Vorlesung

Methodische Grundlagen des Software-Engineering

im Sommersemester 2014

Prof. Dr. Jan Jürjens

TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

Teil 3.2: Metamodellierung

v. 25.06.2014

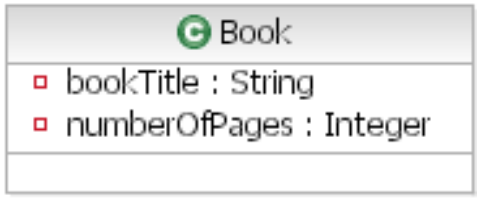
UML: Von informellem Modell zu Werkzeug

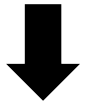
UML-Modelle: bislang

- **informell definiert** (textuelle Beschreibung)
- und **verwendet** (Skizzen auf Papier).

Aber: **UML kann mehr** !

- Codegenerierung
- Testfallgenerierung
- Modellvalidierung
- Modelltransformation
- …





Generiere Getters & Setters !

```
public String getBookTitle(){
  return bookTitle;
}
public void setBookTitle(String input){
  bookTitle = input;
}
...
```

Brauchen dafür Werkzeuge, die UML „verstehen“.

➔ **Formelle Definition** der Notation.

Wie UML werkzeugkonform definieren ?

Wie UML-Notation **„formell“ definieren** ?

➔ Verschiedene Möglichkeiten:
BNF-Grammatik, XML-Schema, …

Wie würde es jemand tun, der **nur UML kennt** ?

➔ **Zentrales Konzept** in UML identifizieren;
UML-Definition damit **„boot-strappen“** !

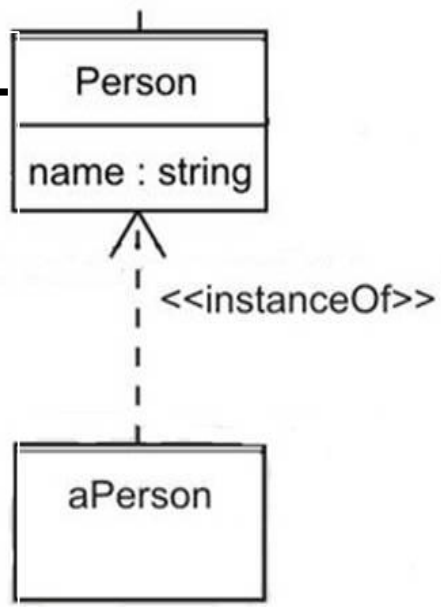
Zentrales Konzept in OO, um **Mengen von Entitäten** zu definieren ?

Zentrales OO-Konzept: Instanz-Beziehung

Klasse *Person* im **UML-Modell** der **UML** (**Metamodell**):
definiert Menge der **UML-Klassen** (**Modelle**).

Modell: **Modell** eines **Systems**:
definiert Menge der **Objekte** (z.B. *aPerson*, *bPerson*, ...).

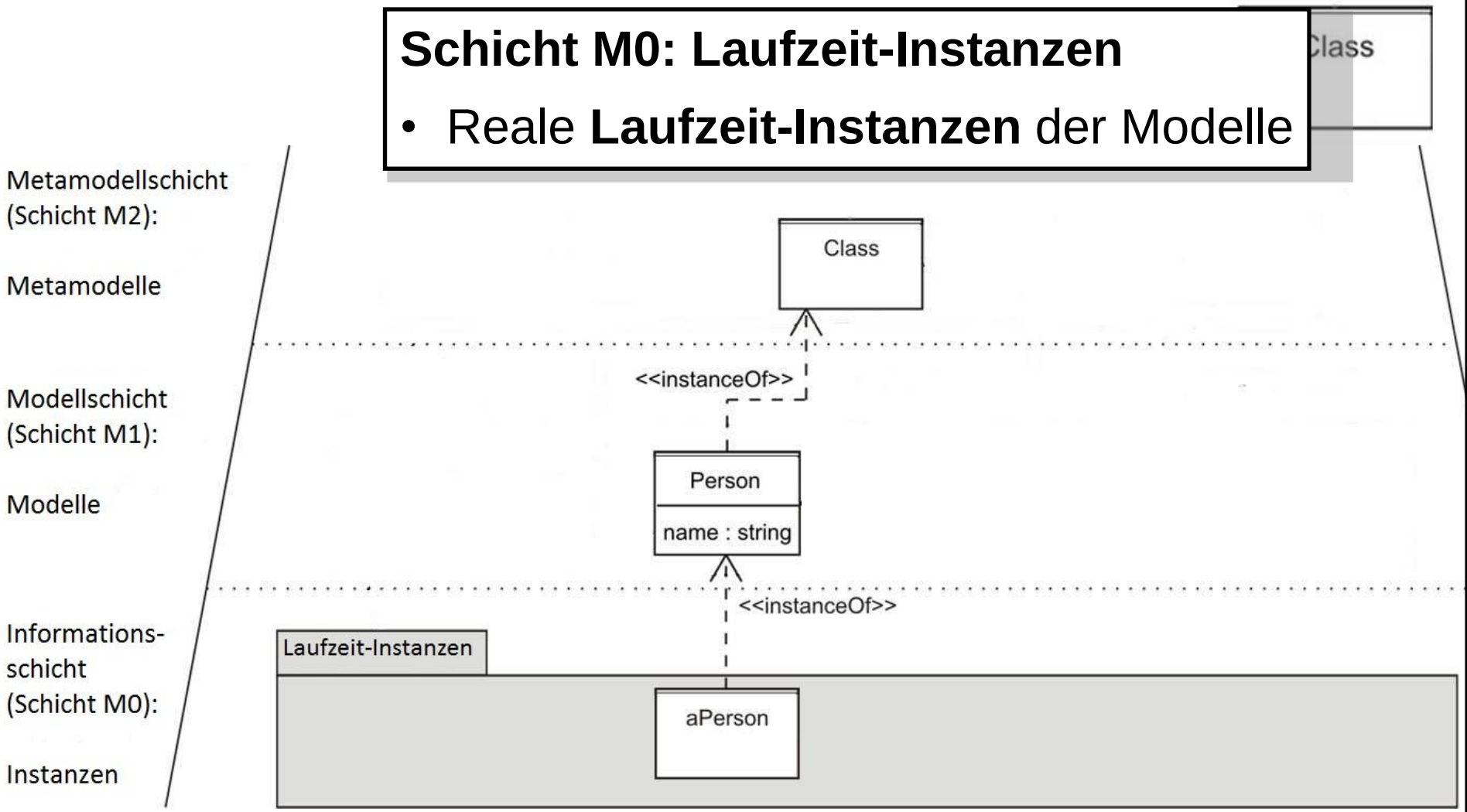
Metamodell: **Modell**, das **Notation** einer Modellierungsnotation **definiert**.



UML-Metamodell-Hierarchie: Schicht M0

Schicht M0: Laufzeit-Instanzen

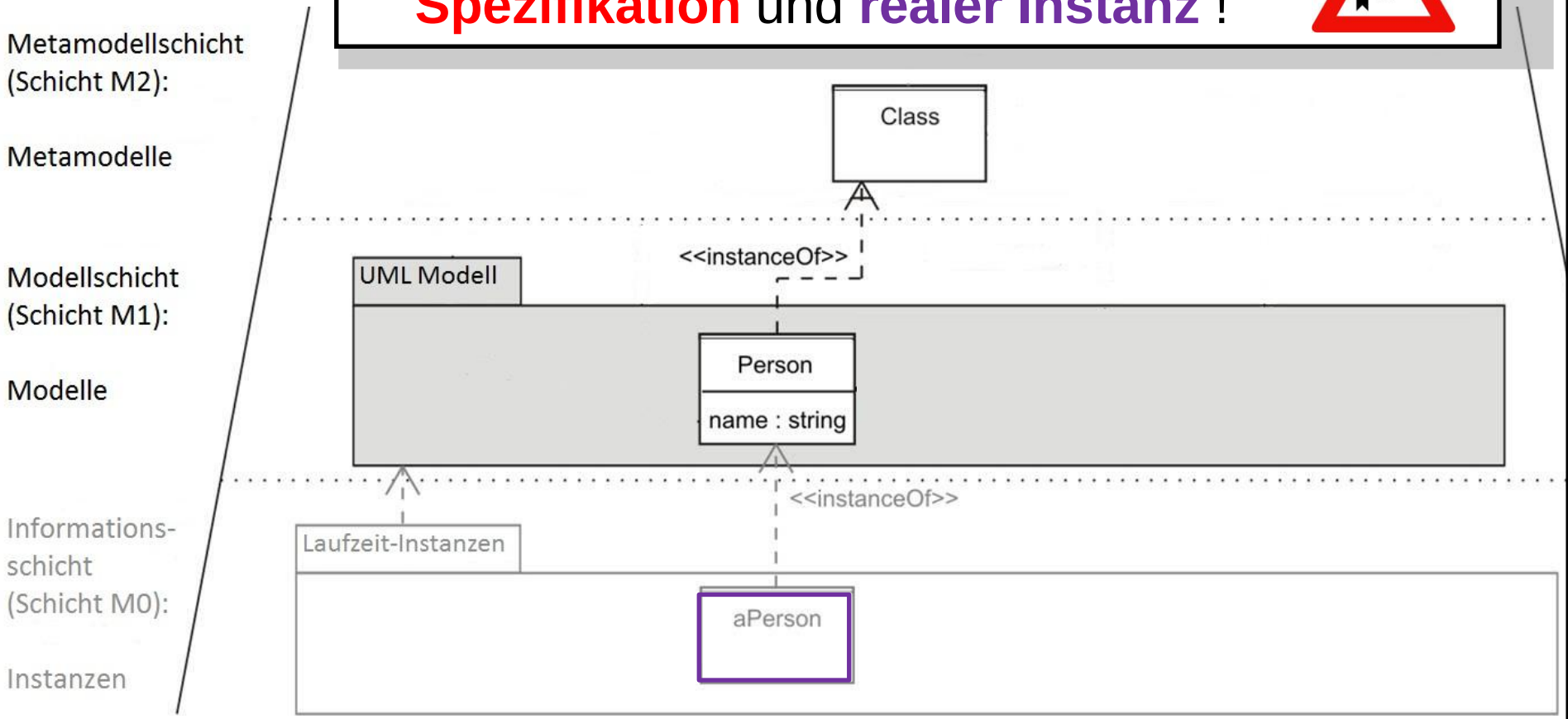
- Reale **Laufzeit-Instanzen** der Modelle



UML-Met

Schicht M1: UML-Modell

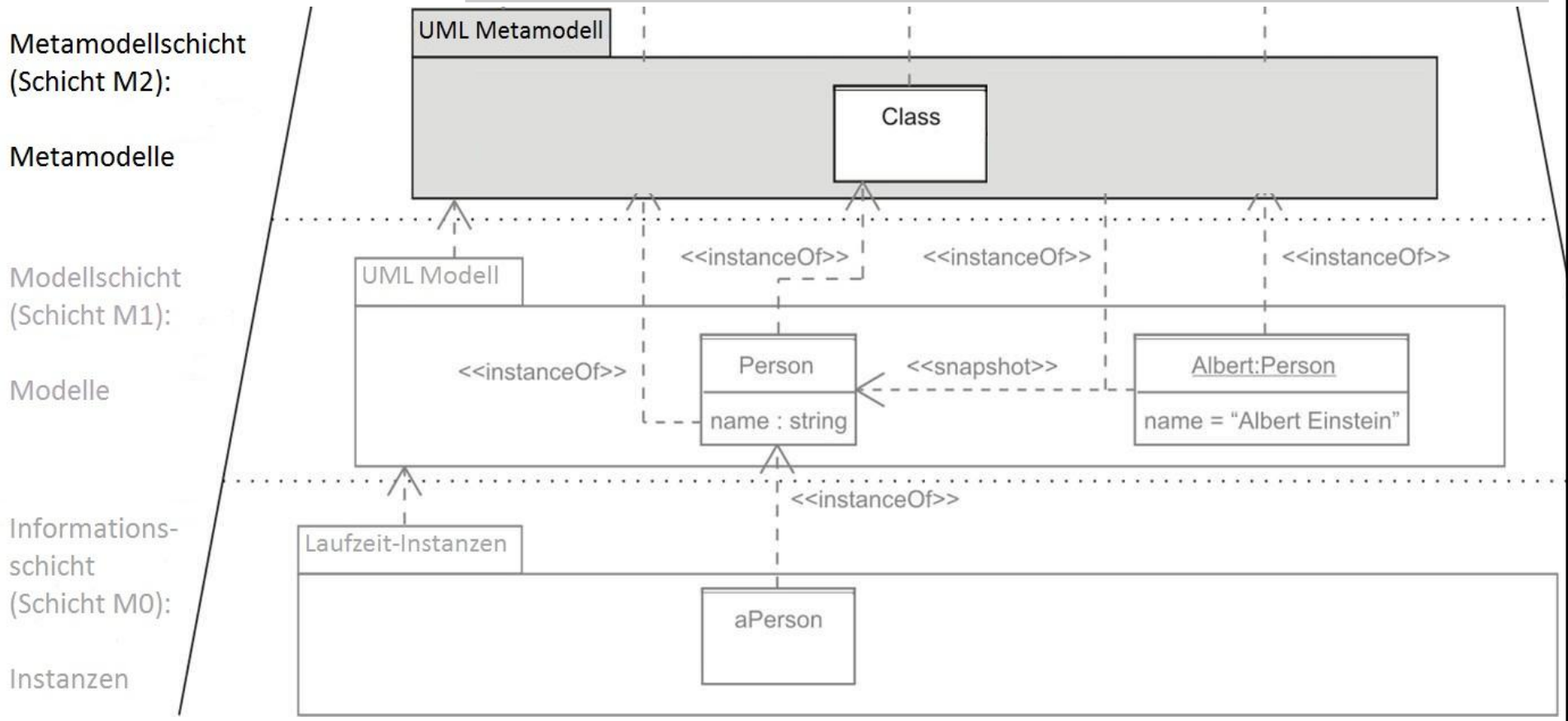
- **Instanziiert** aus **UML-Metamodell**.
- **Definiert** Menge **valider Laufzeit-Instanzen**.
- Unterschied zwischen **Instanz-Spezifikation** und **realer Instanz** !



UML-Metamo

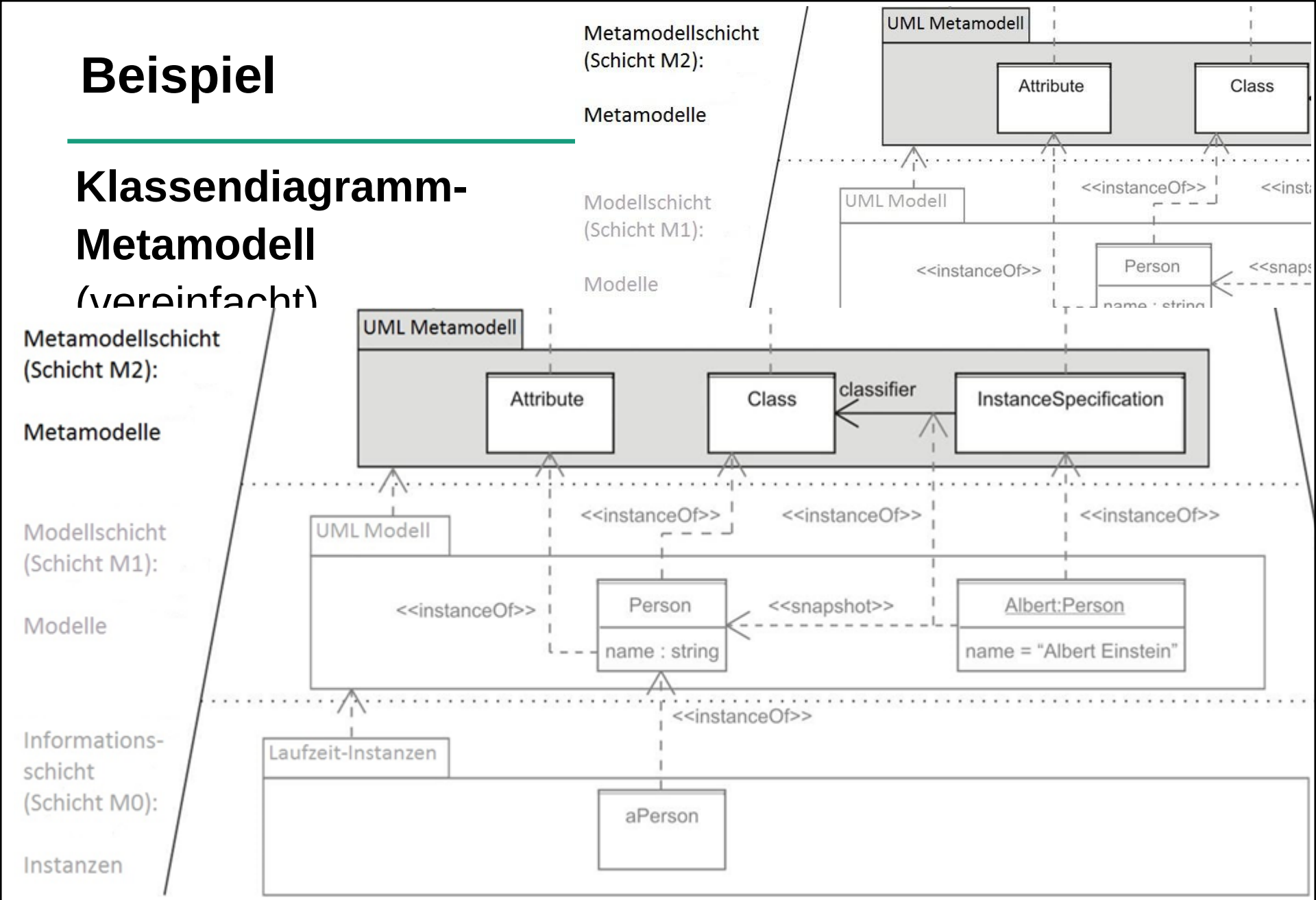
Schicht M2: UML-Metamodellierung

- **Definiert** UML-Notation, d.h. Menge **valider UML-Modelle**.
- Insbesondere darin verwendete Konzepte wie *Klassen, Attribute, Assoziationen.*



Beispiel

Klassendiagramm-Metamodell (vereinfacht)

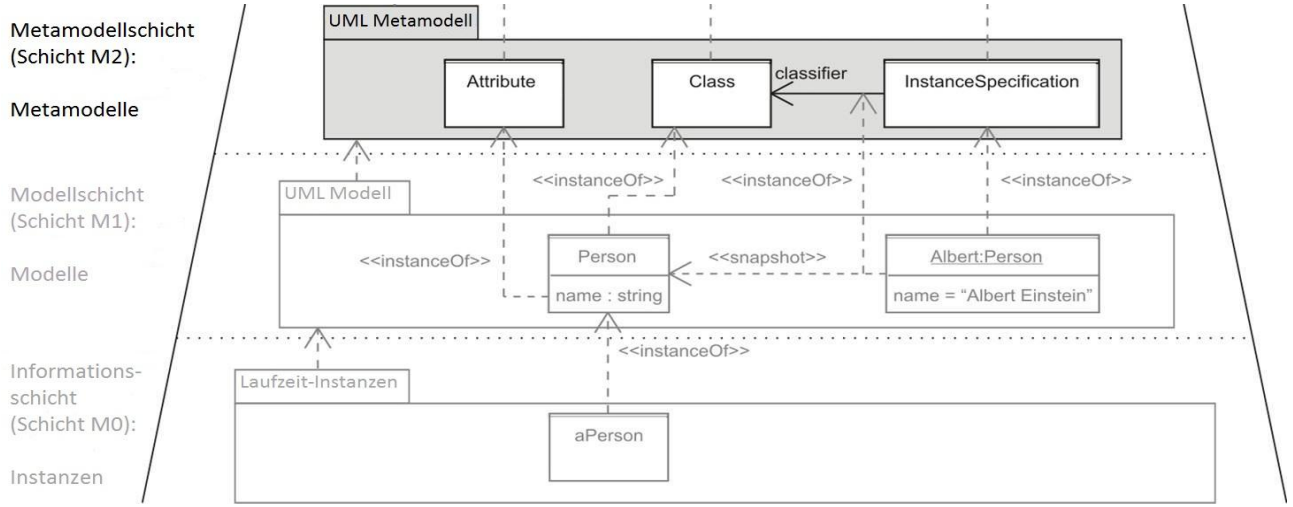


Sind wir jetzt fertig ?

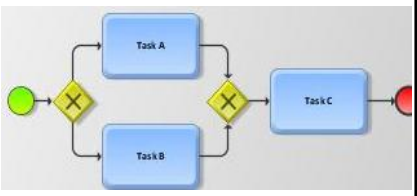
OK, Metamodelle sind cool – sind wir jetzt fertig ?

Könnten hier aufhören:

- **Klassendiagramme** grundlegendes Konzept,
- Übrige **UML-Syntax** damit **definieren**:



Metamodelle definieren ?



Aber: UML ist nicht alles !

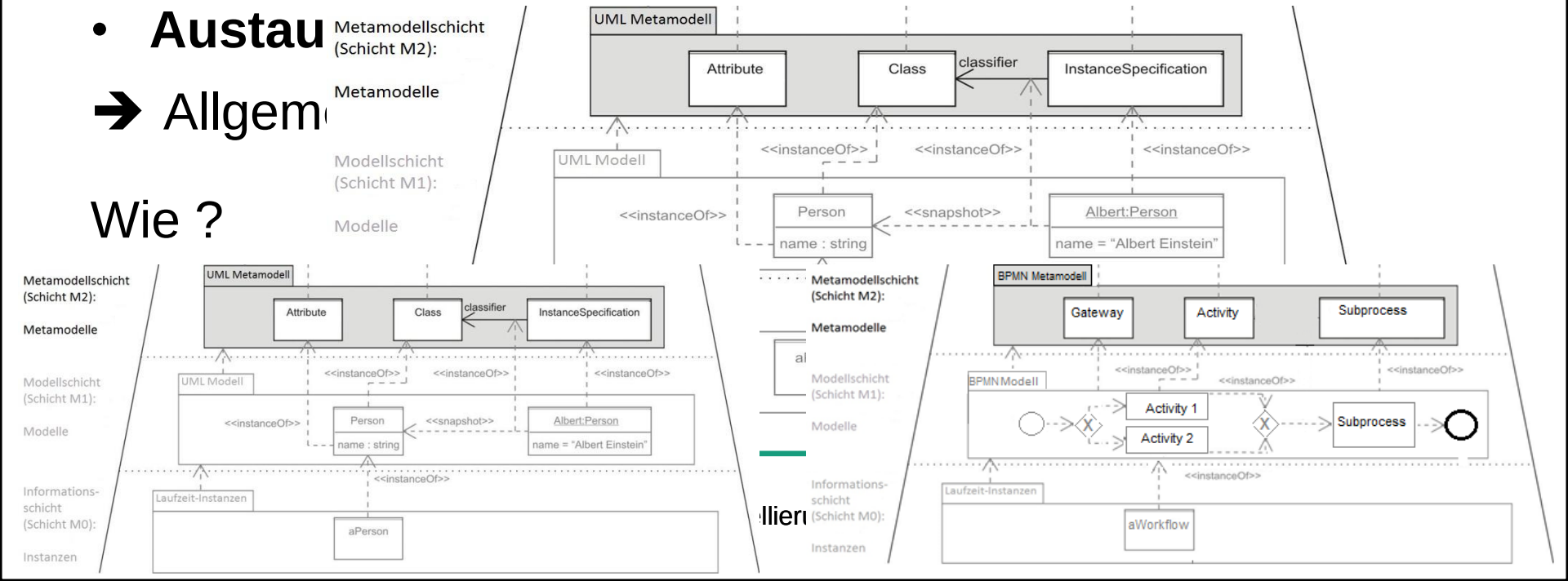
- Weitere Notationen: **eigene Metamodelle** (BPMN; Domain-Specific Languages (DSLs), …).

Gewünscht:

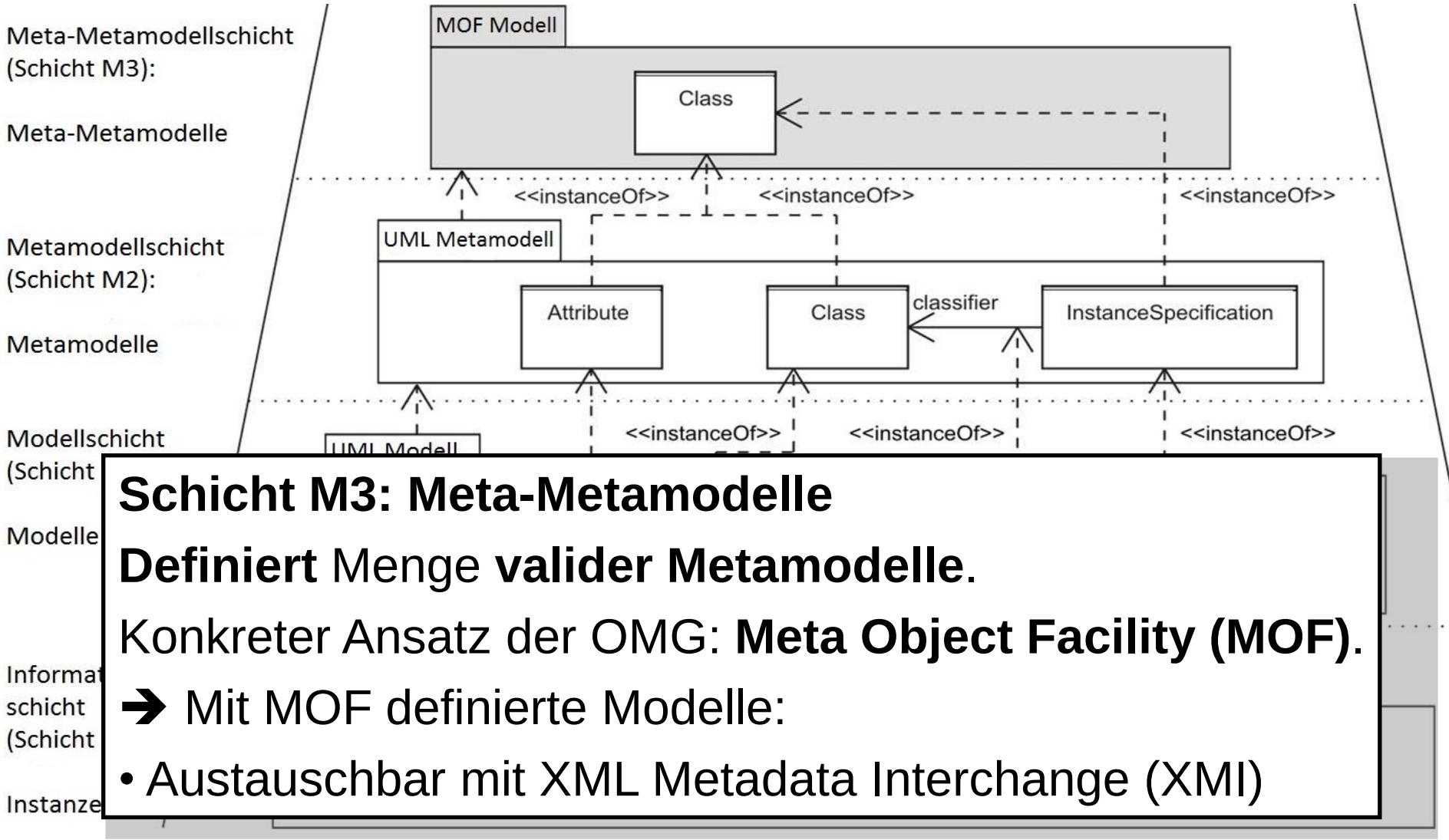
- **Wiederverwendbarkeit** der **Werkzeuge**
- **Austau**

➔ Allgem

Wie ?



Metamodell-Hierarchie: Schicht M3



Schicht M3: Meta-Metamodelle

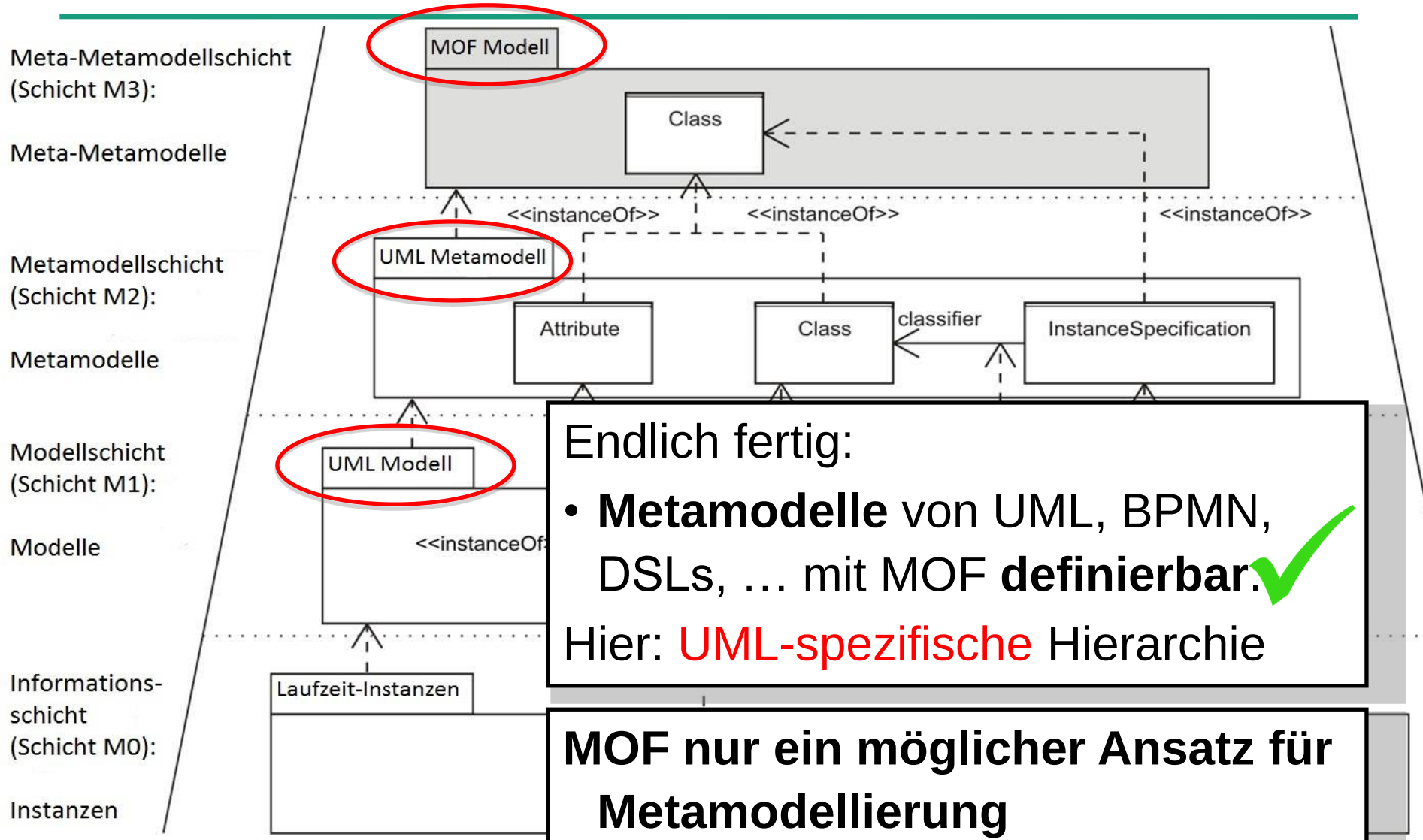
Definiert Menge **valider Metamodelle**.

Konkreter Ansatz der OMG: **Meta Object Facility (MOF)**.

➔ Mit MOF definierte Modelle:

- Austauschbar mit XML Metadata Interchange (XMI)

Metamodell-Hierarchie vs. UML



Endlich fertig:

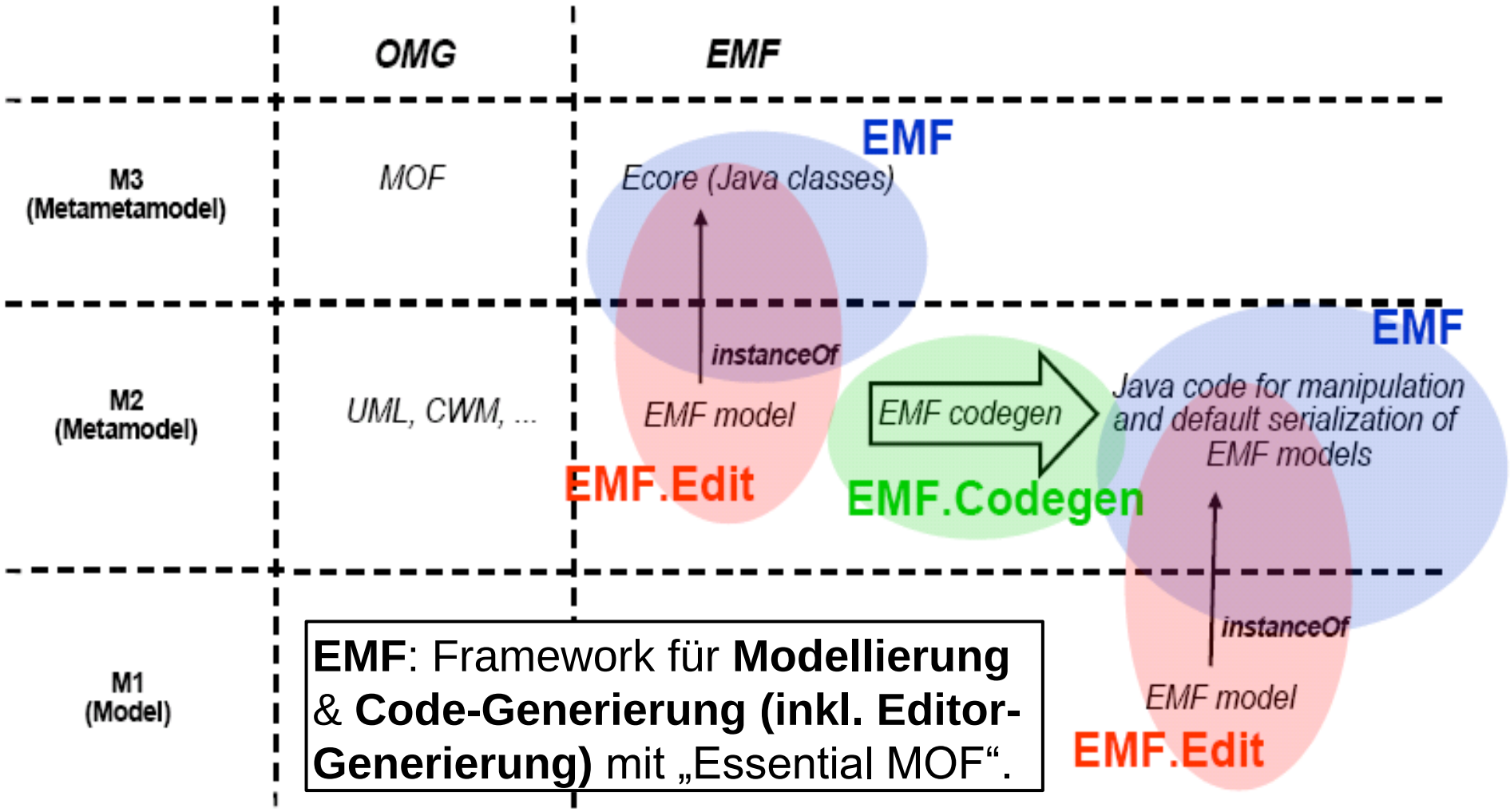
- **Metamodelle** von UML, BPMN, DSLs, … mit MOF **definierbar**. ✓

Hier: UML-spezifische Hierarchie

MOF nur ein möglicher Ansatz für Metamodellierung

- Viele andere Möglichkeiten

Beispiel MOF-Implementierung: Eclipse Modeling Foundation (EMF)



Vorteile von Metamodellierung

Vorteile im **Allgemeinen:**

- **Präzise** Definition von Modellierungsnotation

Vorteile von **MOF:**

- **Wohlverstandene** Notation für Definition der Modellierungsnotation (UML-Klassendiagramme)
- Weitgehende **Werkzeugunterstützung** (Modelltransformation, Codegenerierung etc)

Einschränkungen von Metamodellierung

Einschränkungen im **Allgemeinen**:

- Erstellung / Anpassung Metamodell: Benötigt **Aufwand** und **Expertise**.

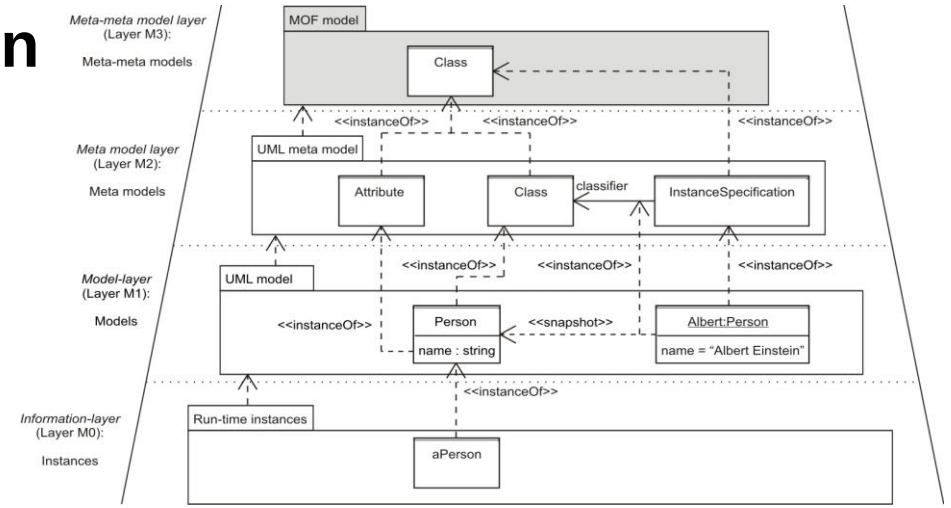
Nachteile von **MOF**:

- Dieselbe Notation (Klassendiagramme) auf verschiedenen Ebenen evt. **verwirrend**.
- **Logisch zirkulär** (definiere Klassendiagramme mit Klassendiagrammen).

Zusammenfassung

Einführung Metamodellierung:

- **Metamodell**: Modell, das Modellierungsnotation definiert.
- **Metamodellierungshierarchie** (M0 bis M3), **MOF**
- Beispiel: Metamodell für **Klassendiagramm**
- **Werkzeugunterstützung** (EMF)
- **Vorteile / Beschränkungen**



Literatur

Bernd Bruegge & Allen H. Dutoit
“Object-Oriented Software Engineering:
Using UML, Patterns, and Java”

MOF Specification

http://www.omg.org/technology/documents/modeling_spec_catalog.htm#MOF

RSS Feed:

feed://www.omg.org/mof/rss/index.xml