10. Classical Metamodelling in the Technical Space MOF/EMOF

Prof. Dr. rer. nat. Uwe Aßmann
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie
Fakultät für Informatik
Technische Universität Dresden
http://st.inf.tu-dresden.de/teaching/most
Version 21-1.3, 20.11.21

1) Metamodelling
   1) Meta-Hierarchy
2) Metametamodels (Metalanguages)
   1) Meta-Object-Facility (MOF)
   2) EMOF

Obligatory Literature

- Kurtev, I., Bezivin, J., Aksit, M.: Technological Spaces: An Initial Appraisal. In: International Symposium on Distributed Objects and Applications, DOA Federated Conferences, Industrial track, Irvine. (2002)
- Model-based Technology Integration with the Technical Space Concept. Jean Bezivin and Ivan Kurtev. Metainformatics Symposium, 2005.
- Jean Bézivin. Model Driven Engineering: An Emerging Technical Space. In R. Lämmel, J. Saraiva, and J. Visser (Eds.): GTTSE 2005, LNCS 4143, pp. 36 – 64, 2006. Springer.
- Ed Seidewitz. What models mean. IEEE Software, 20:26-32, September 2003.
  - http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1231147&tag=1

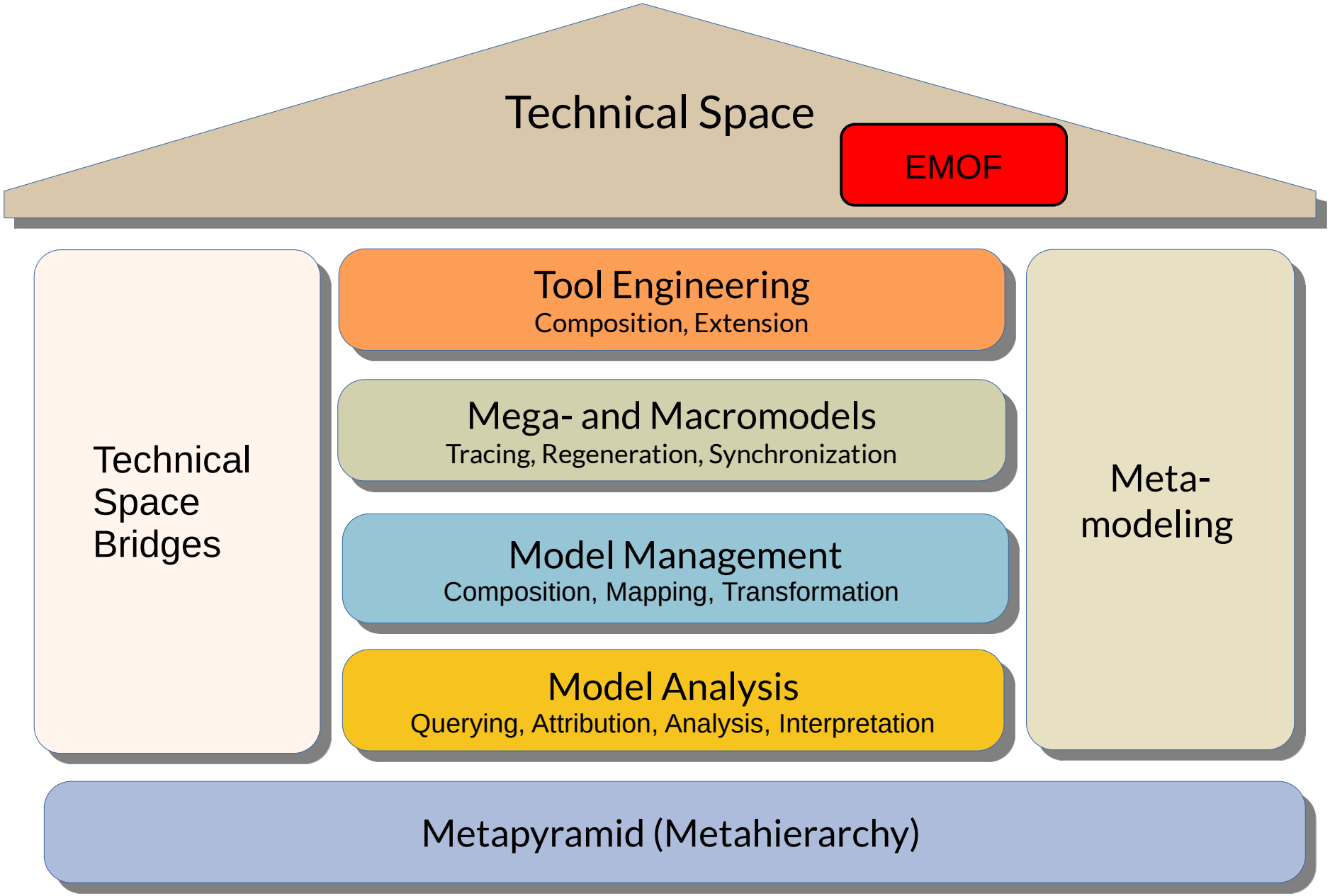
© Prof. U. Aßmann

Other Literature

- Gaševic, Dragan, Djuric, Dragan, Devedžic, Vladan. Model Driven Engineering and Ontology Development, 2nd ed., 2009, ISBN 978-3-642-00281-6
  - http://www.springer.com/computer/swe/book/978-3-642-00281-6?cm_mmc=Google-_-Book%20Search-_-Springer-_-0
- [MOF] Metaobject Facility. OMG. 1.4 and 2.0. www.omg.org
- [Nill] C. Nill. Analysis and Design Modeling Using Metaphorical Modeling Entities. A Modeling Language for the Tools and Materials Approach. Diplomarbeit Technische Universität Dresden, 2006.
- [Atkinson/Kühne] Colin Atkinson and Thomas Kühne. Model-driven development: A metamodeling foundation. IEEE Software, 20(5):36-41, 2003.
- [Favre] Jean-Marie Favre. Foundations of model (driven) (reverse) engineering: Models. Technical report, ADELE Team, Laboratoire LSR-IMAG Université Joseph Fourier, Grenoble, France, 20010. vol. 1-3.
- [Flatscher] Rony Flatscher. Metamodeling in EIA/CDIF - meta-metamodel and metamodels. ACM Trans. Model. Comput. Simul, 12(4):322-342, 2002.
- [Kendall] D. T. Chang and E. Kendall. Metamodels for RDF Schema and OWL. Proceedings of the First International Workshop on the Model-Driven Semantic Web (MDSW 2004), Monterey, USA, September 21, 20010.

© Prof. U. Aßmann

Q10: The House of a Technical Space

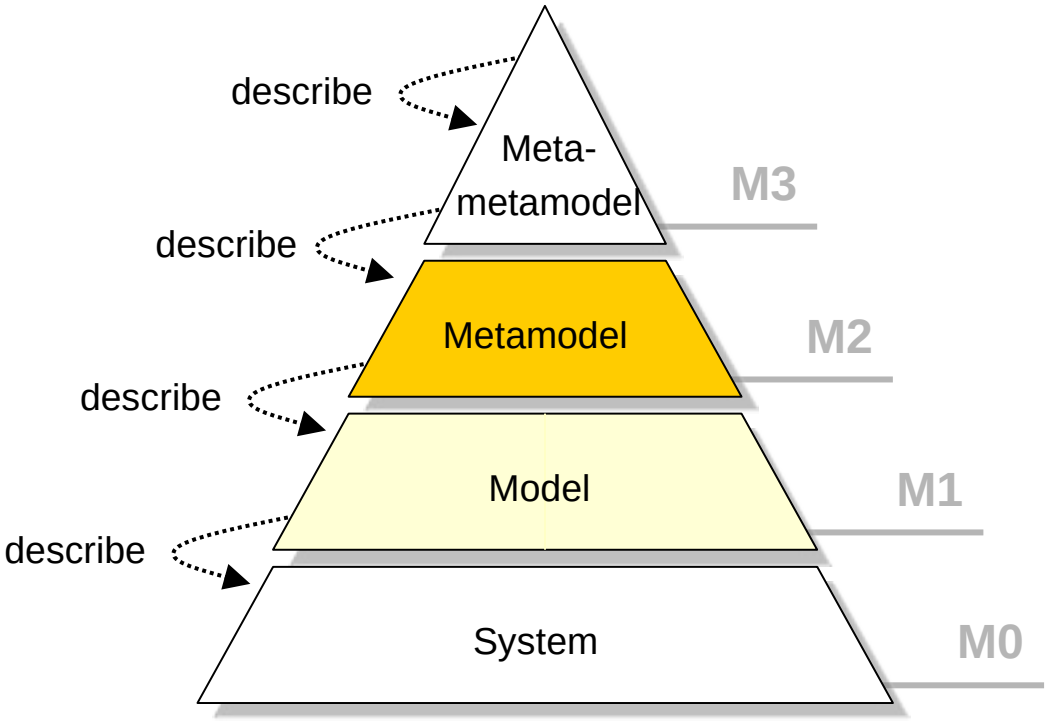


© Prof. U. Aßmann

10.1 Metamodelling in the Classical Metapyramid

The Metamodel Hierarchy (Metapyramid, Metahierarchy)

- Models are widely used in engineering disciplines
  - Need for **tool support** that enables model-editing
- Domain experts want **domain specific languages** (DSL)
  → domain specific models with types from the domain lifted from M1 to M2
- Do not build model editors from scratch each time
  → **reuse** functionality
  → use meta-information

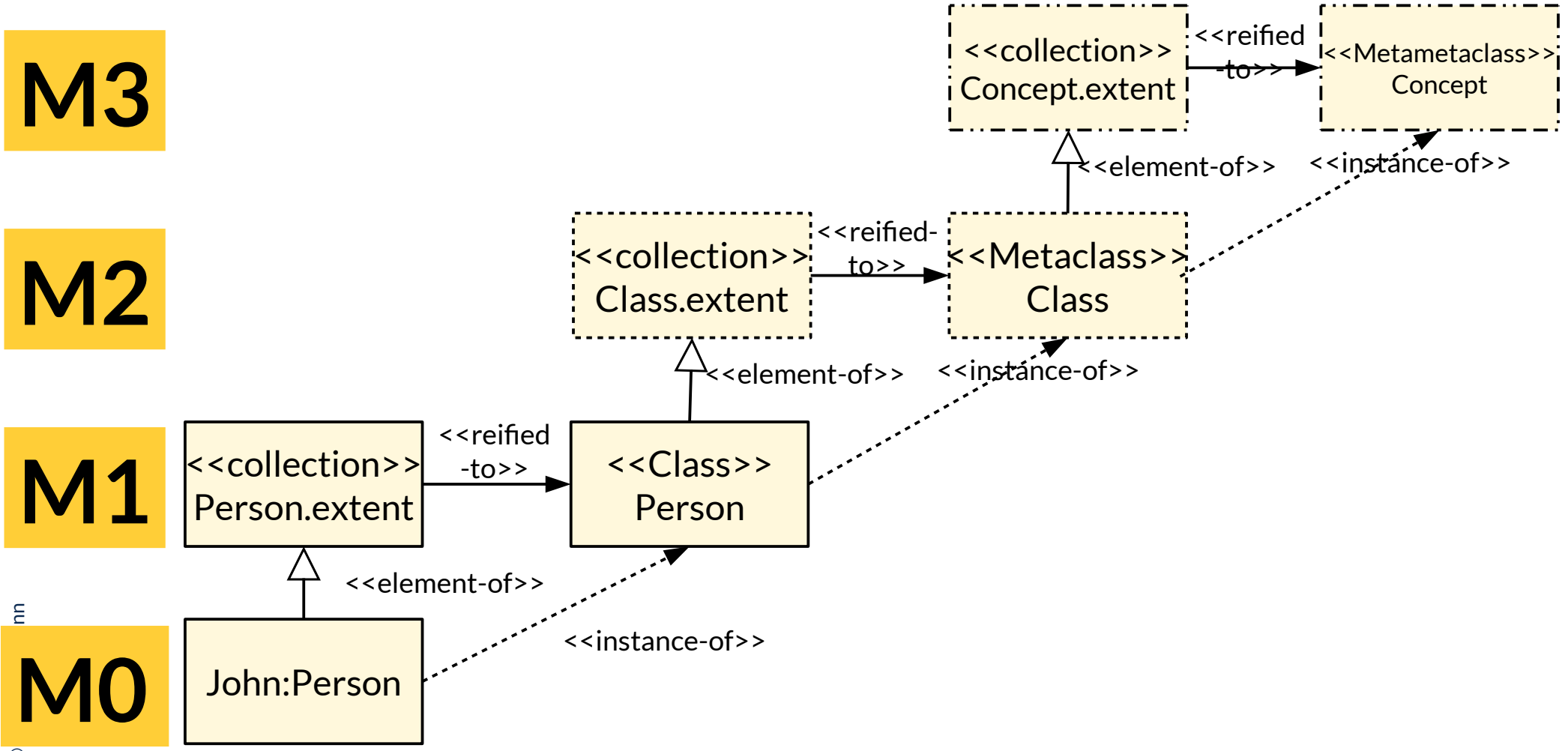


[F. Klar, TU Darmstadt]

© Prof. U. Aßmann

Remember: The Clabject Metahierarchy and Metapyramids

- We call a hierarchy of instance-of relationships a ***metahierarchy*.**
- **A *metapyramid*** is a network of element-of, reified-to, and instance-of relationships

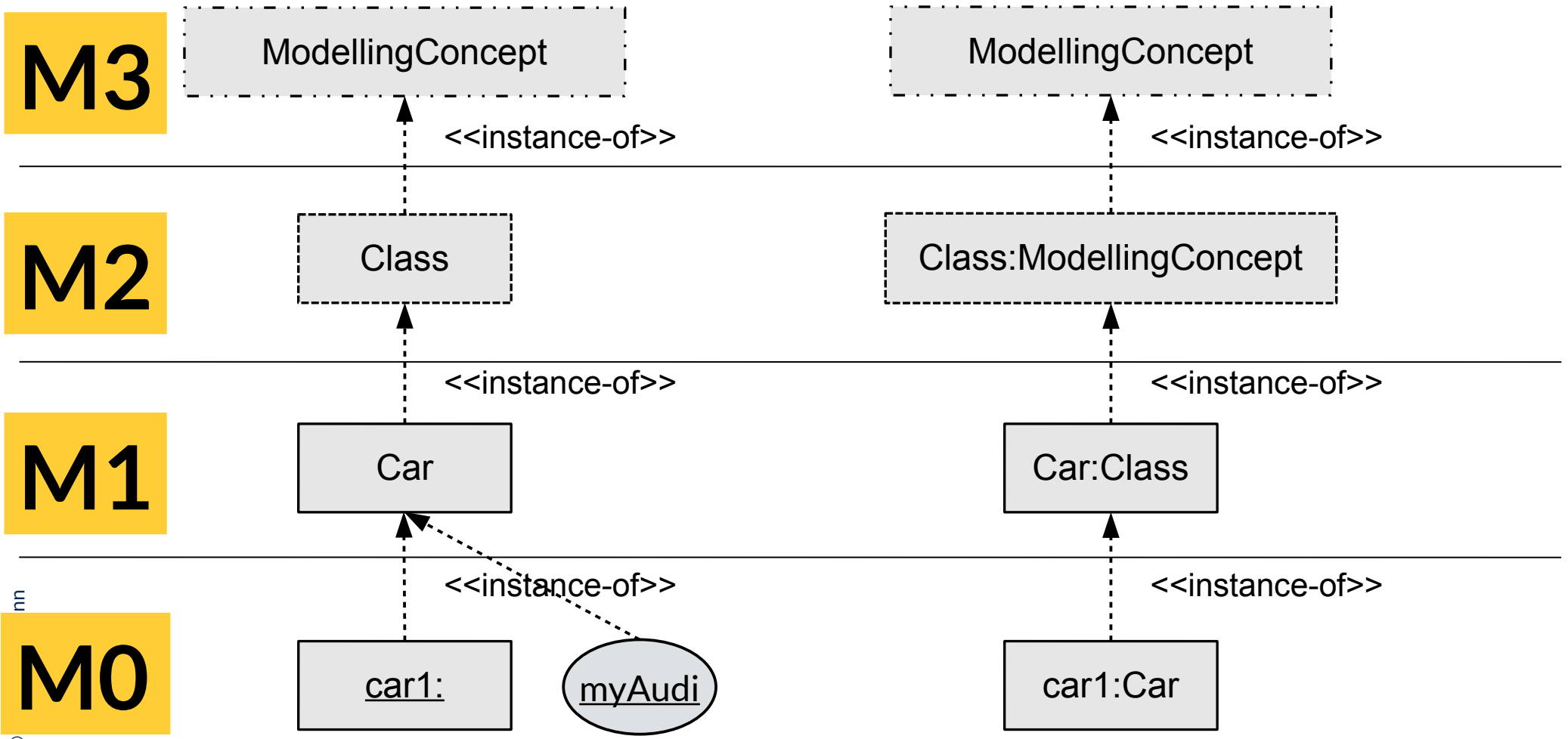


©

Notation

Clabject Hierarchy

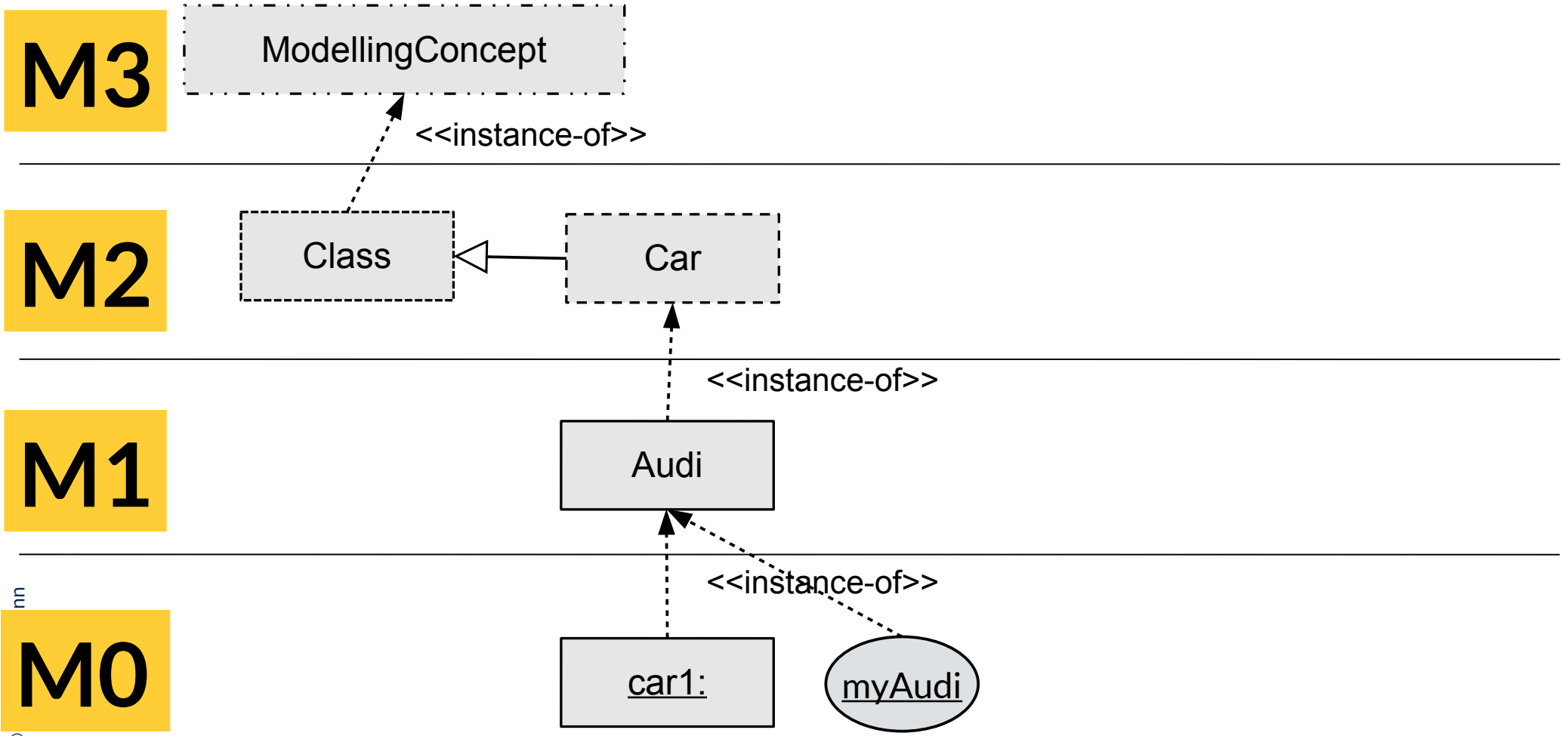
- We write metaclasses (clabjects) with dashed lines, metametaclasses (clabjects) with dotted-dashed lines



©

Lifting a Domain Concept to a Language Concept

- Advantages: support of domain-specific semantics by language semantics
- Which domain semantics has the concept `Car`?



©

Models in Software Engineering

Models define abstractions of realities.

- **Process models (Workflow models)** define workflows and other processes
- **Domain models** describe a domain of the world, or a problem domain from the world of the customer
- **System models** specify systems or artefacts:
  - **Software models** define the structure of code
  - **Architecture models** define computational units, distribution, runtime issues, design patterns or architectural styles
  - **Data models** define die structure of materials and the data (e.g. relational model)

Metamodels define types for model elements.
They define the *structure* of models. Their instances are models.

- **Process metamodels** define concepts for workflows
- **Domain metamodels** define concepts of domains
- **System metamodels** define concepts of systems
- **Programming Language Metamodels** define concepts of programming languages
- **Modeling Language Metamodels** define concepts of modeling languages
- **Domain-specific language (DSL) metamodels** define concepts of DSL
- **Pattern Language Metamodels** define stereotypes for classes
- **Data metamodels** define concepts for materials

© Prof. U. Aßmann

10.2 Metametamodels on M3

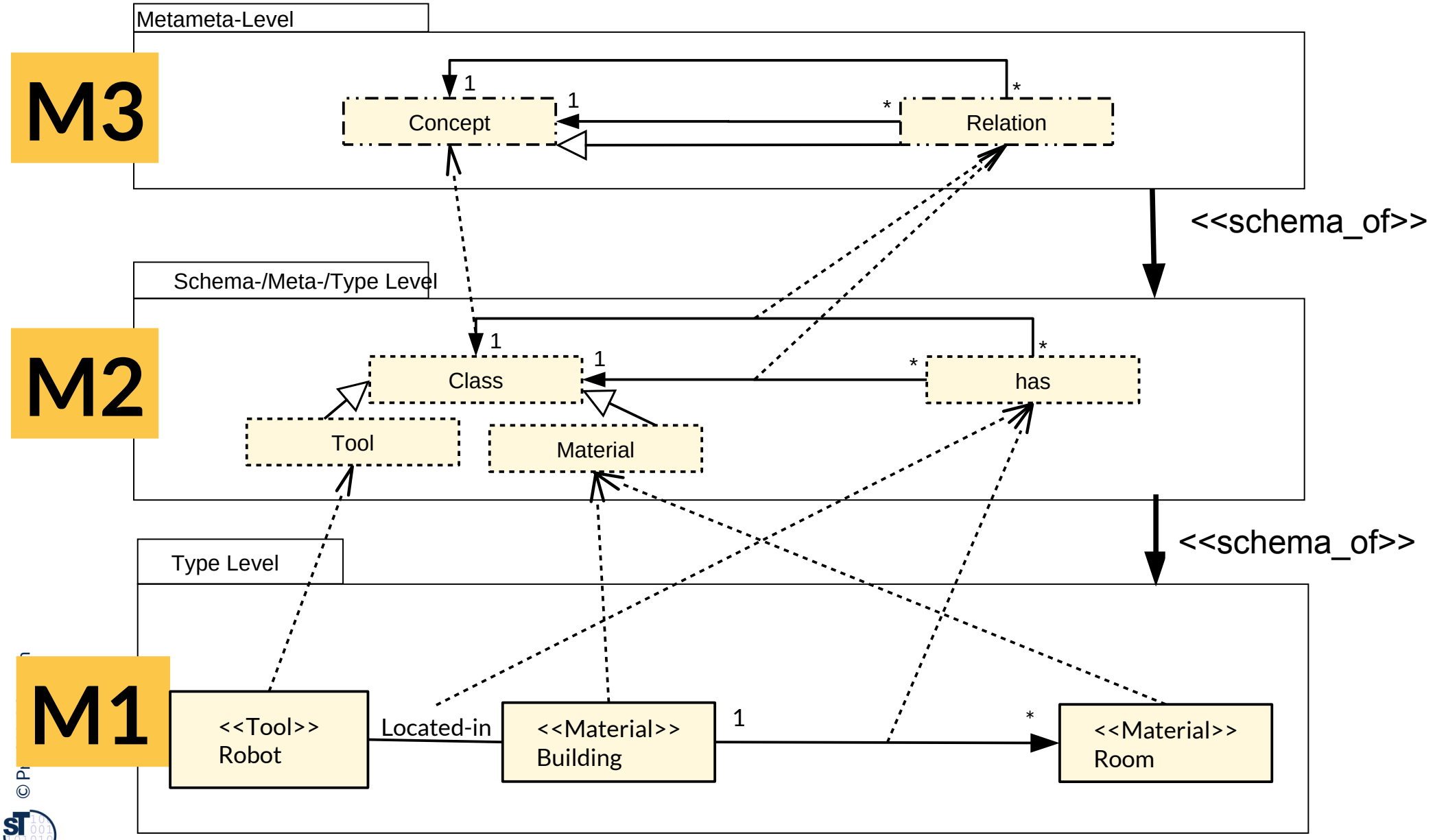
The Metametamodel (Metalanguage)

- Def.: A ***Metametamodel (MMM, Metalanguage)*** is a structural graph schema of a language
  - Defines types for the concepts of a language (the metaclasses on M2)
    - Contains the modeling concepts for languages
  - Structural – no behavior
  - Contains **wellformedness rules** for the graphs on M2
  - Via its **multiplicity constraints**, the metametamodel defines the form of data structure on M0 (sequence, list, table, tree, link tree, reducible graph, graph)
  - Should be minimalistic

Problem: All tools and materials heavily depend on the MMM of the technical space

© Prof. U. Aßmann

Objects, their Clabjects in Models and Metamodels



Tower of Babel Problem

Tragically, no uniform metametamodell has appeared... (tower of babel)

Tools depend on their MMM

[Jan-Pieter Breughel (wikipedia)]

© Prof. U. Aßmann

Metametamodels - Overview

- A ***metametamodel*** describes the context-free and -sensitive structure of a **metalanguage**. It can be augmented with wellformedness rules of the metalanguage.

Examples:

- Meta Object Facility – MOF
  - Complete MOF – CMOF
    - **UML core**
  - Essential MOF – EMOF
    - **Ecore** (Eclipse implementation of EMOF)
- GOPRR – Graph Object Property Role Relation (MetaCase.com)
- CROM of ROSI (DFG training group at TU Dresden)
- GXL – Graph eXchange Language

Problem: All tools and materials heavily depend on the MMM of the technical space

© Prof. U. Aßmann

10.2.1 Ecore and MOF as Simple Metametamodels

Overview of Metalanguage MOF (CMOF: Complete MOF)

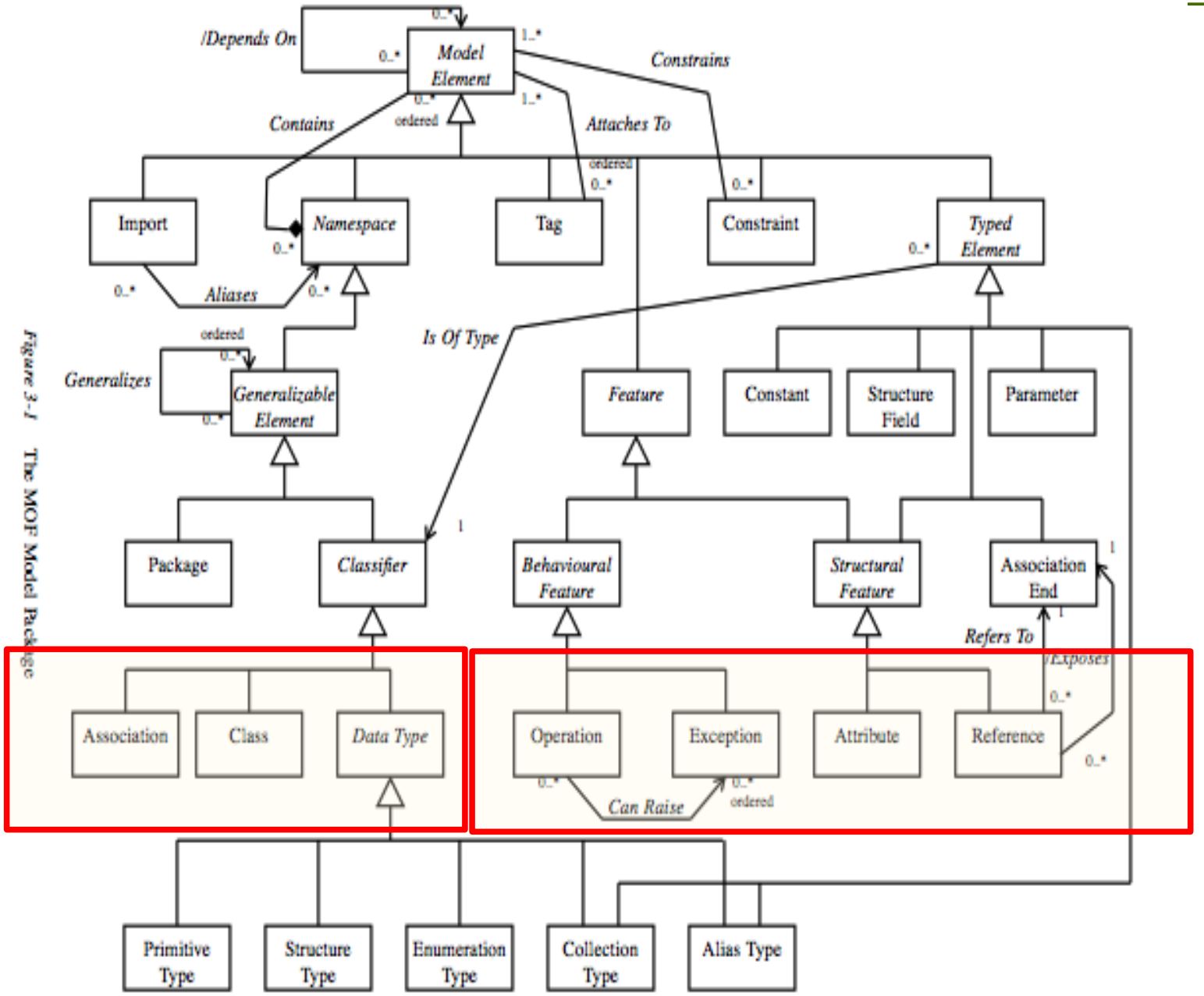


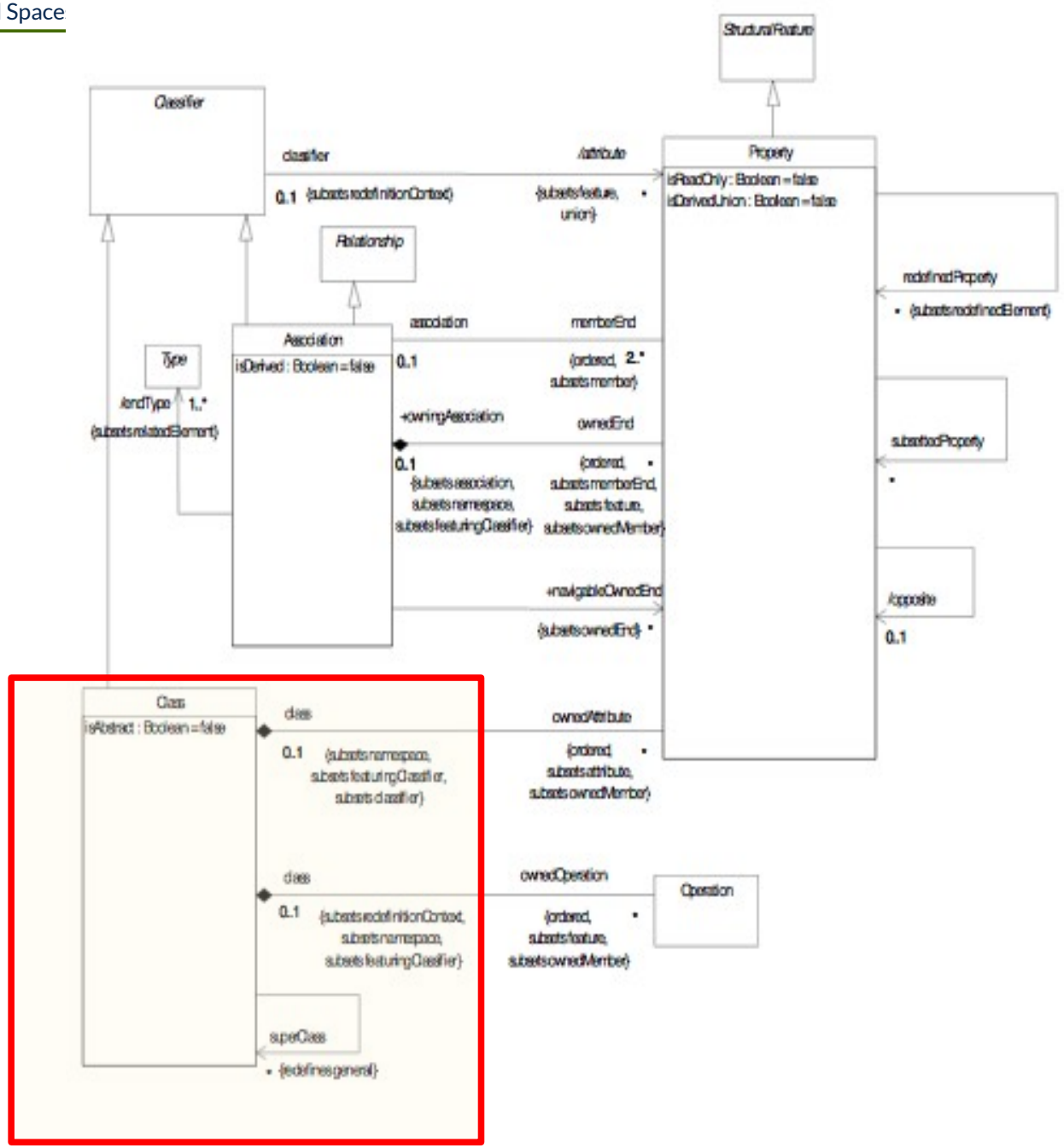
Figure 3-1 The MOF Model Package

[MOF]

© Prof. U. Aßmann

UML Core

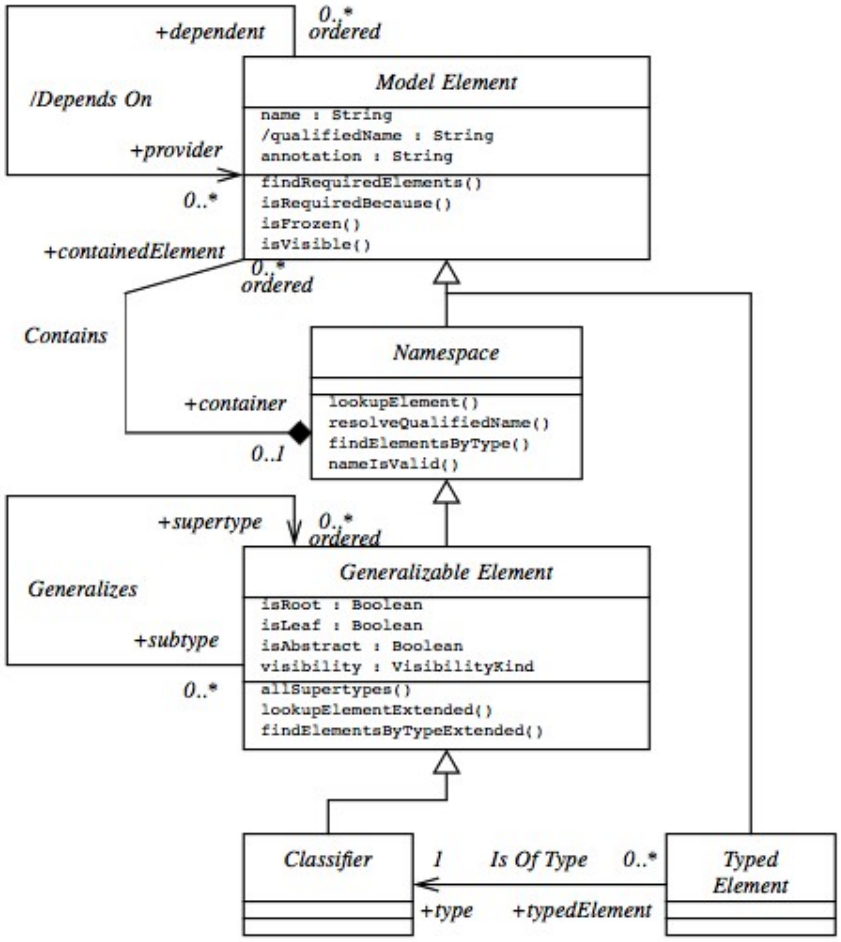
- UML core is subset of MOF, and UML-CD
- It is rather minimalistic



[MOF]

© Prof. U. Aßmann

MOF Central Types

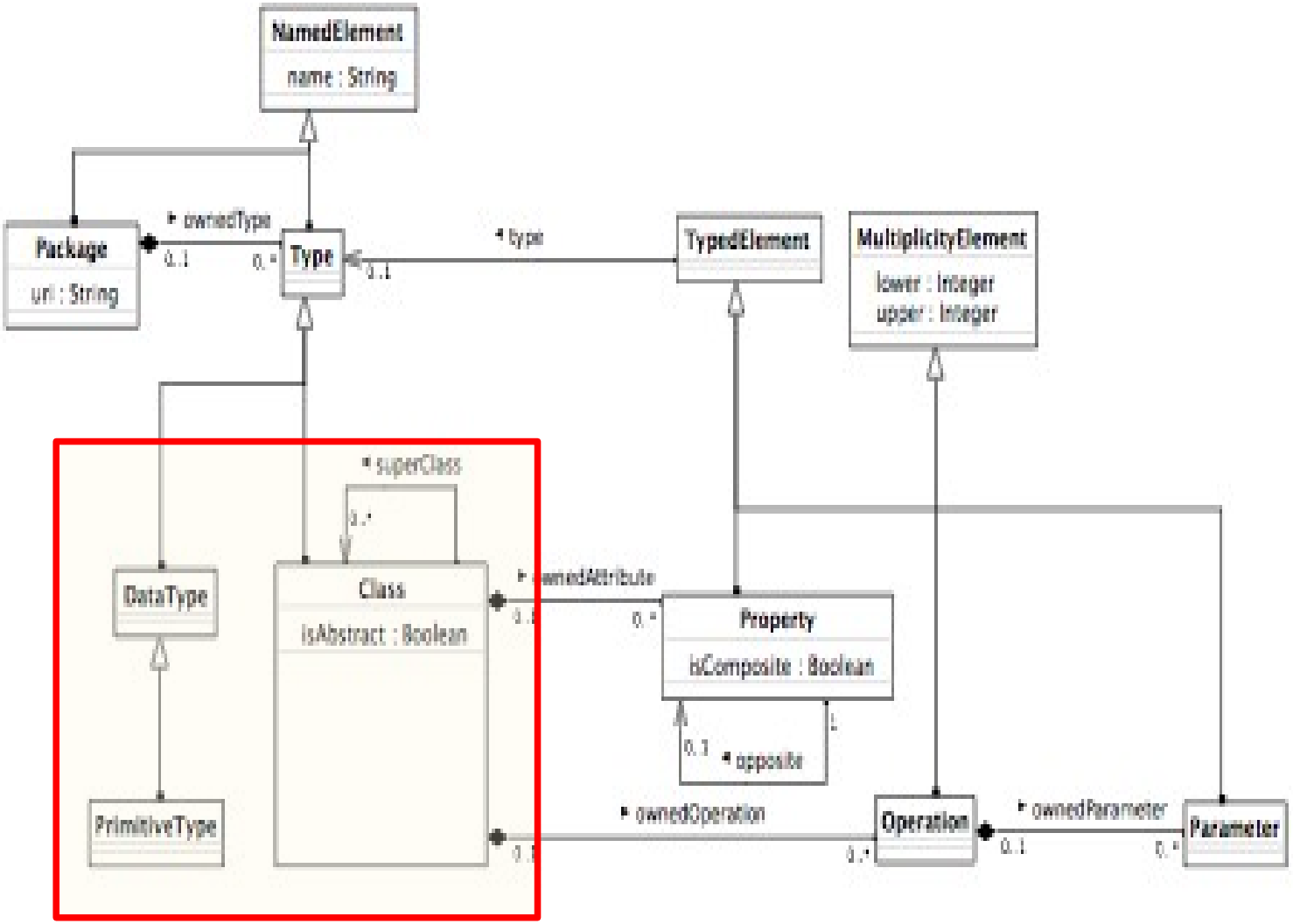


- MOF is for modeling of material, tools, automata (not distinguished)

[MOF]

© Prof. U. Aßmann

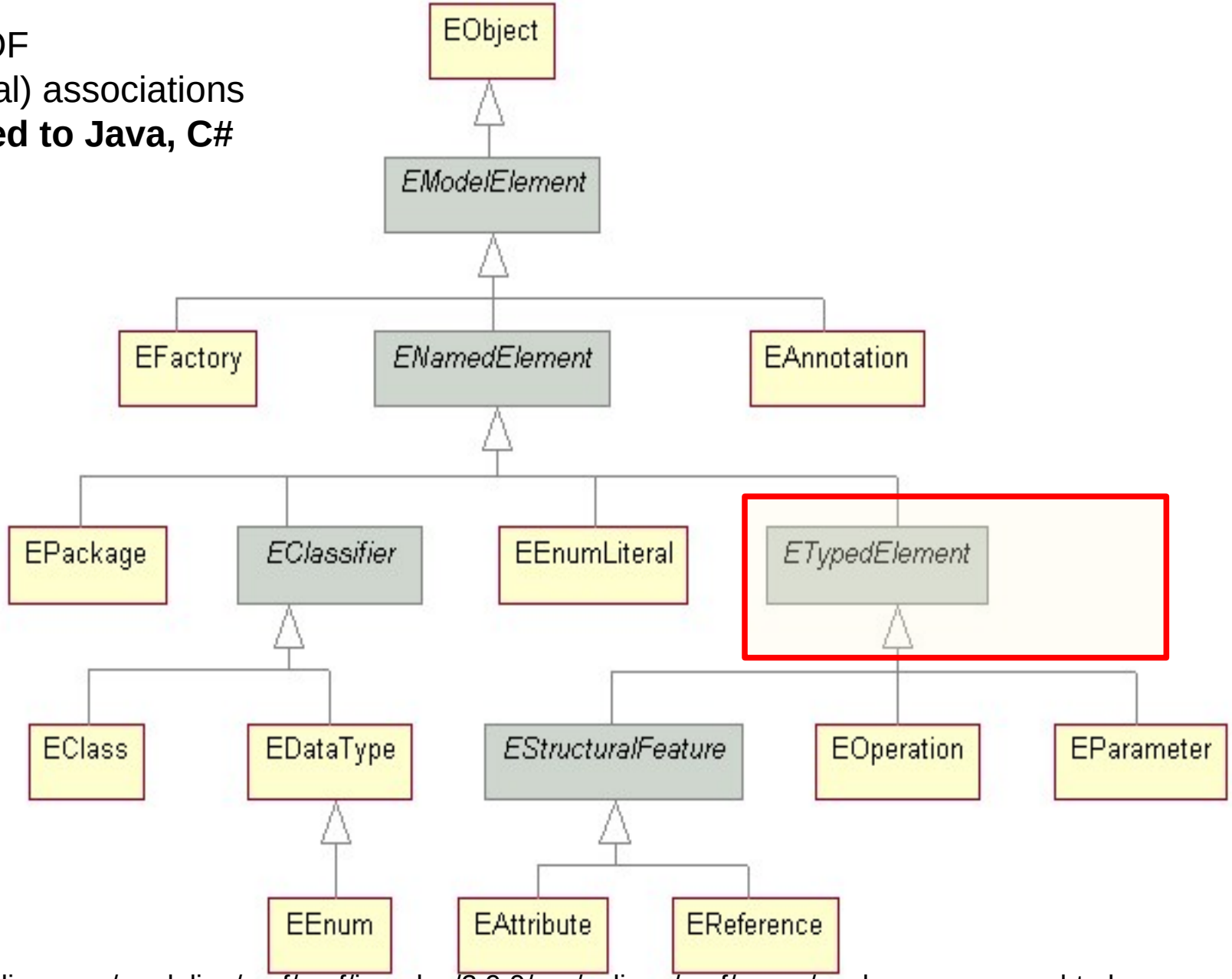
Central MOF Metaclasses with Associations



© Prof. U. Aßmann

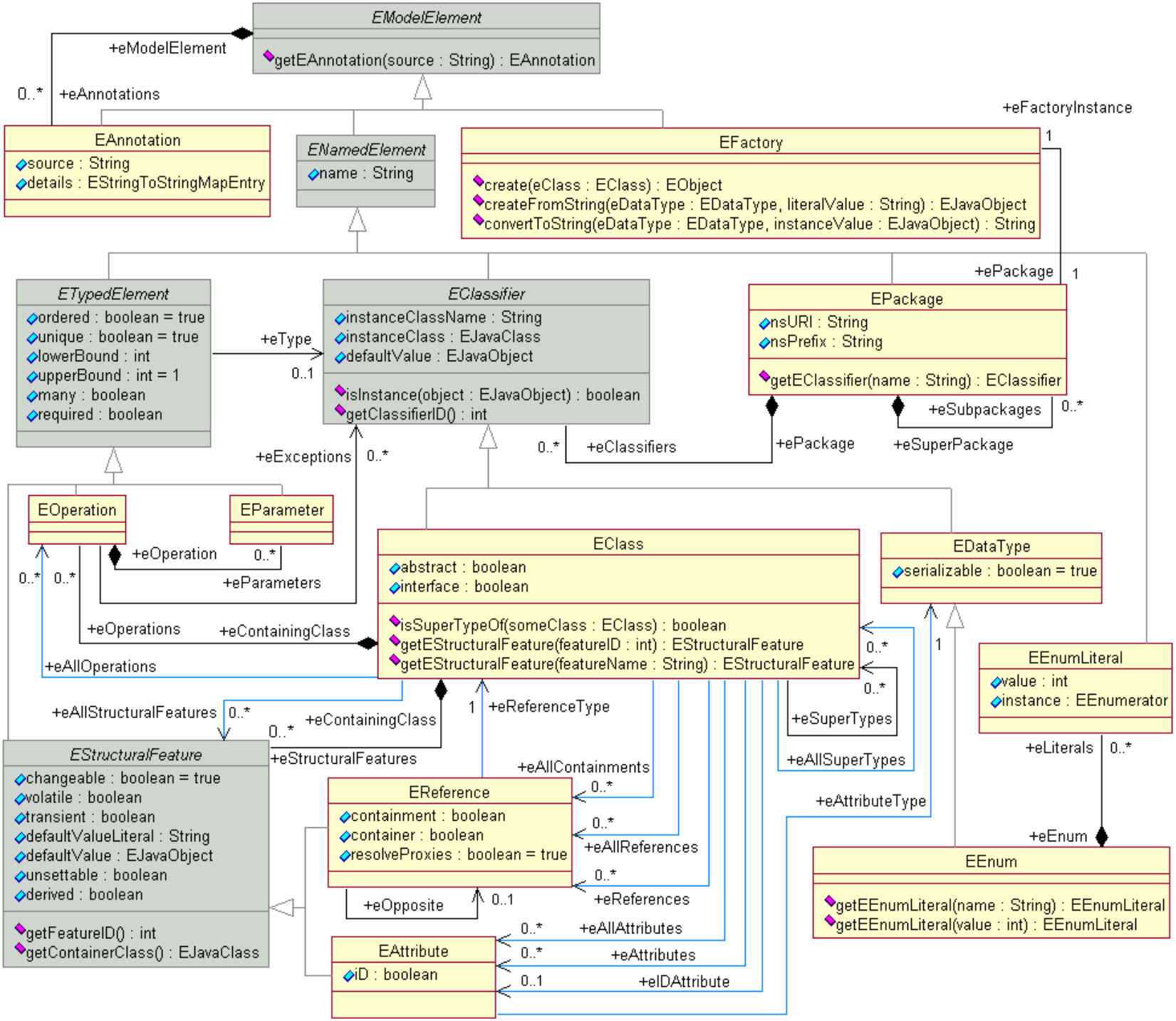
EMOF (Essential MOF)

Subset of CMOF
No (bidirectional) associations
Can be mapped to Java, C#



[MOF] http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html

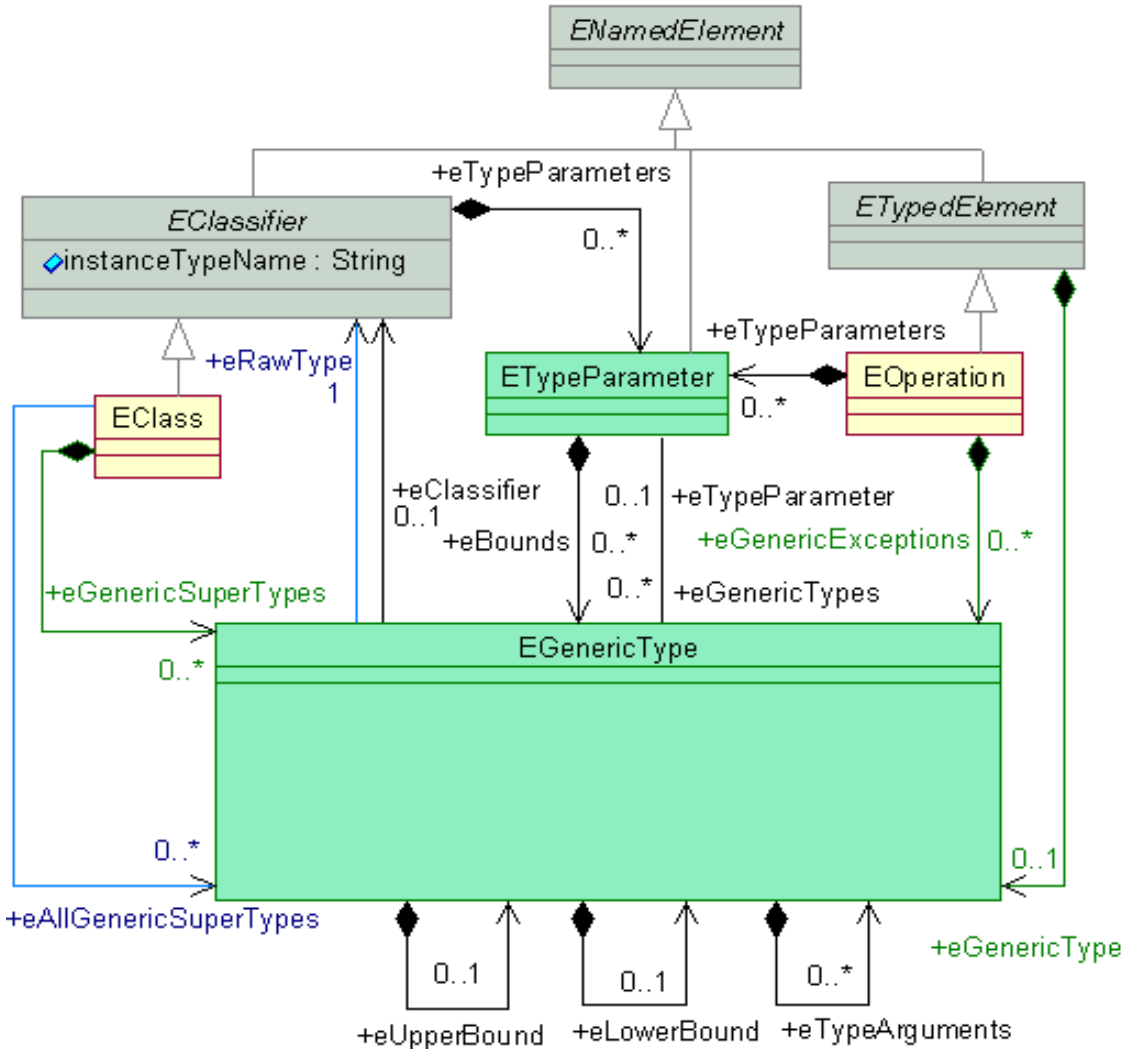
© Prof. U. Aßmann



© Prof. U. Aßmann

http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html

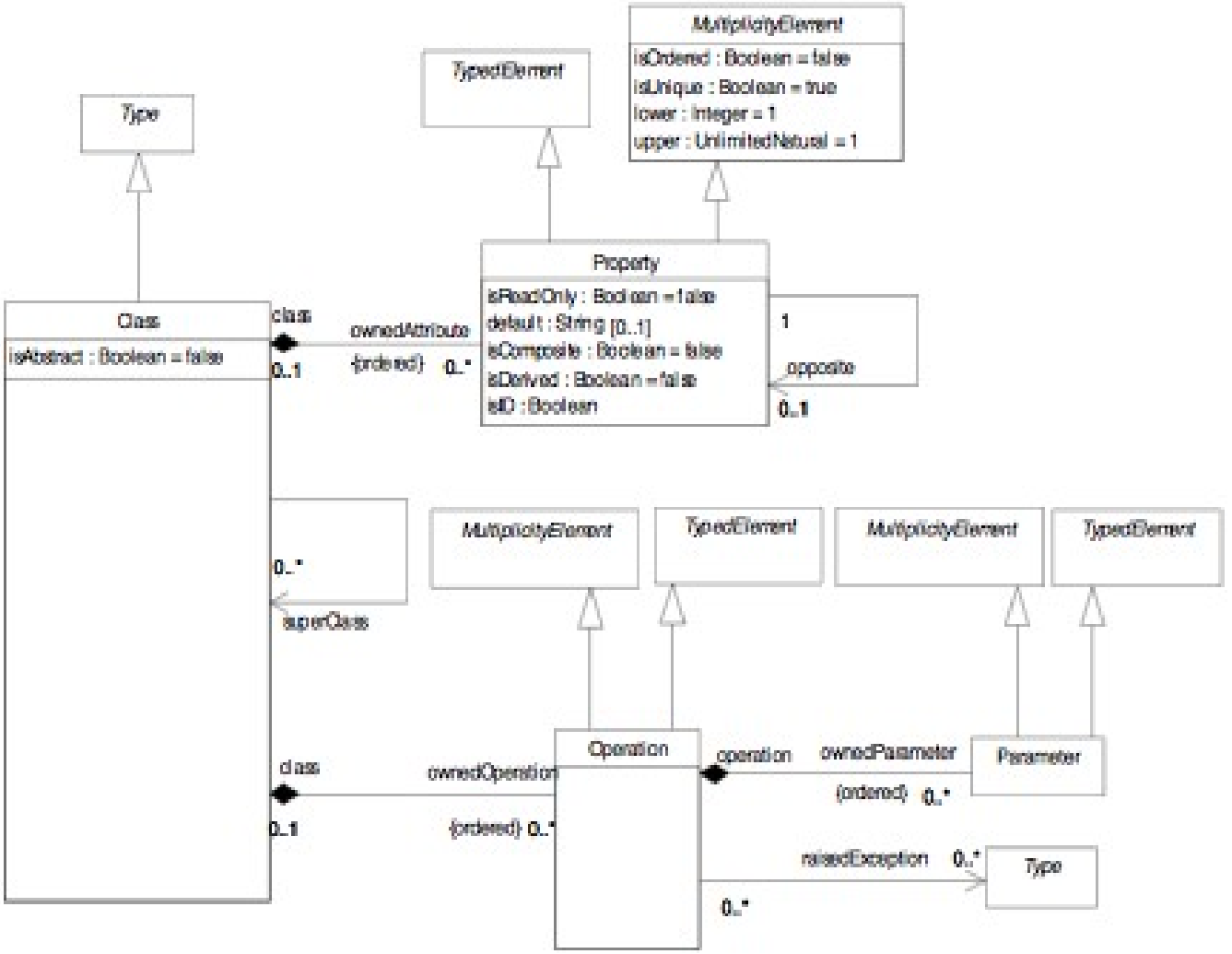
Generic Types



© Prof. U. Aßmann

http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html

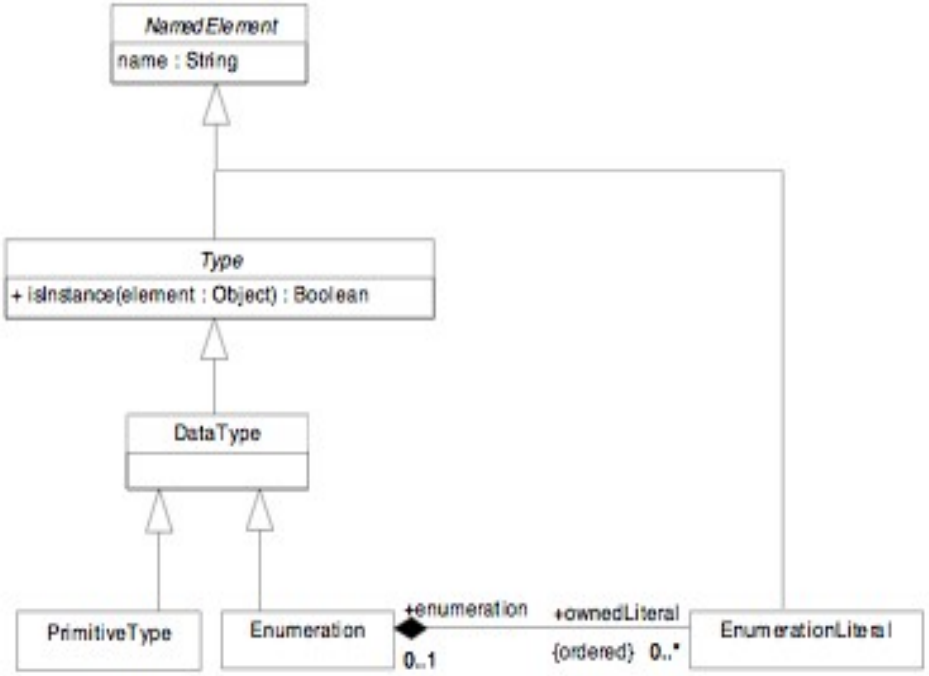
EMOF Classes in Detail

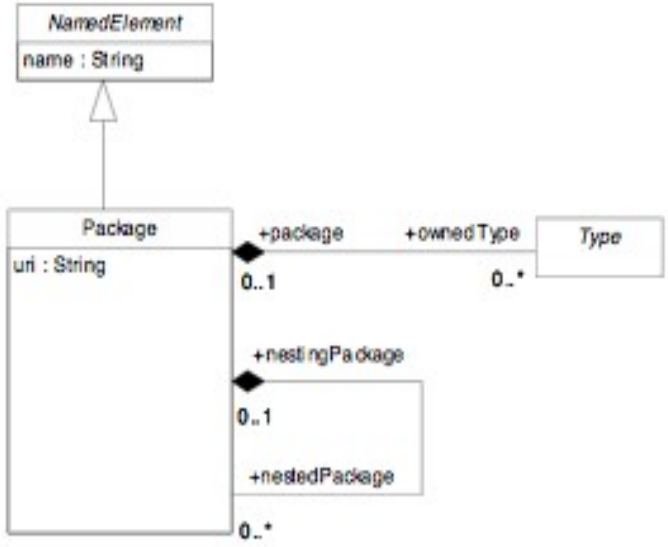


© Prof. U. Aßmann

[MOF]

EMOF Data Types and Packages

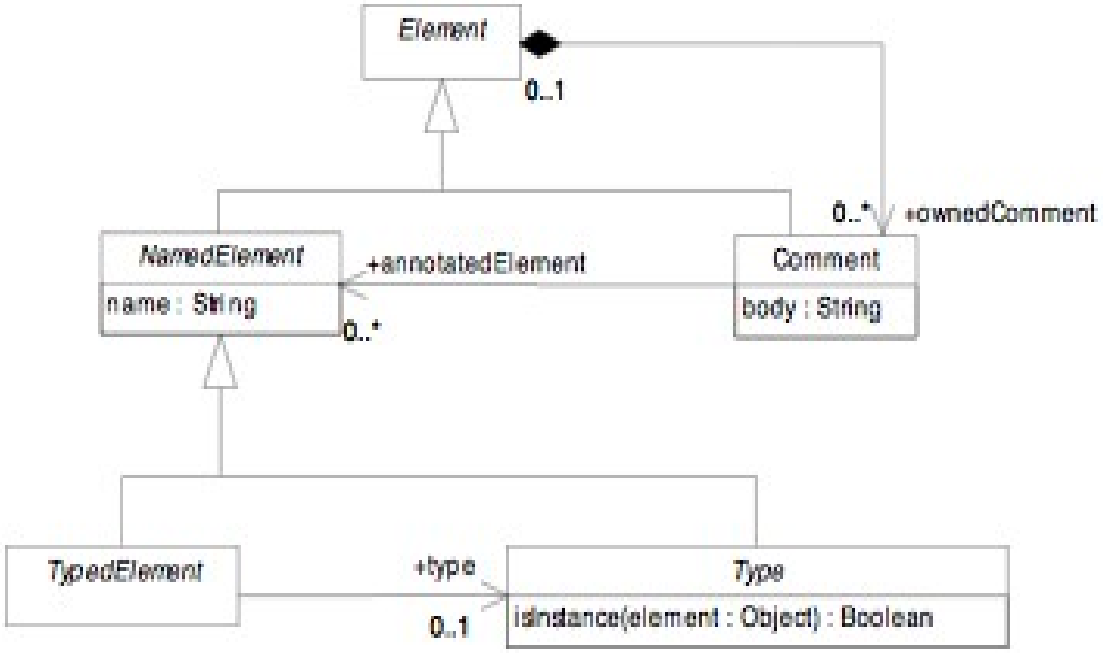




© Prof. U. Aßmann

[MOF]

EMOF Types

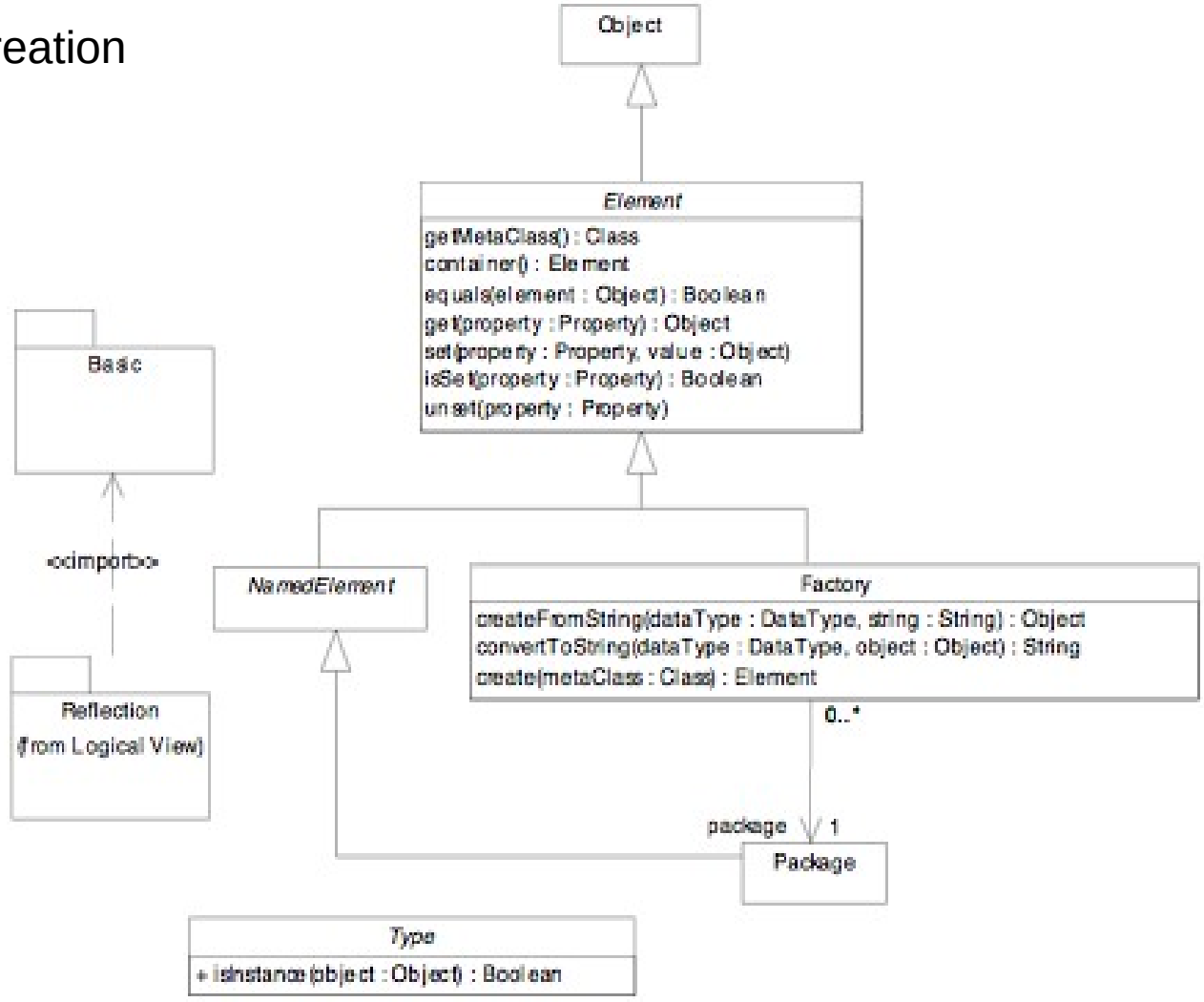


[MOF]

© Prof. U. Aßmann

EMOF Reflection

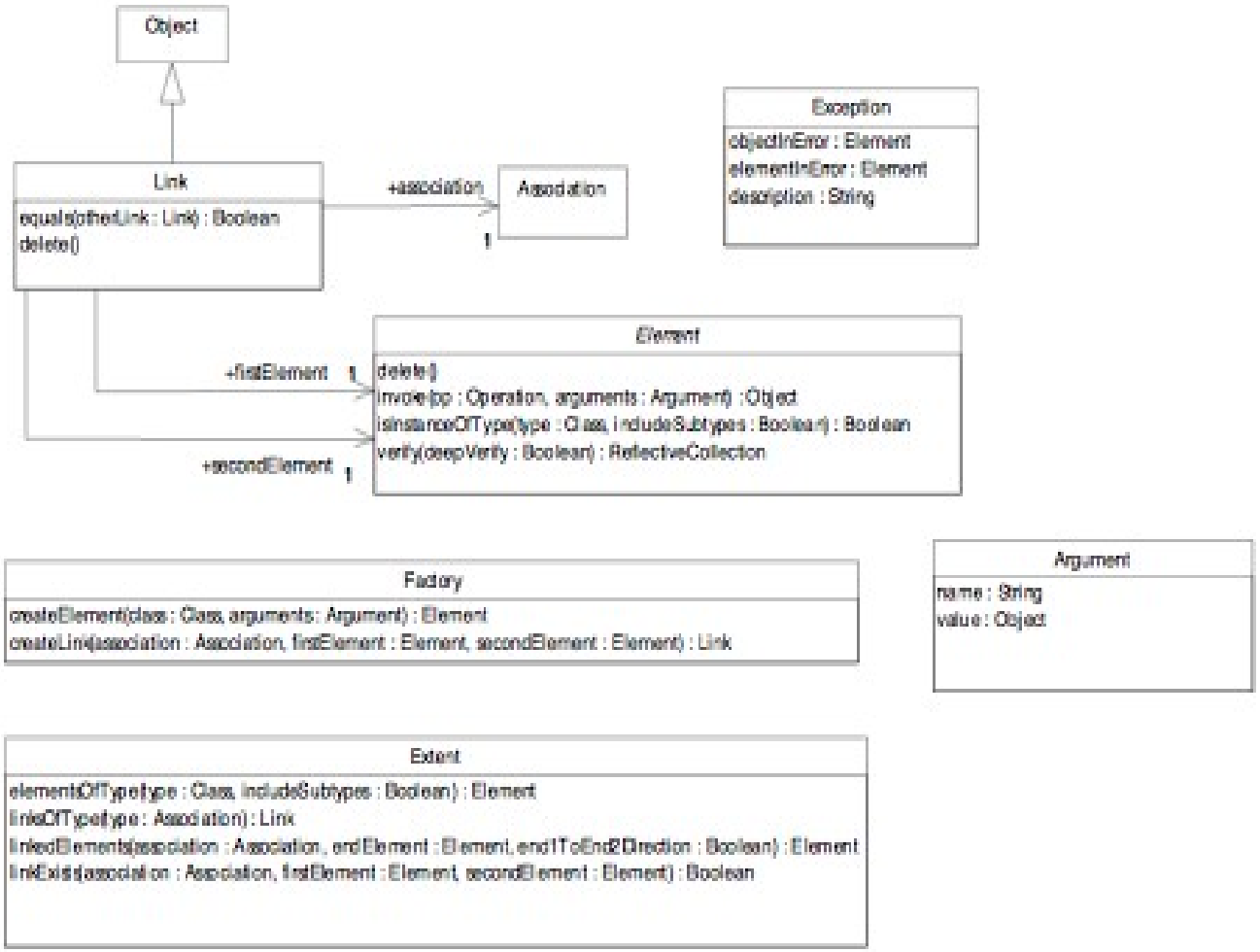
offers access to the metamodel
(getMetaClass())
provides a Factory, for creation
of a Class from String



[MOF]

© Prof. U. Aßmann

CMOF Reflection

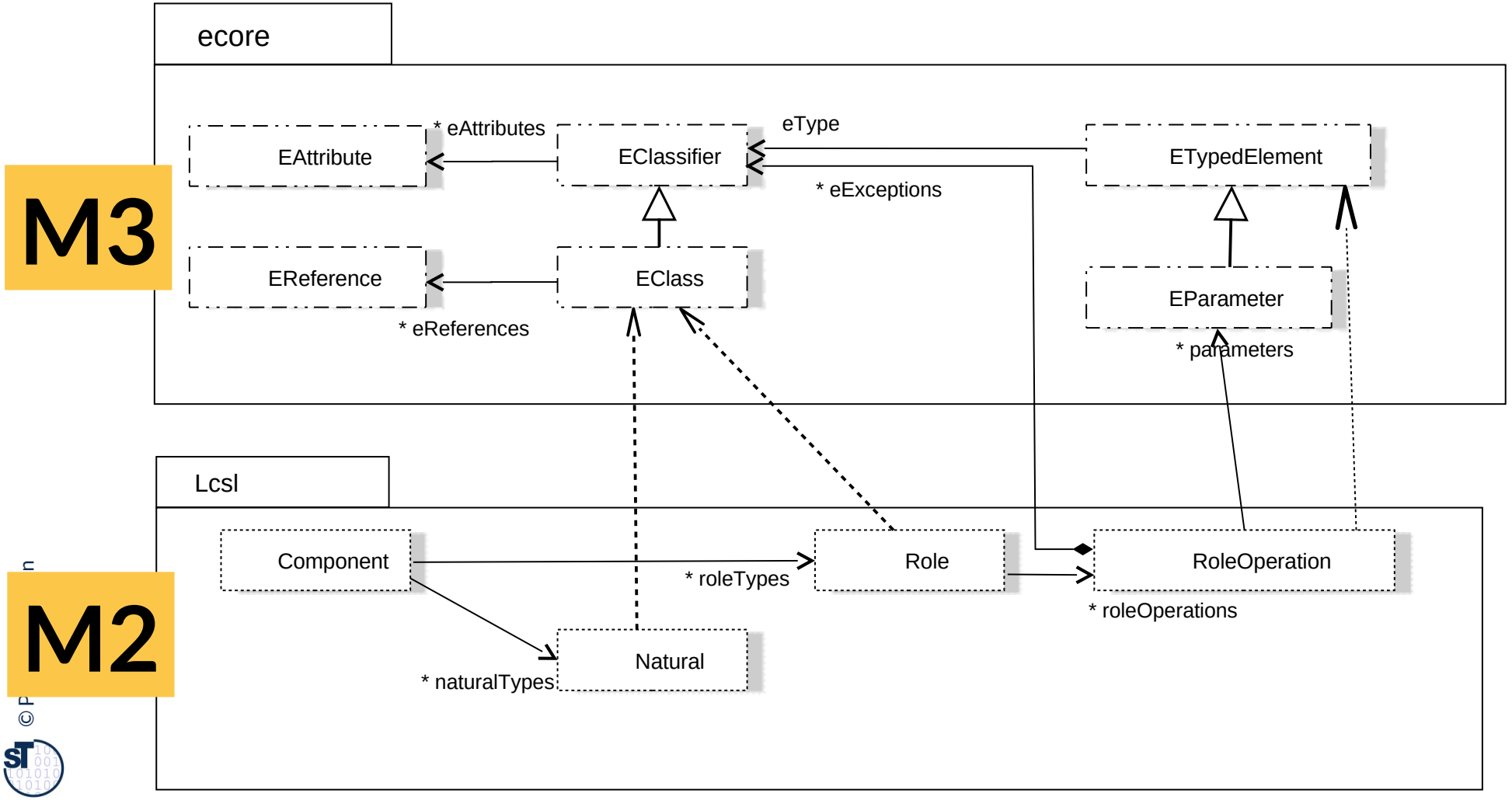


© Prof. U. Aßmann

[MOF]

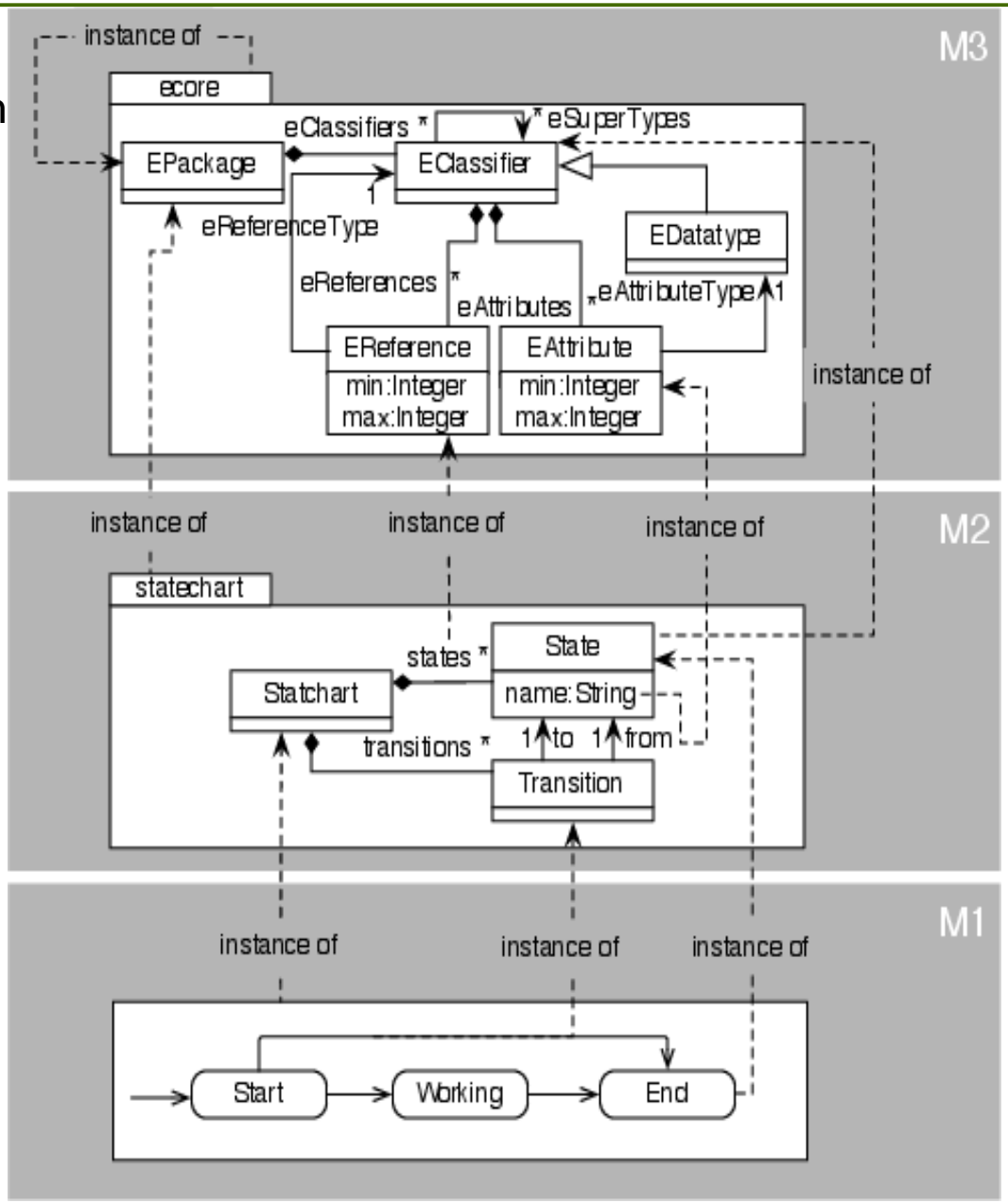
Ex.: Deriving a DSL from EMOF and its Implementation Eclipse ecore

- Ecore is the Eclipse implementation of EMOF
- lcsl is a domain-specific language for component-based modeling [C. Wende]
- Two new Metaclasses Natural and Role derived from EClass



Ex. EMOF/Ecore based Metamodel of Statecharts

- Ecore is the Eclipse implementation of EMOF, provided by the Eclipse Modeling Framework (EMF) on M3
- Here: a metamodel of statecharts (M2), (which is a little DSL)
- a set of states and their transitions (M1)



© Prof. U. Aßmann

10.2.2 Lifting of a Metamodel to a Metametamodel

Lifting of Metamodels

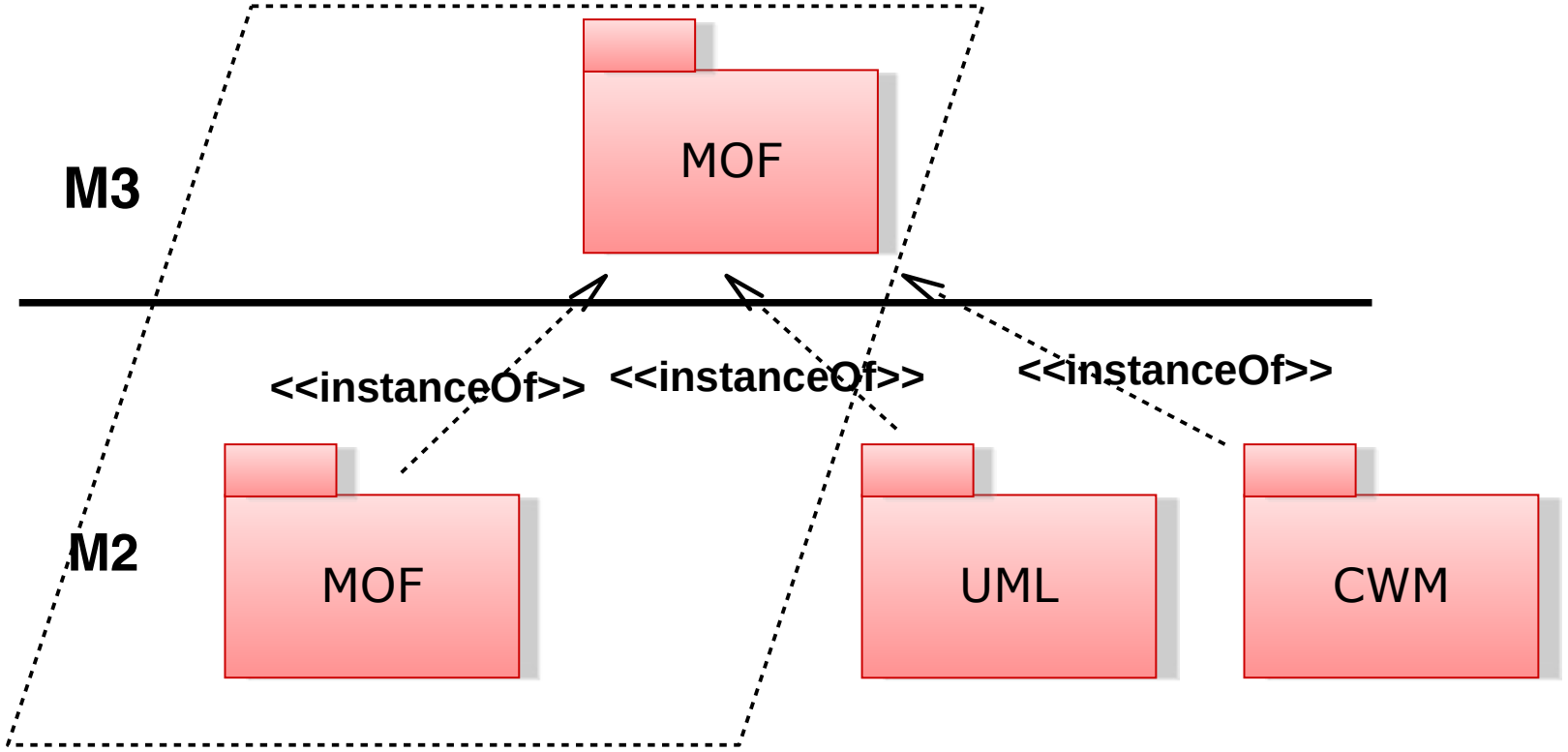
A Metamodel of a data definition language in M2 is being ***lifted (promoted)***, if it is used as metametamodel on M3

- Ex. MOF is a simple DDL (Datendefinitionssprache, structural language) for graphs
  - It can be used on M2 to define new languages with package merge (see UML)
  - It can be used on M3 to define metamodels on M2 as instances
  - MOF is self-descriptive

© Prof. U. Aßmann

Self-Descriptive MOF

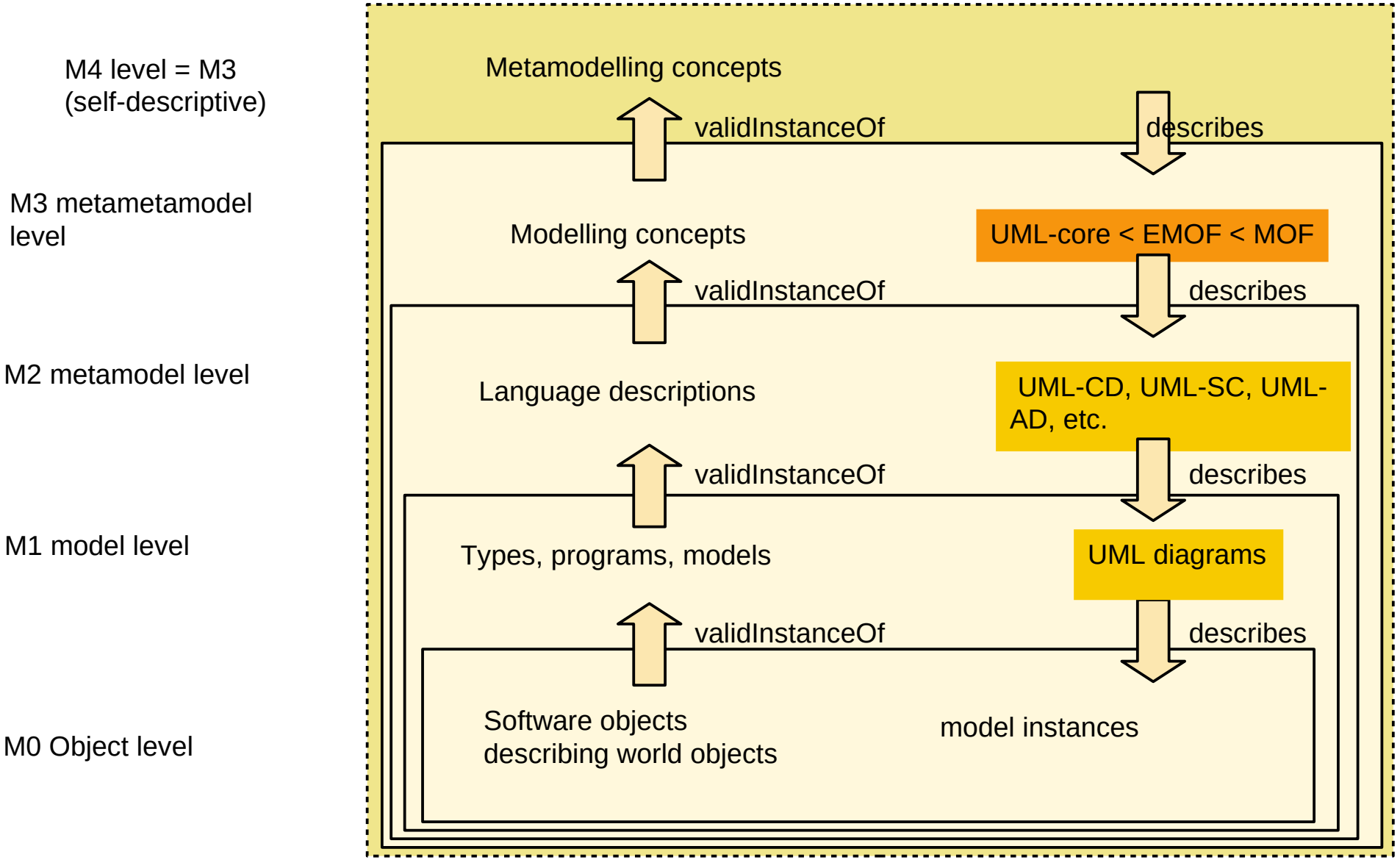
- MOF is ***self-descriptive (selbstbeschreibend)***, because the structure of MOF (M2) is defined in the lifted MOF (M3)
- MOF is *lifted*, because it is used on M2 and M3
- Many other metamodels are also lifted, e.g., EMOF



© Prof. U. Aßmann

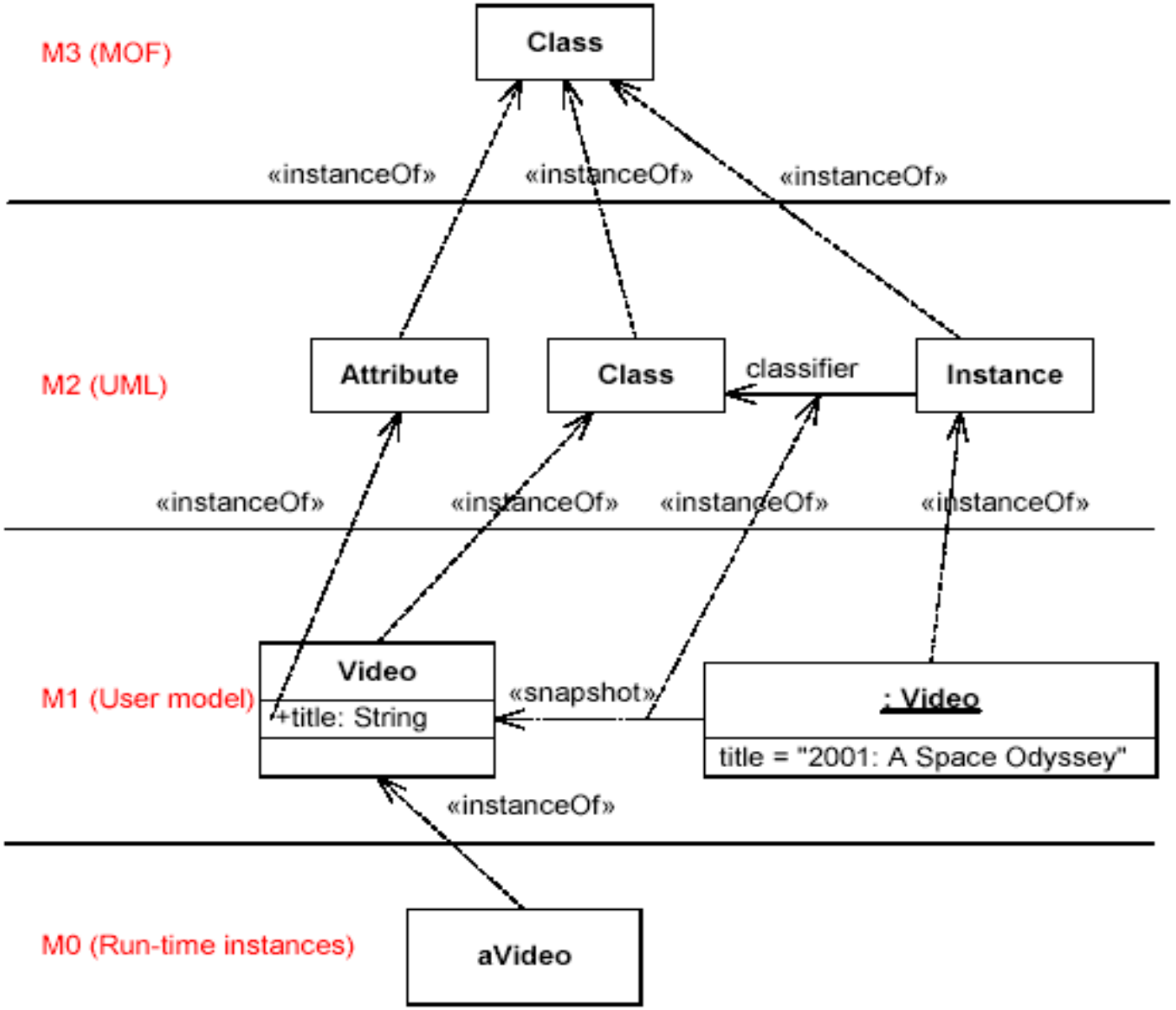
The UML-Core/MOF Metahierarchy

- The UML language manual uses UMLcore, a subset of MOF, as metalanguage



© Prof. U. Aßmann

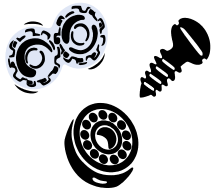
Ex.: MOF-Metahierarchy for UML



From: UML 2.0 Infrastructure Specification; OMG Adopted Specification ptc/03-09-15

© Prof. U. Aßmann

Metamodeling – Benefits



Constraints

- Wellformedness Constraints for detailed definition of language
- Definition of *erroneous states*
- Rules to comply with special *design guidelines*

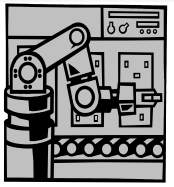
- (Meta-)Modeling of language constructs
- Definiton of language structure
- Domain specific semantics

Metamodel

- *Transformationen* to repair erroneous models
- *Conversion* of incompatible models into design compliant models
- Automatic *adaption* to design guidelines

Abstract Syntax

Transformation



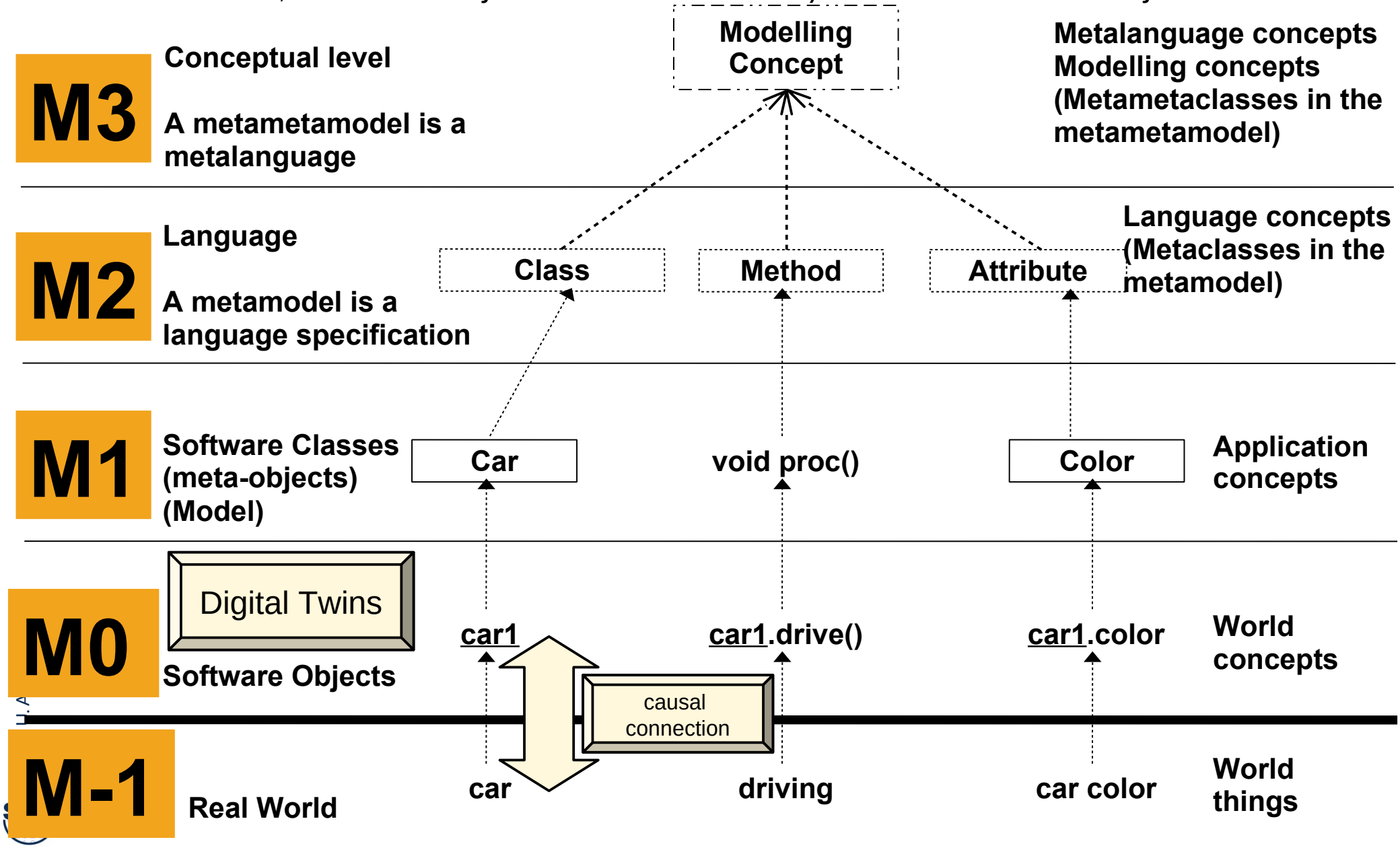
Model

[F. Klar, TU Darmstadt]

© Prof. U. Aßmann

The Metahierarchy for CPS: Objects on M0 are *Digital Twins*

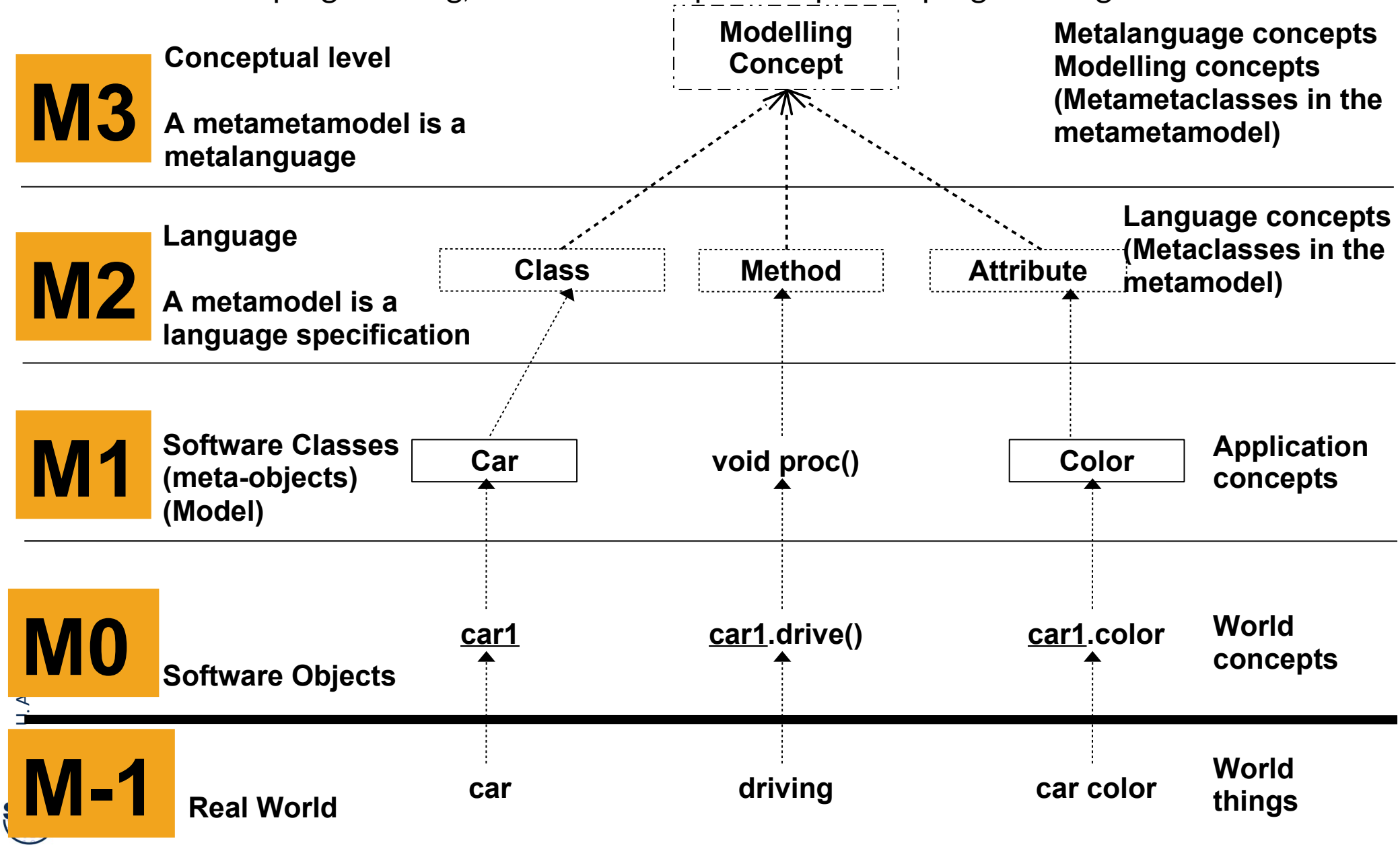
- In a CPS, real-world objects on M-1 are *causally connected* with MO-objects



10.2.3 Metahierarchies for Metaprogramming

Metalevels in Programming Languages (The Metahierarchy for Metaprogramming)

- In Metaprogramming, all meta*-concepts are open for programming

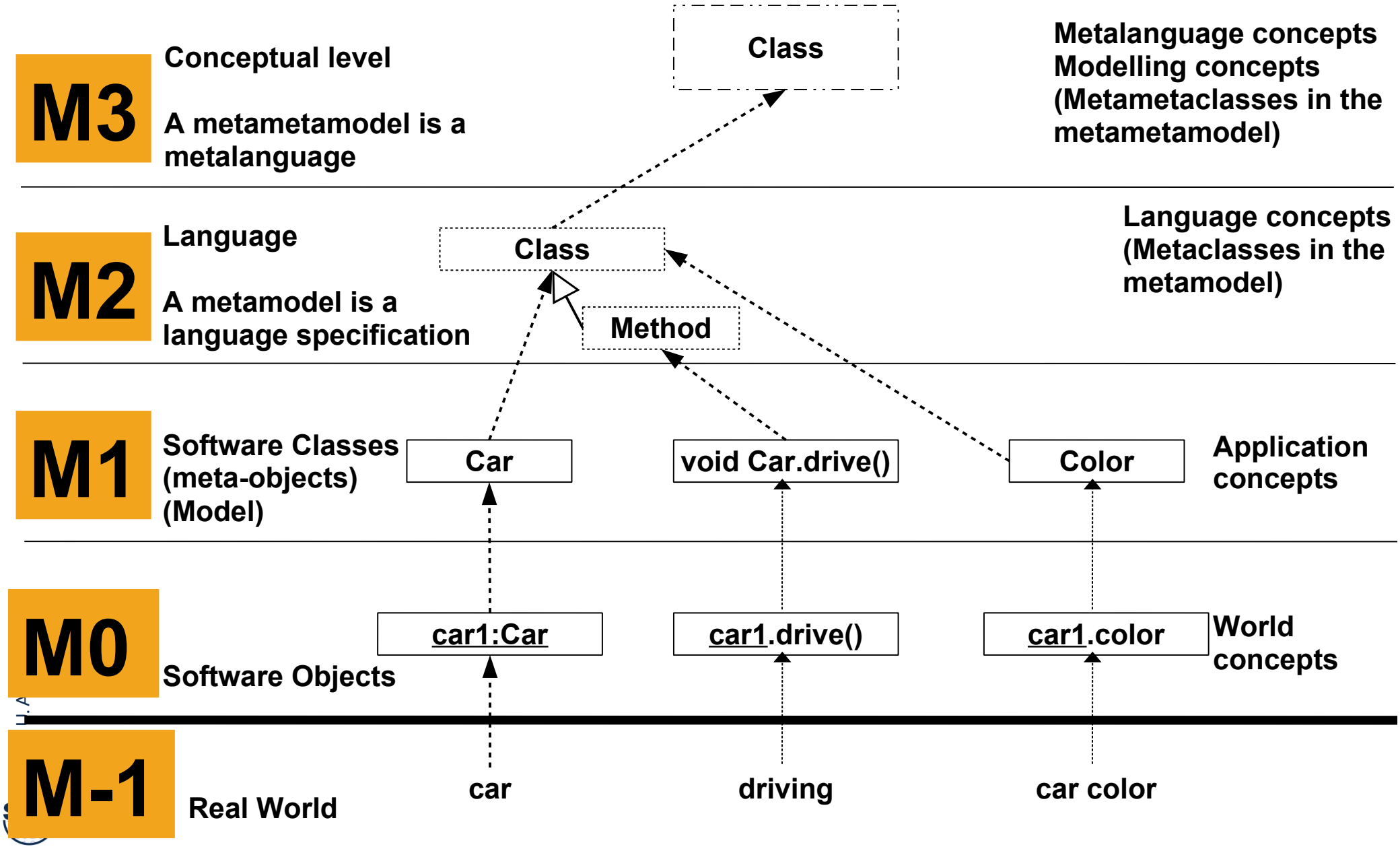


Excursion: Metaprogramming with M2

- **(Dynamic) Metaprograms (reflective programs)** contain code on the basis of the metamodel of their own language (self model)
  - They permanently run in the application and regenerate its parts
  - Hard to statically analyse on termination and other features
  - Reflection is slow
- **Metaprogram-Procedures** (Semantic Macros, Hygenic Macros, Programmable Macros [Weise/Crew], Orchestration Style Sheets) can be typed by a metamodel
  - Parameter types and return types of prodedures are metaclasses
  - → See course CBSE
- **Introspective Programs** inspect the metamodels or metadata of other programs / components and adapt to them (-> CBSE)

© Prof. U. Aßmann

Metalevels in Smalltalk, a Dynamic Metaprogramming Language



Static Metaprograms

- **Codegenerators** are metaprograms producing *new* code or models by introspection
- **Static Metaprograms** run in the compiler and code-generate a program

© Prof. U. Aßmann

The End

- Why is lifting an application concept to M2 advantageous?
- Compare MOF and EMOF. Why do many programmers like EMOF more than MOF?
- Explain the advantages that MOF sopports general associations.
- Why is MOF semantically more rich than EMOF and UMLcore?
- What is the purpose of a metametamodel?
- Would it make sense to use Tools-and-Materials Pattern Language (TAM) on the M3 level, i.e., in the metametamodel?
- Explain why TAM stereotypes do not occur on M2.

© Prof. U. Aßmann

Different Types of Semantics and their Metalanguages (Description Languages)

- **Structure**
  - Described by a context-free grammar or a metamodel
  - Does not regard context
- **Static Semantics** (context conditions on structure), Wellformedness
  - Described by context-sensitive grammar (attribute grammar, denotational semantics, logic constraints), or a metamodel with context constraints
  - Describes context constraints, context conditions, meaning of names
  - Can describe consistency conditions on the specifications
    - "If I use a variable here, it must be defined elsewhere"
    - "If I use a component here, it must be alive"
- **Dynamic Semantics** (Behavior)
  - Interpreter in an interpreter language (e.g., lambda calculus), or a metaobject protocol
  - A dynamic semantics consists of sets of run-time states or run-time terms
  - In an object-oriented language, the dynamic semantics can be specified in the language itself. Then it is called a meta-object protocol (MOP).

© Prof. U. Aßmann