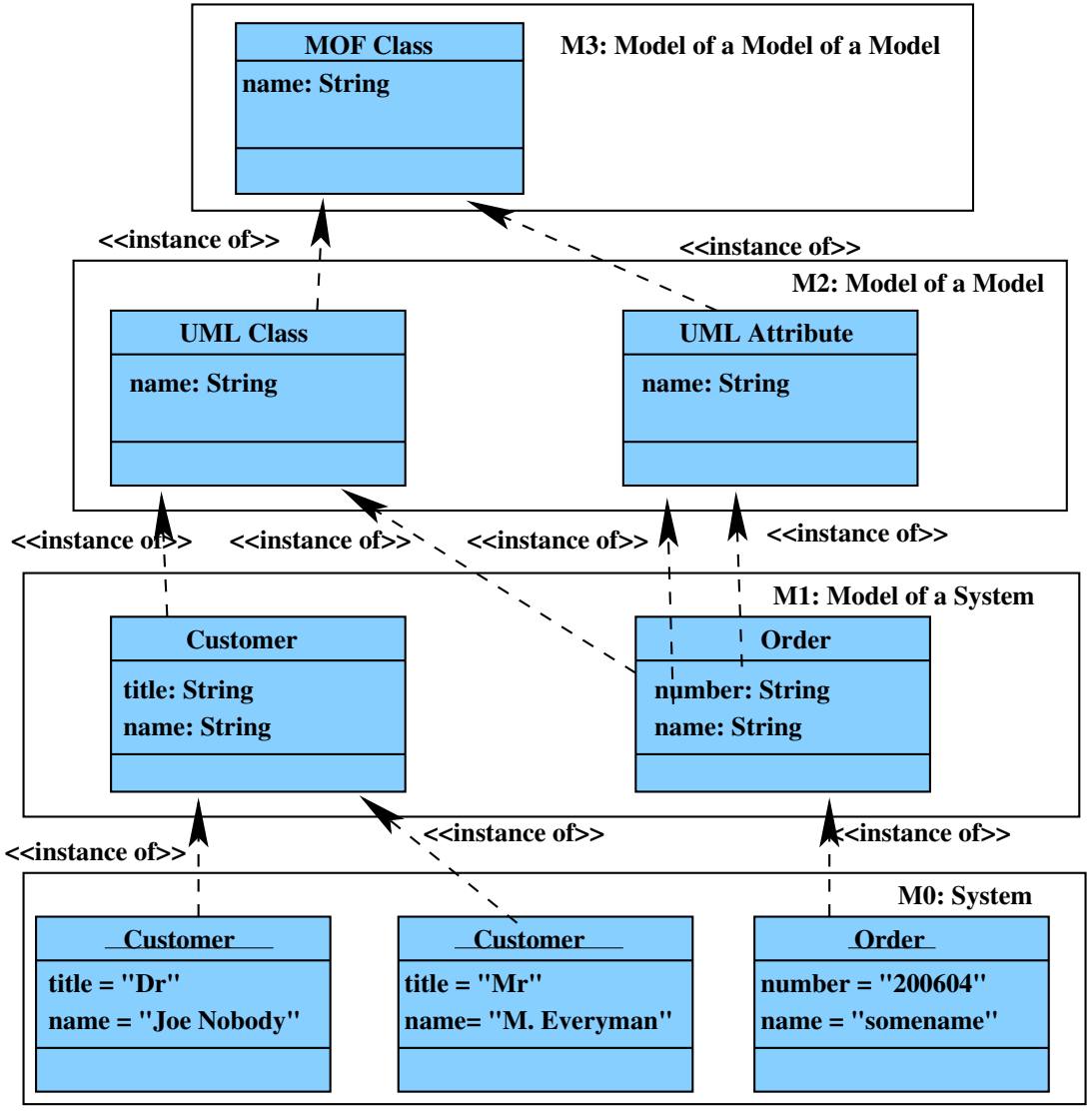
Ebene M3: Meta-Metamodell

- Ebene zur Definition der Definition der Modellierungselemente
- M3-Elemente kategorisieren M2-Elemente: Metaklassen, Metaassoziationen, Metaattribute, etc
- Typisches Element eines M3 Modells: MOF-Klasse
- Beispiele
  - Die Metaklassen Class, Association, Attribute, usw sind alle Instanzen von MOF::Class
- Die Ebene M3 ist selbst-beschreibend

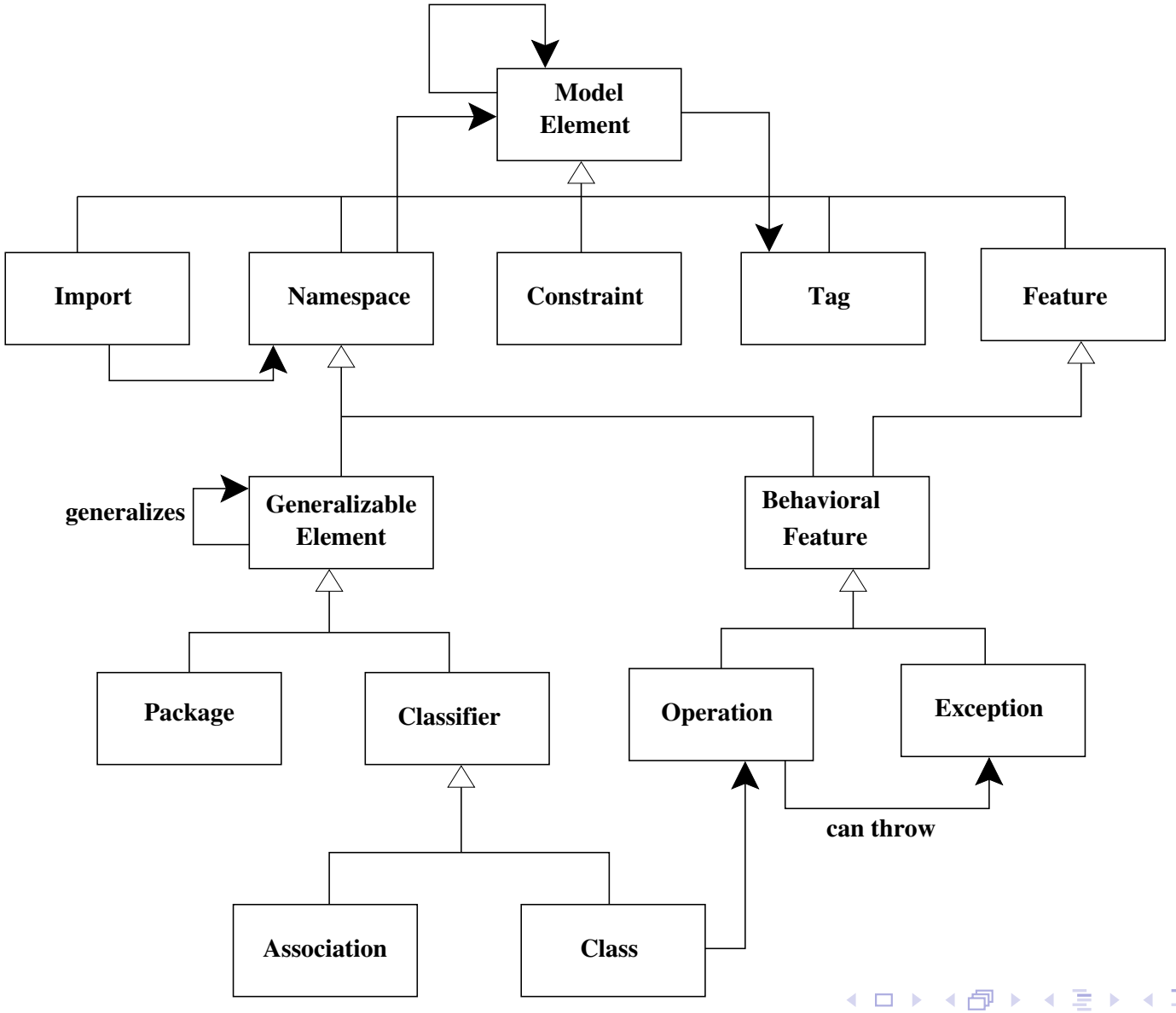
Verhältnis zwischen M2 und M3



Übersicht über die Ebenen



Auszug aus MOF/UML

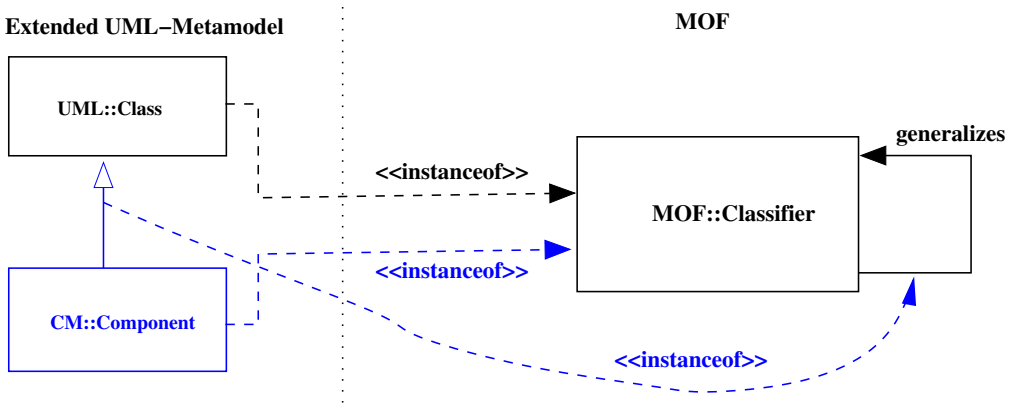


Erweitern von UML
Entwurf einer DSL

Entwurf einer DSL

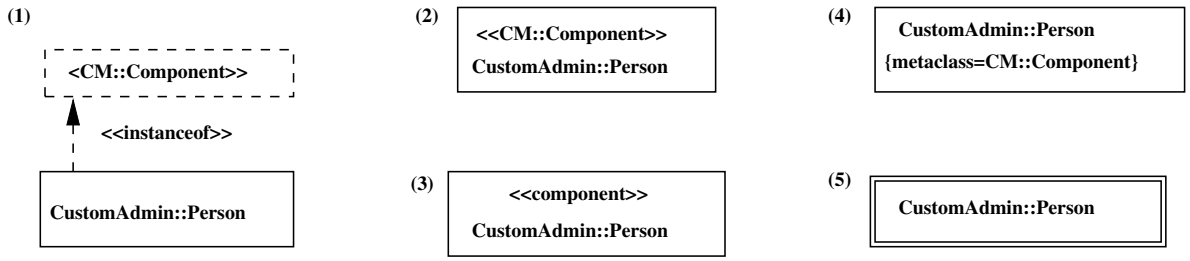
- Definition einer neuen M2-Sprach von Null ist zu aufwändig
- Typischer Ansatz: Erweiterung von UML
- Erweiterungsmechanismen
  - Erweiterung des UML 2 Metamodells
    anwendbar für alle MOF-definierten Metamodelle
  - Erweiterung mit Stereotypen (à la UML 1.x)
  - Erweiterung mit Profilen (à la UML 2)

Erweitern des UML-Metamodells



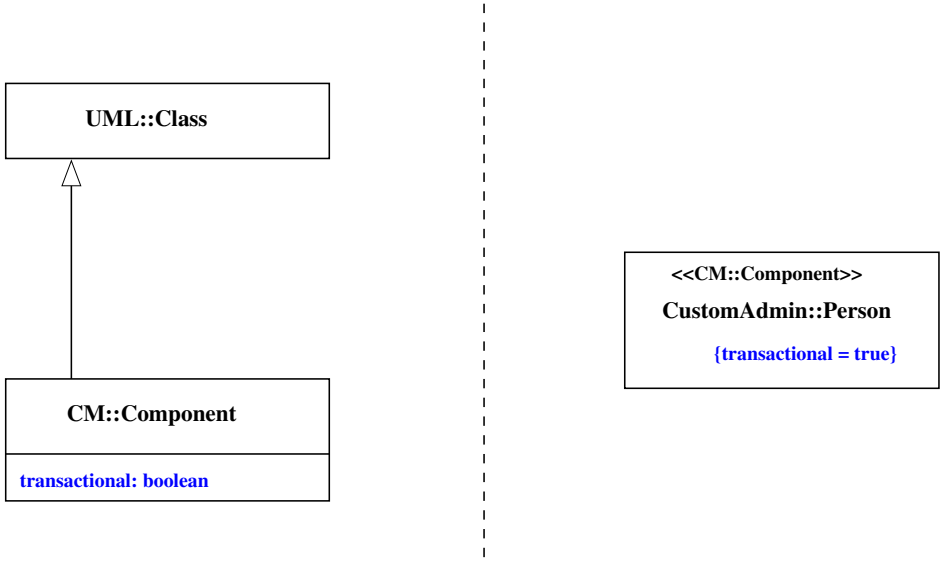
- MOF erlaubt die Ableitung neuer Metaklassen **CM::Component** von **UML::Class**
- **CM::Component** ist Instanz von **MOF::Classifier**
- Generalisierung ist Instanz von MOFs **generalizes** Assoziation

Erweitern des UML-Metamodells/Konkrete Syntax



1. Explizite Instanz einer Metaklasse
2. Name der Metaklasse als Stereotyp
3. Konvention
4. Tagged value mit Metaklasse
5. Eigene grafische Repräsentation (falls unterstützt)

Erweitern einer Klasse

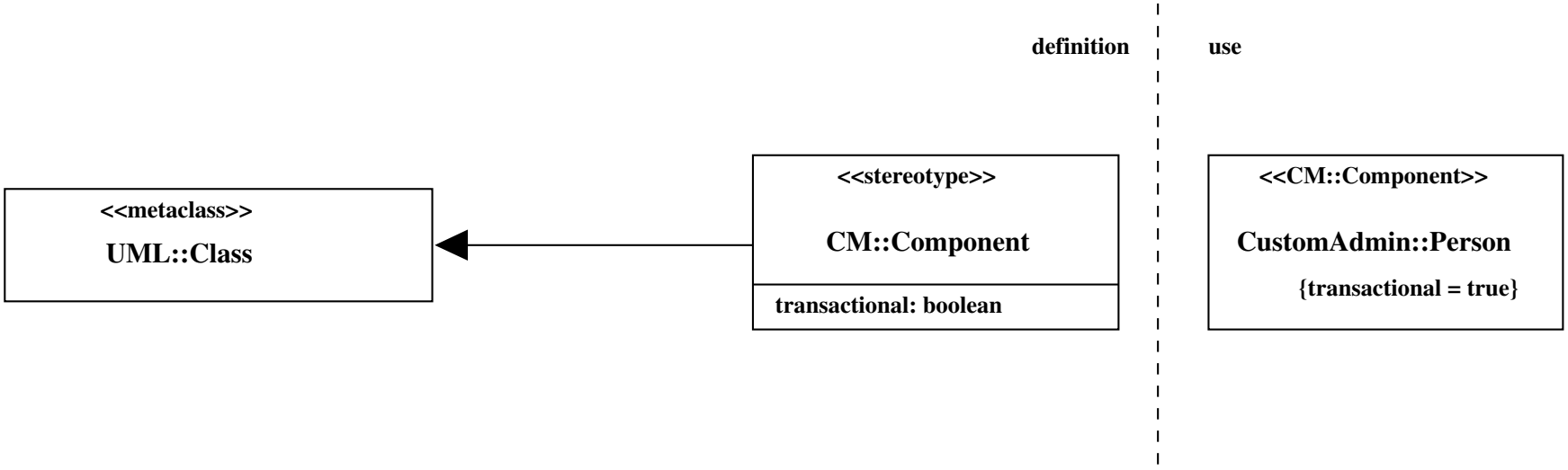


- das reine Erben von **UML::Class** liefert eine identische Kopie
- Hinzufügen eines Attributs zur **CM::Component** Metaklasse führt zu
  - einem Slot für einen Attributwert in jeder Instanz
  - Notation: tagged value (getypt in UML 2)

Erweitern mit Stereotypen (UML 1.x)

- Einfacher Specialisierungsmechanismus von UML
- Kein Rückgriff auf MOF erforderlich
- Tagged Values, aber ungetypt
- Keine neuen Metaassoziationen möglich

Erweitern mit Profile (UML 2)

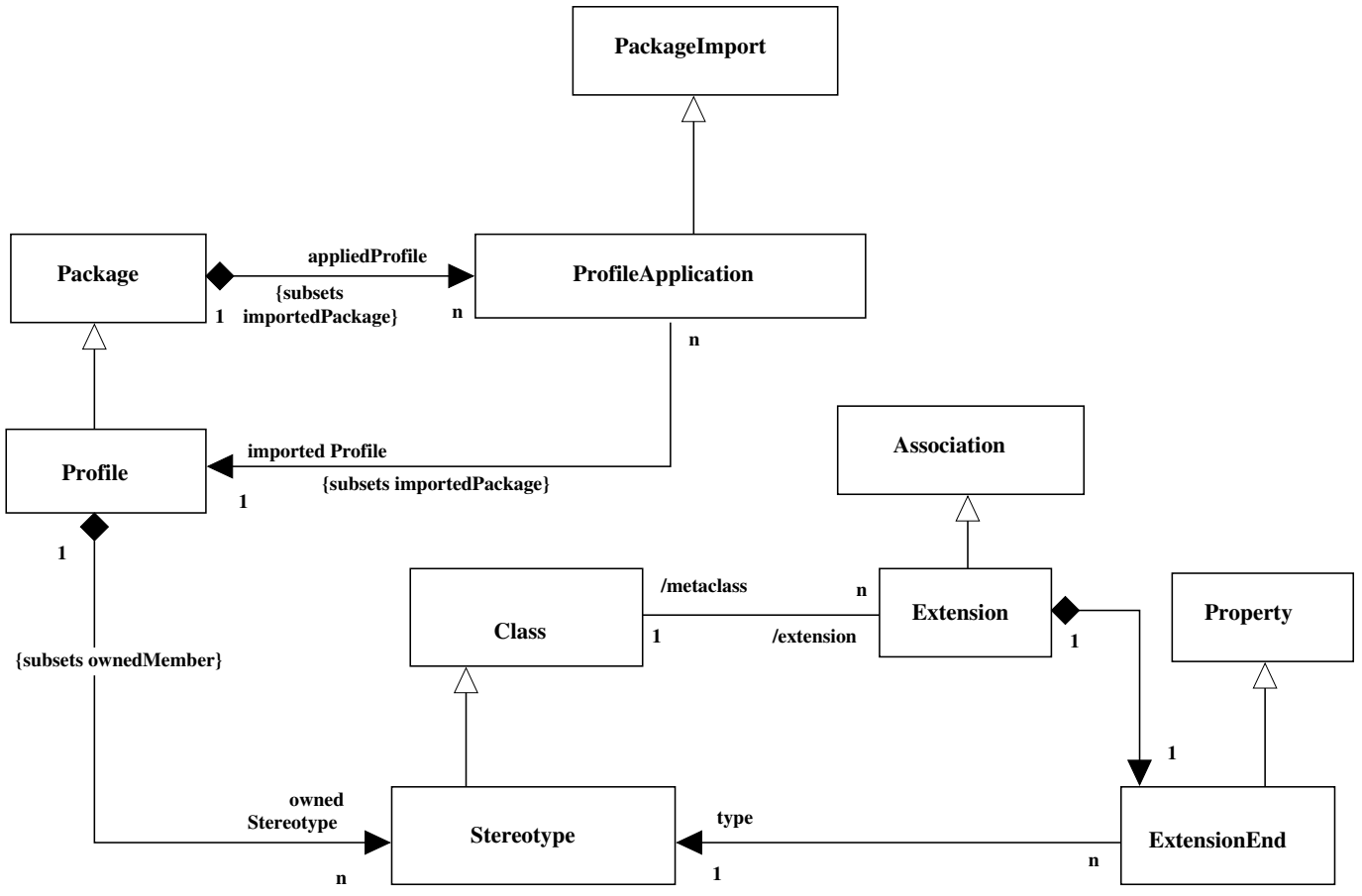


- Erweiterung des Stereotype-Mechanismus
- Erfordert den “Erweiterungspfeil” als neues UML Sprachkonstrukt (Generalisierungspfeil mit gefülltem Kopf)
- Nicht: Generalisierung, Implementierung, Abhängigkeit mit Stereotyp, Assoziation, . . .
- Attribute $\Rightarrow$ getypte tagged values
- Mehrere Stereotypen möglich

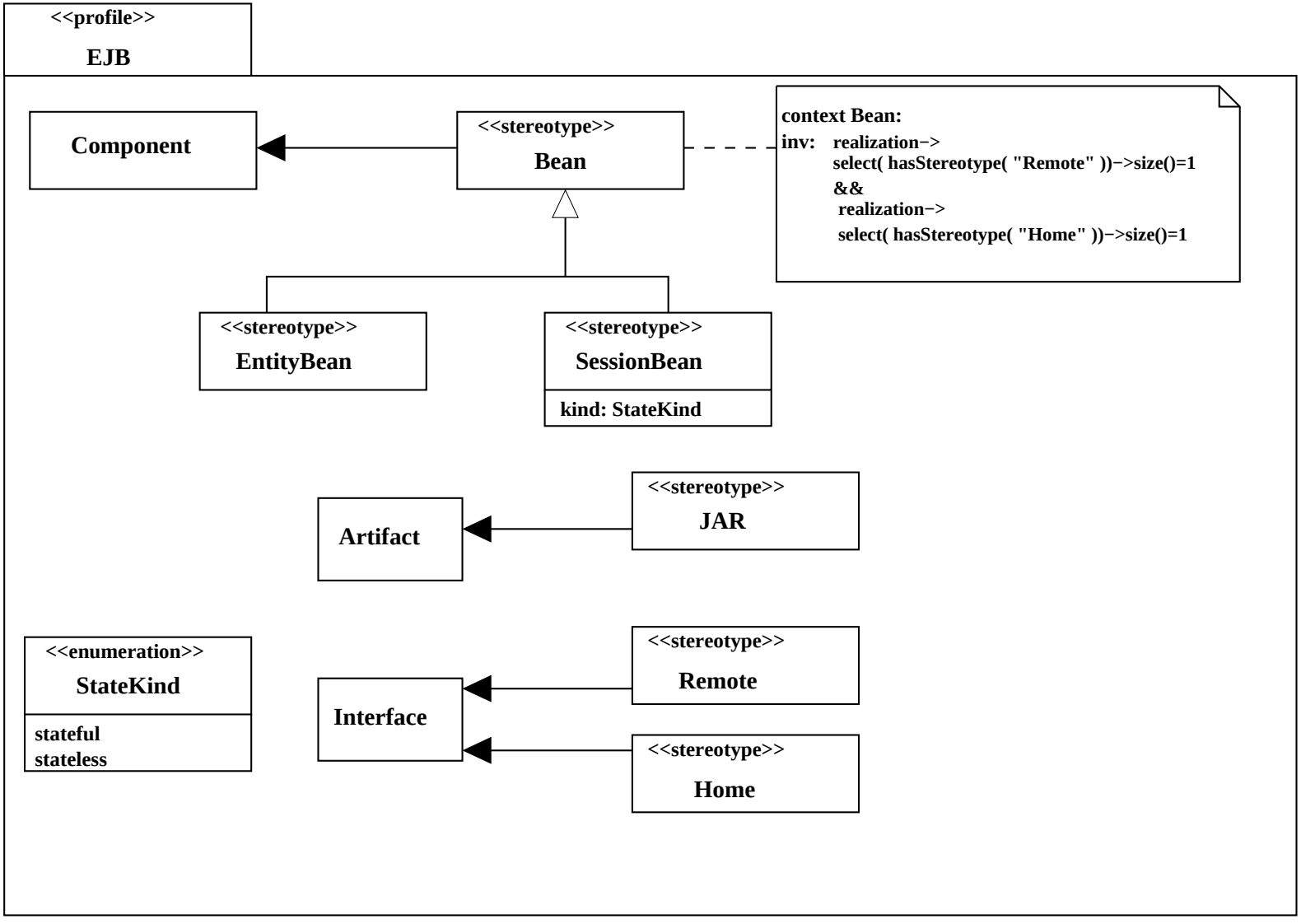
Mehr über Profile

- Profile machen aus UML eine Sprachfamilie
- Jedes Mitglied ist definiert durch Anwendung von einem oder mehreren Profilen auf das Basis-UML-Metamodell
- Werkzeuge sollte Profile und zugehörige Transformationen laden und verarbeiten können
- Profile haben drei Bestandteile
  - Stereotypen
  - Tagged values
  - Constraints
- Profile können nur existierenden Modellierungselementen weiter Restriktionen auferlegen
- Profile sind formal definiert durch ein Metamodell

Metamodell für Profile



Beispiel: Profil für EJB



Weitere Aspekte von Profilen

- Stereotypen können von anderen Stereotypen erben
- Stereotypen können abstrakt sein
- Constraints eines Stereotyps gelten für den stereotypisierten Klassifikator
- Profile sind relative zu einem Referenz-Metamodell zu verstehen
  z.B., zum UML-Metamodell oder zu einem existierenden Profil
- Bislang unterstützen wenige Werkzeuge Profil-basierte Modellierung, warum sollte man sie verwenden?
  - Constraints als Dokumentation
  - Spezialisierte UML-Werkzeuge
  - Validierung durch Transformation bzw Programmgenerator