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

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

Version 19-0.3, 03.10.19

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

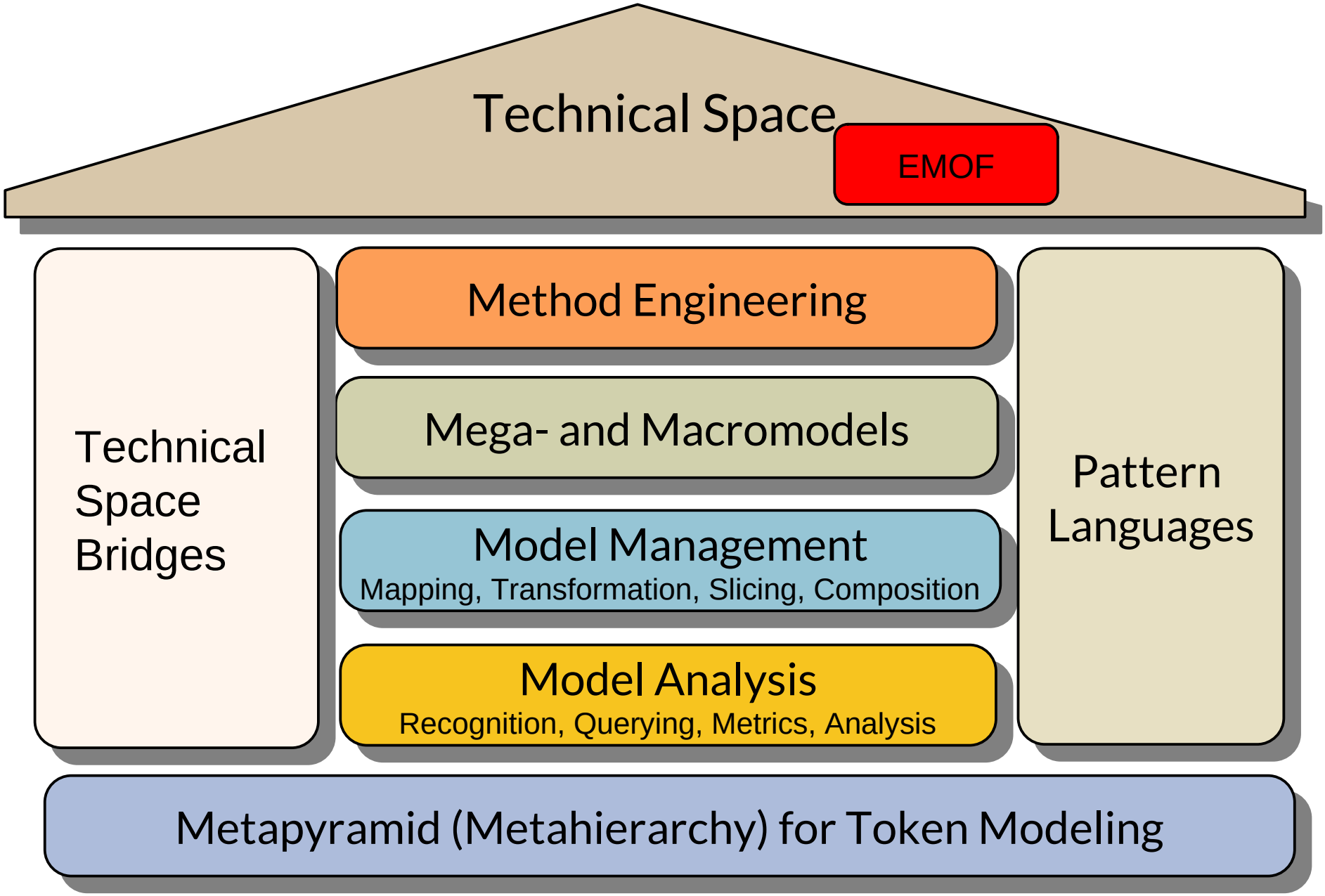
Obligatory Literature

- 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

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.

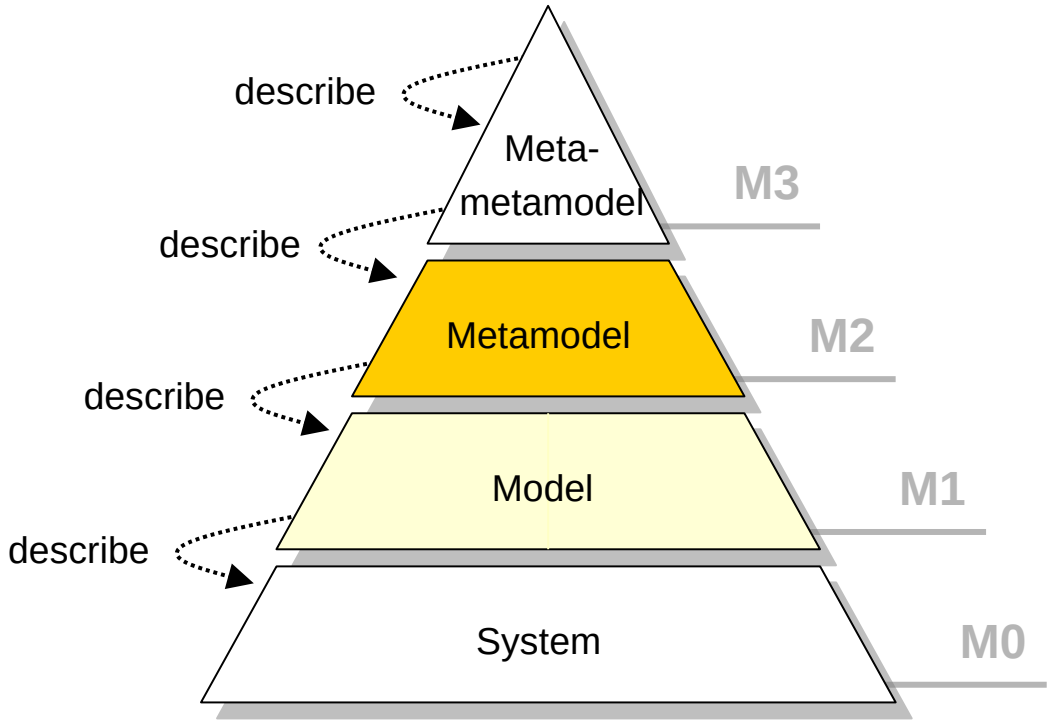
Q10: The House of a Technical Space



10.1 Metamodelling in the Classical Metapyramid

The Metamodel Hierarchy (Metapyramid, Metahierarchy)

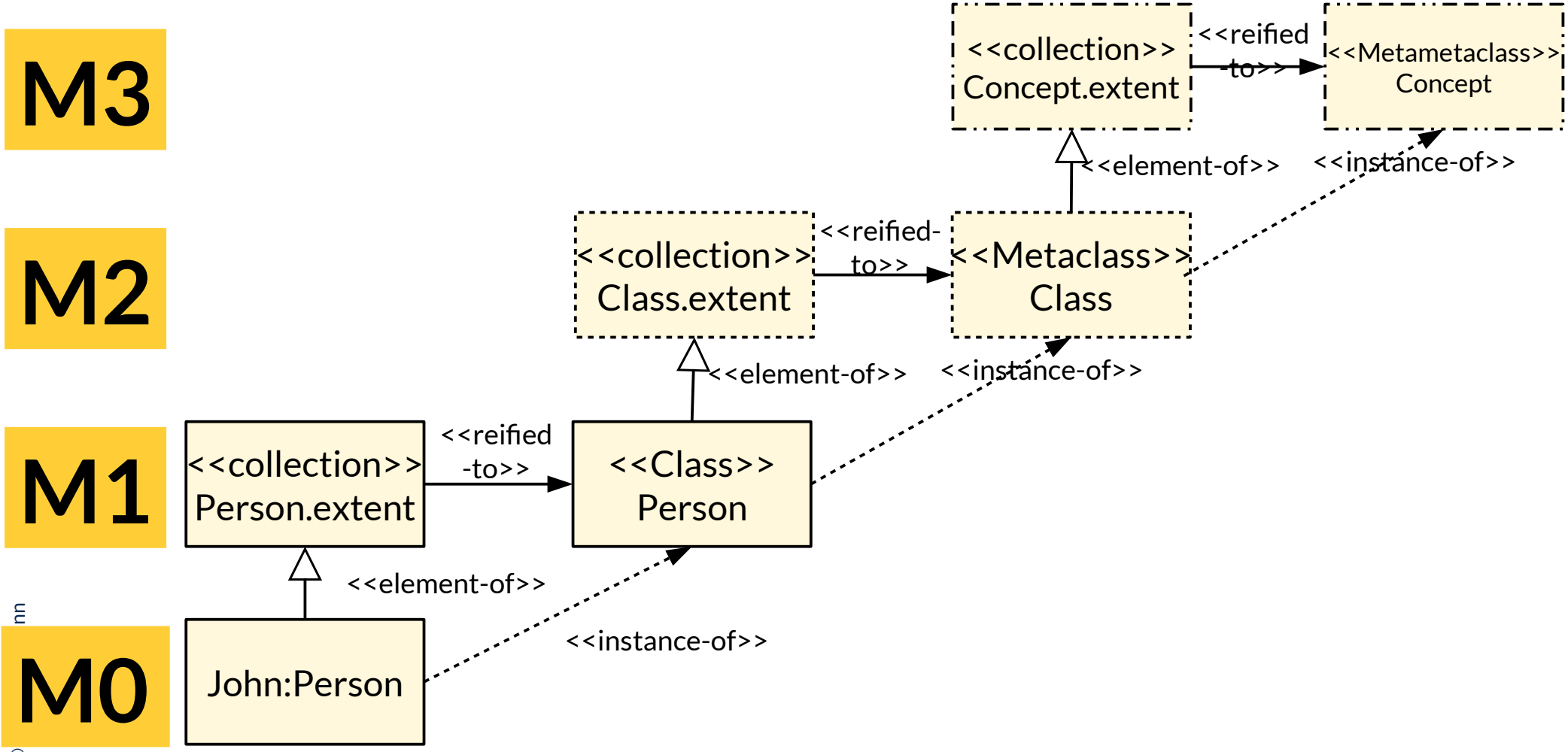
- Models are widely used in engineering disciplines
- Need for **tool support** that enables model-editing
- Domain experts want **domain specific languages** (DSL)
  → domain specific models with types from the domain
- Do not build model editors from scratch each time
  → **reuse** functionality
 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



[F. Klar, TU Darmstadt]

Remember: The Clabject Metahierarchy and Metapyramids

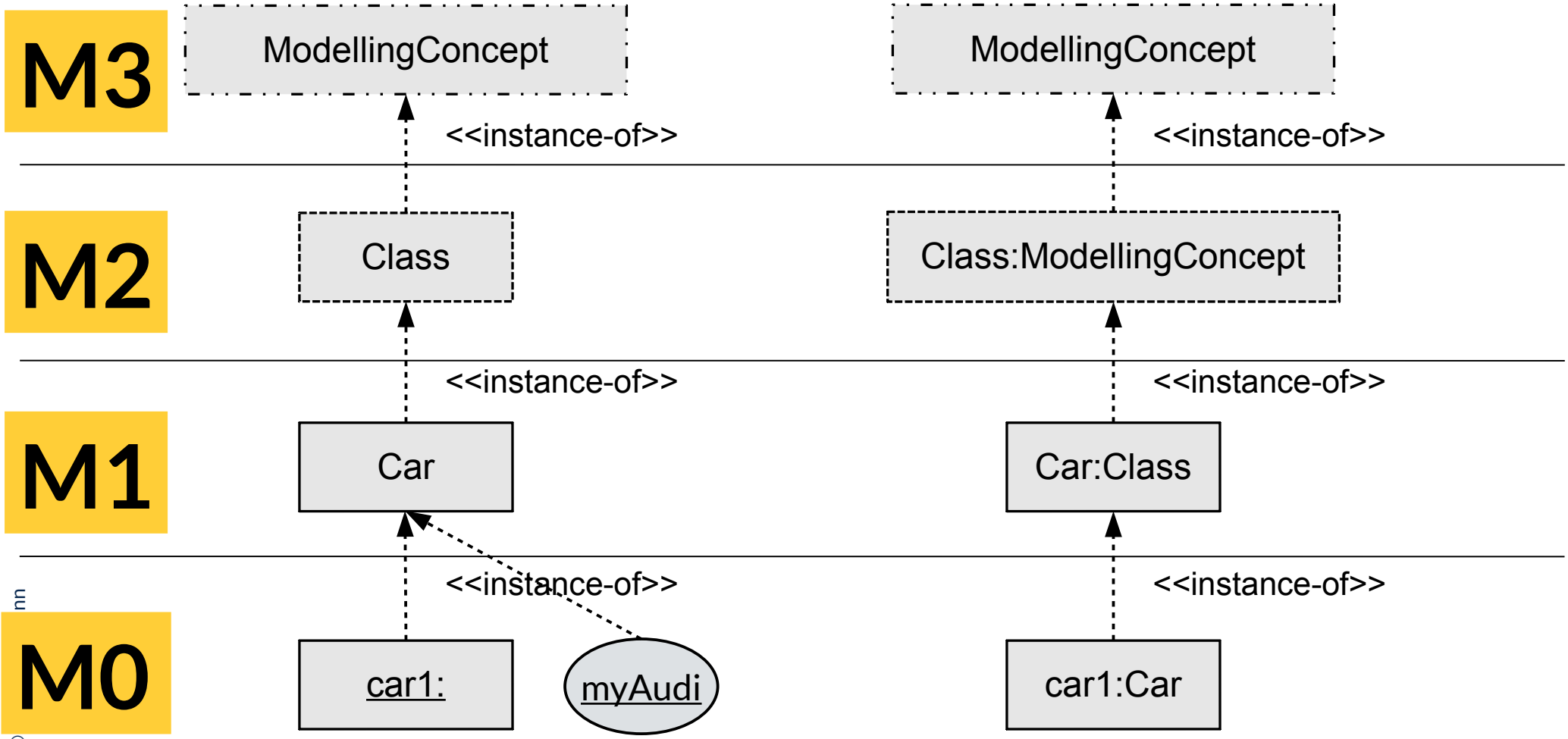
- We call a hierarchy of instance-of relationships a ***metahierarchy.***
- **A *metapyramid*** is a network of element-of, reified-to, and instance-of relationships



Notation

Clabject Hierarchy

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



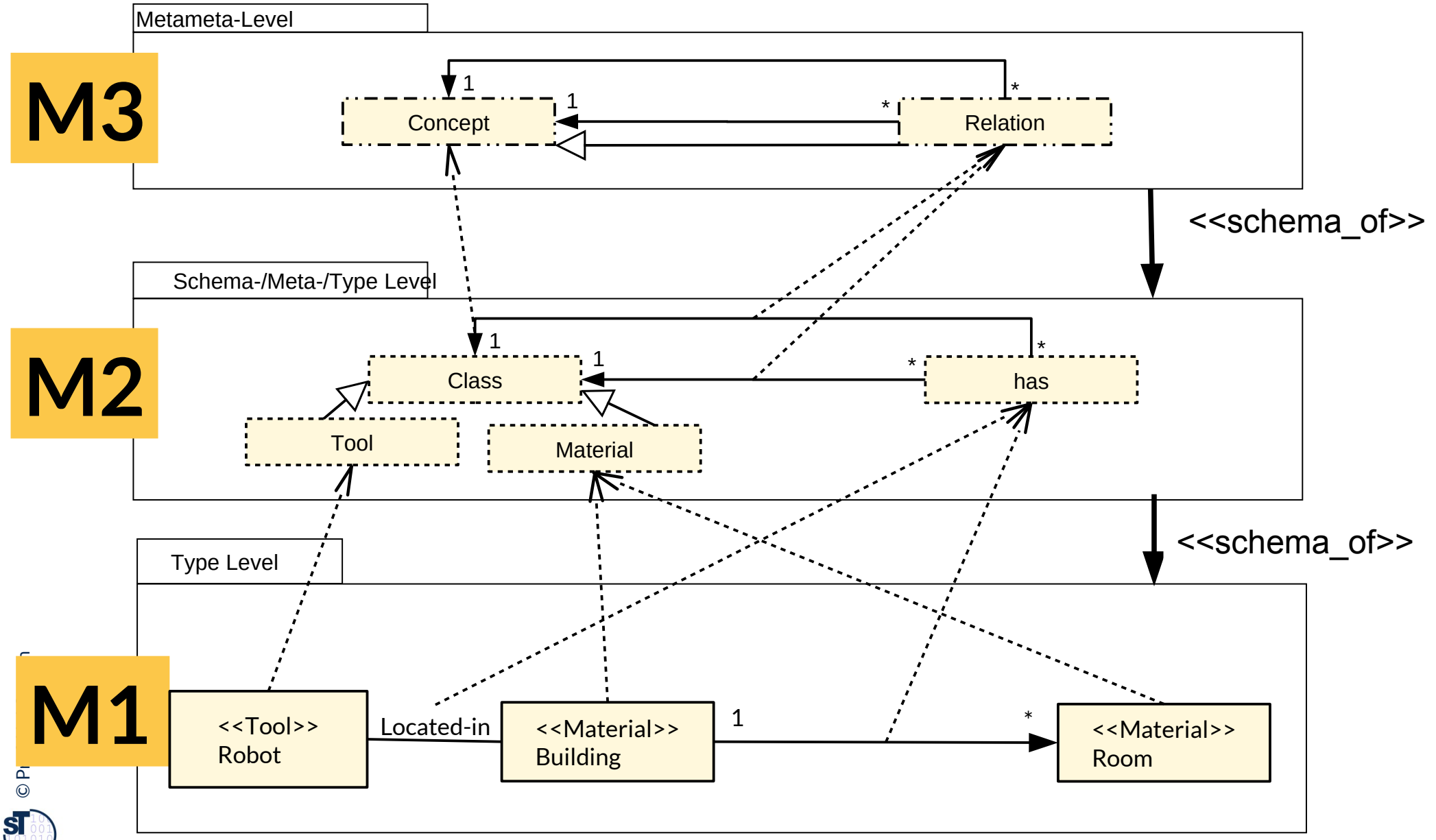
10.2 Metametamodels on M3

The Metametamodel (Metalanguage)

- 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

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)]

Metametamodels - Overview

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

10.2.1 Ecore and MOF as Simple Metametamodels

Overview of Metalanguage MOF (CMOF: Complete MOF)

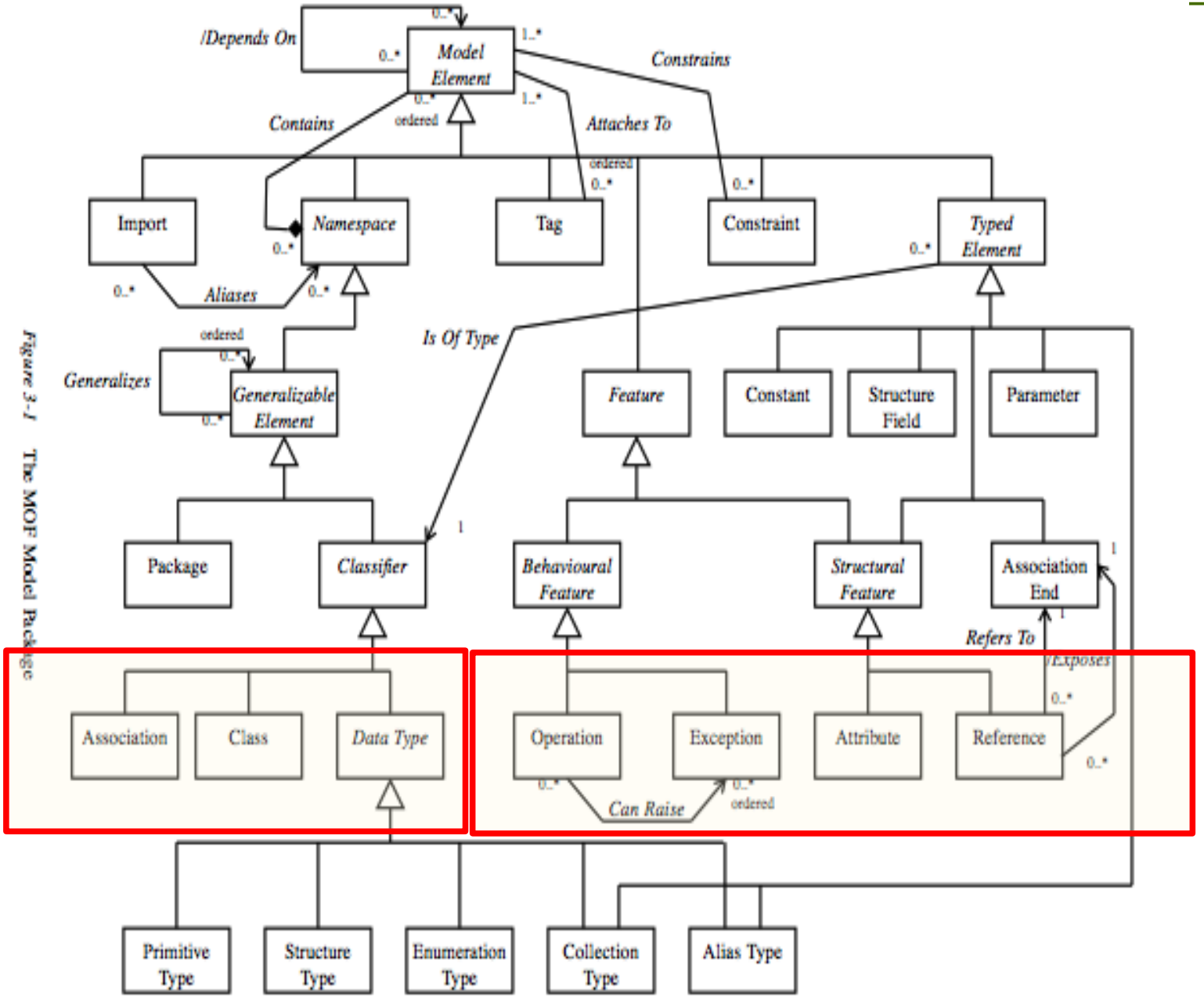
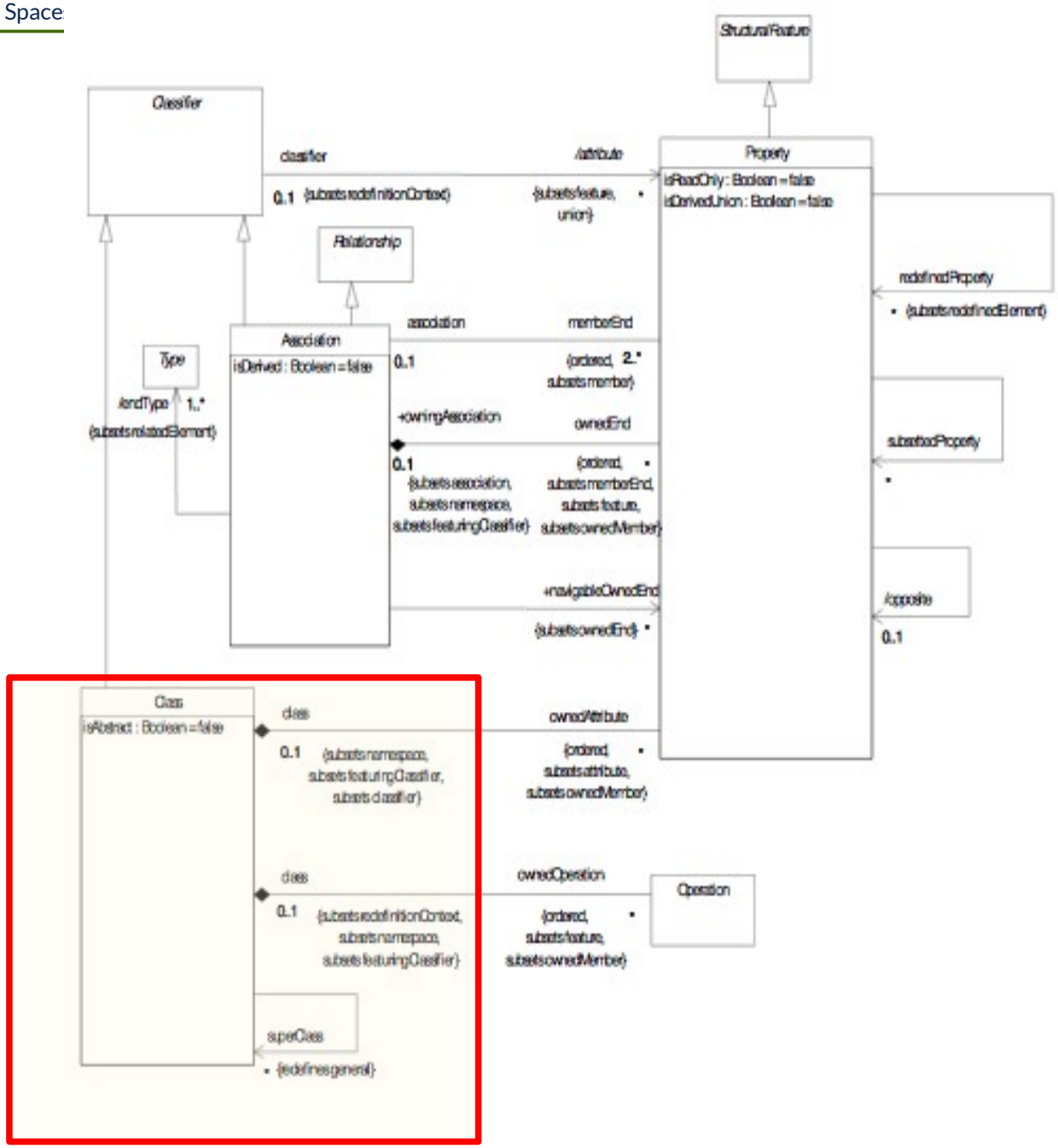


Figure 3-1 The MOF Model Package

[MOF]

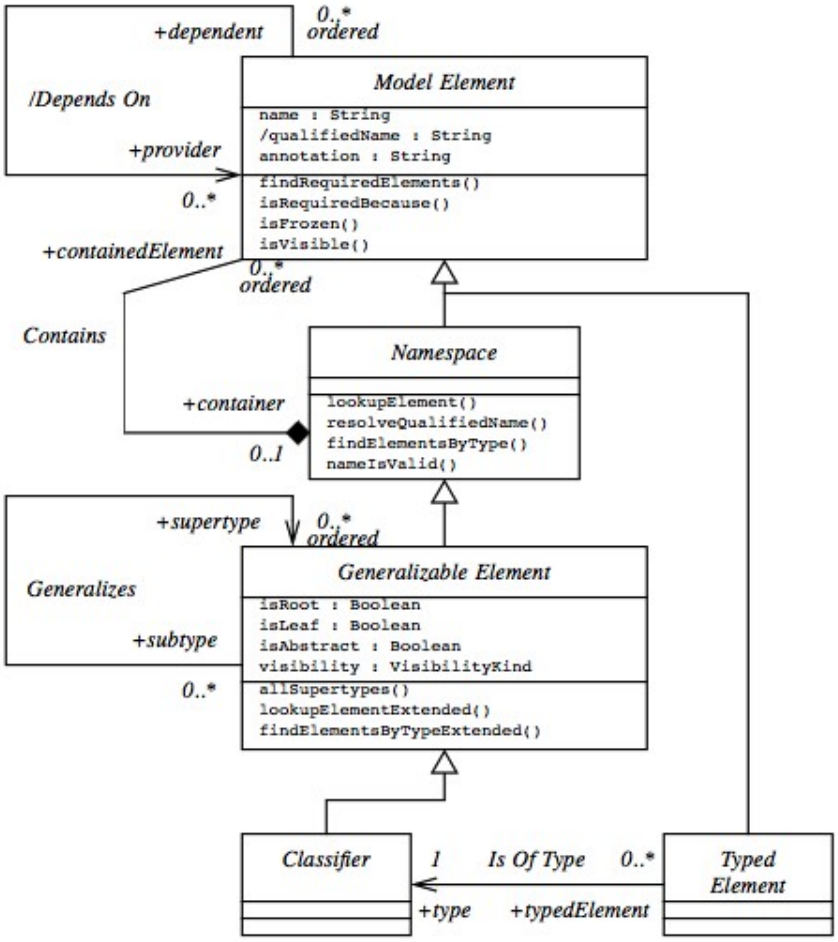
UML Core

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



[MOF]

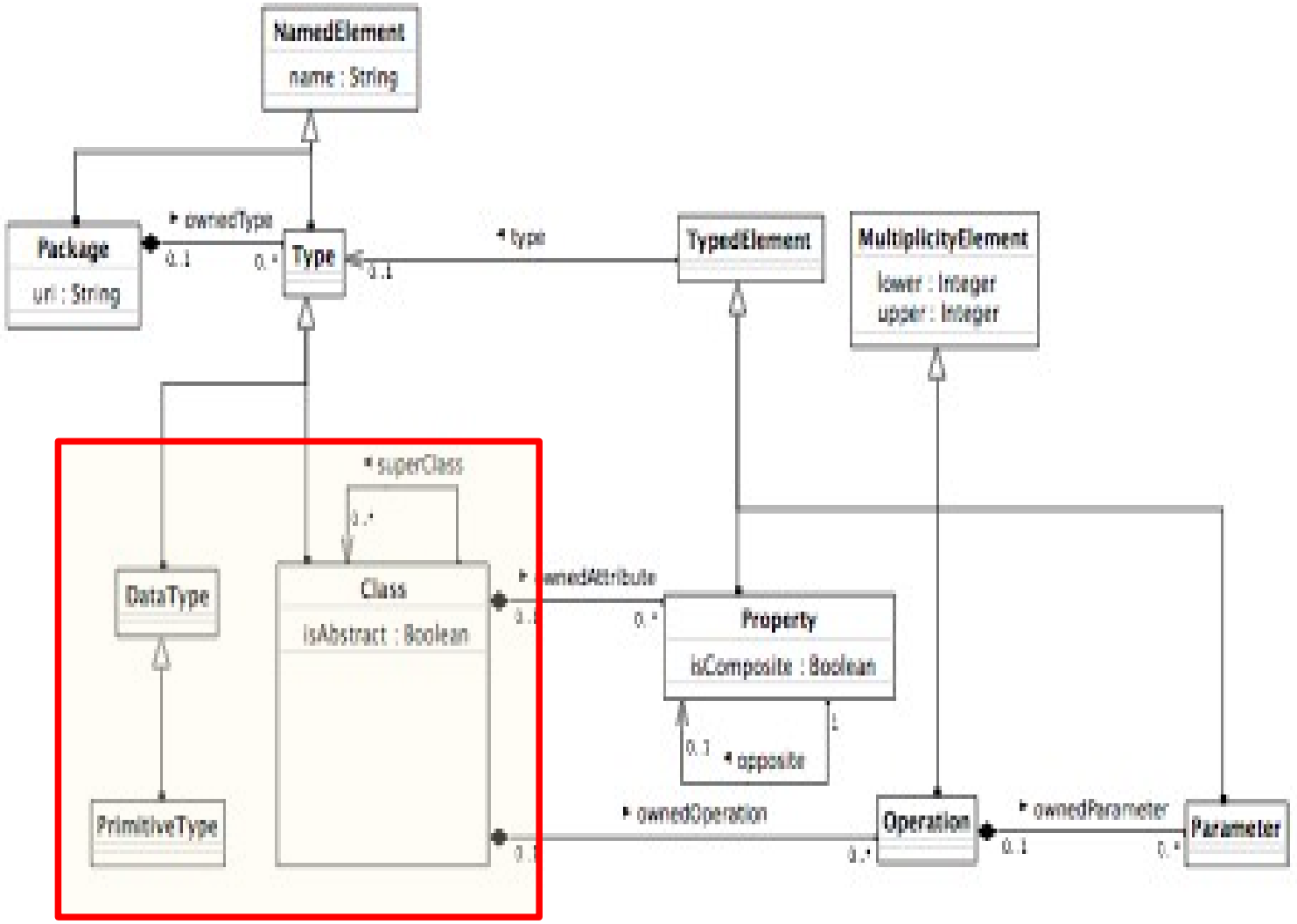
MOF Central Types



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

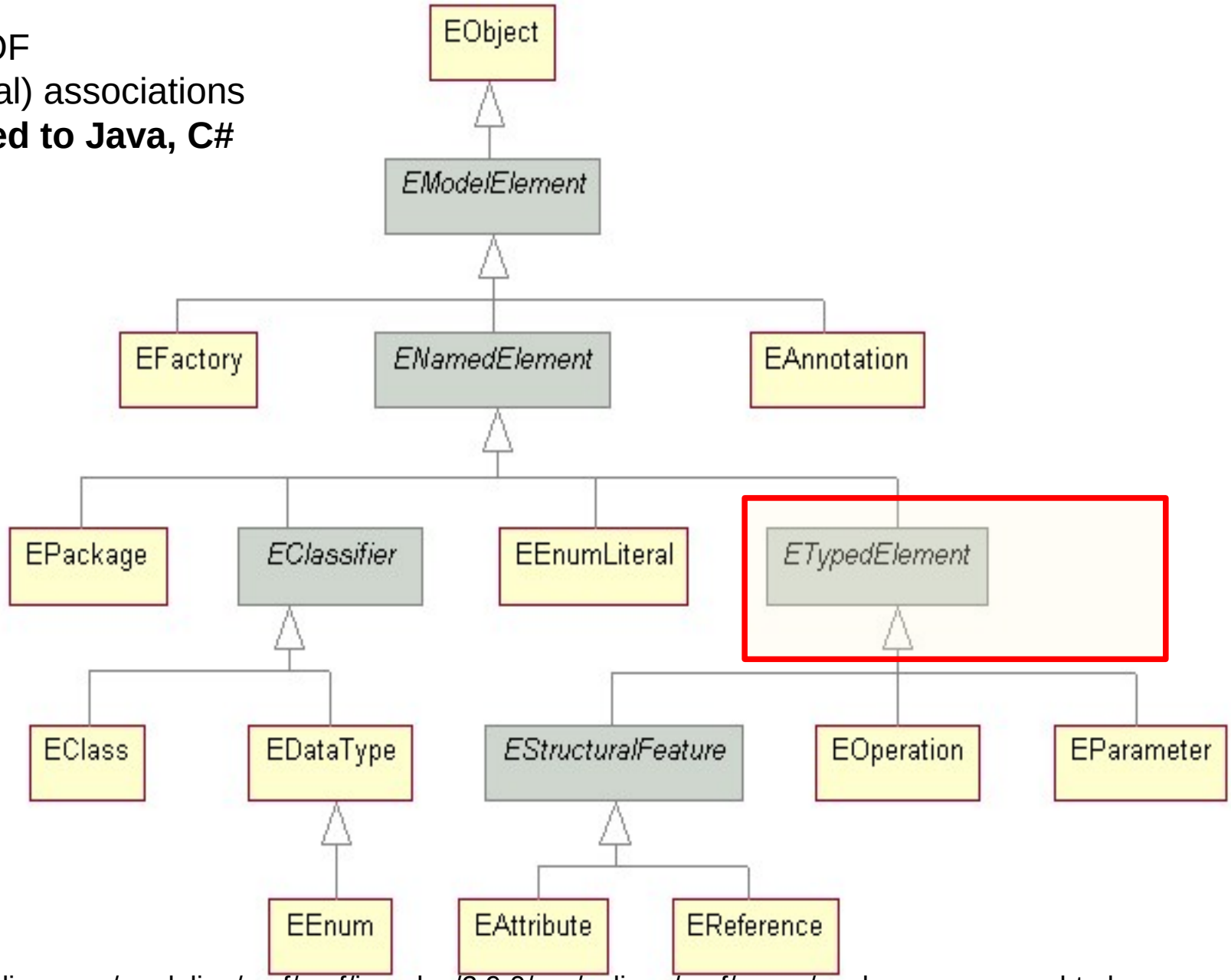
[MOF]

Central MOF Metaclasses with Associations

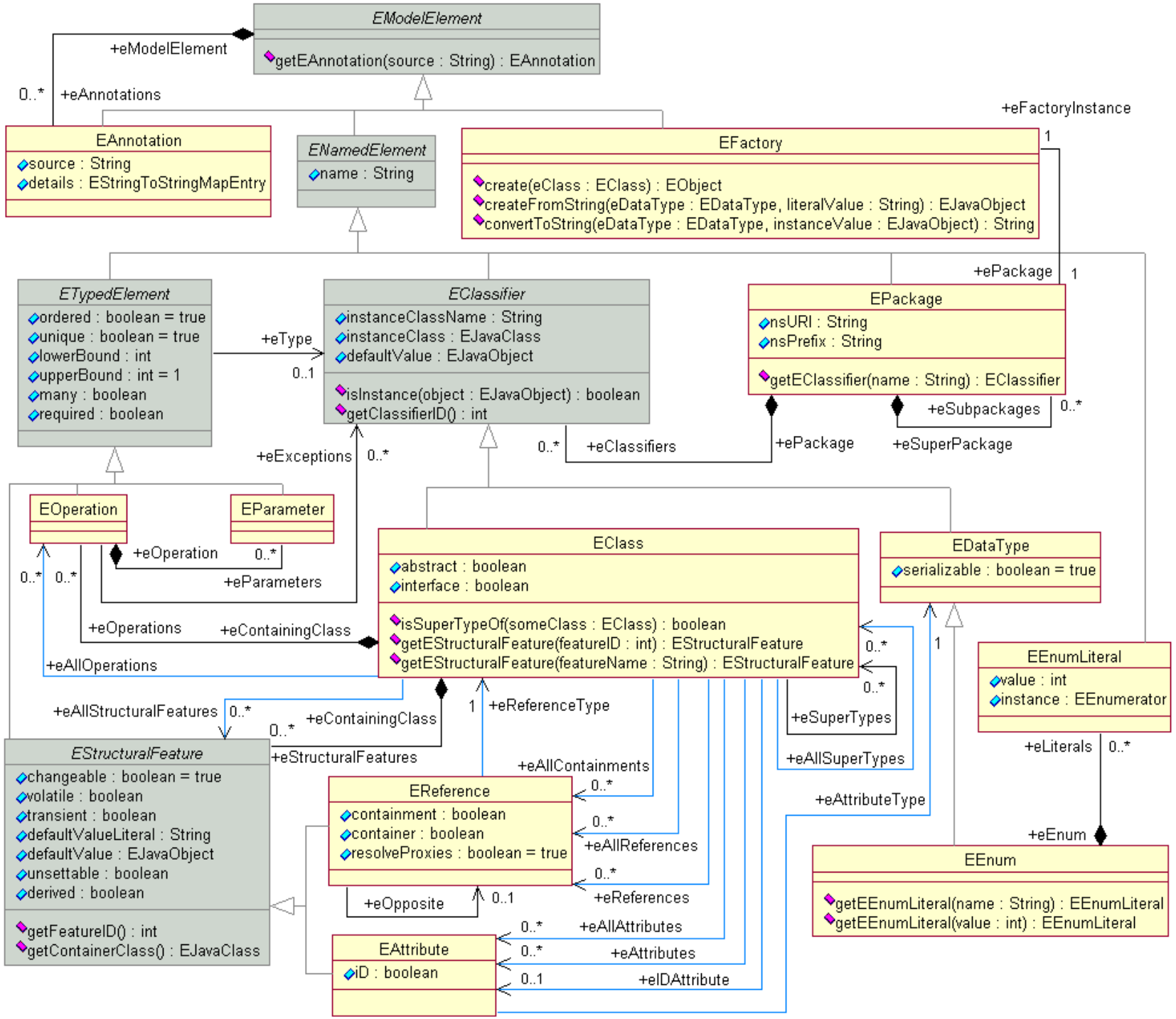


EMOF (Essential MOF)

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

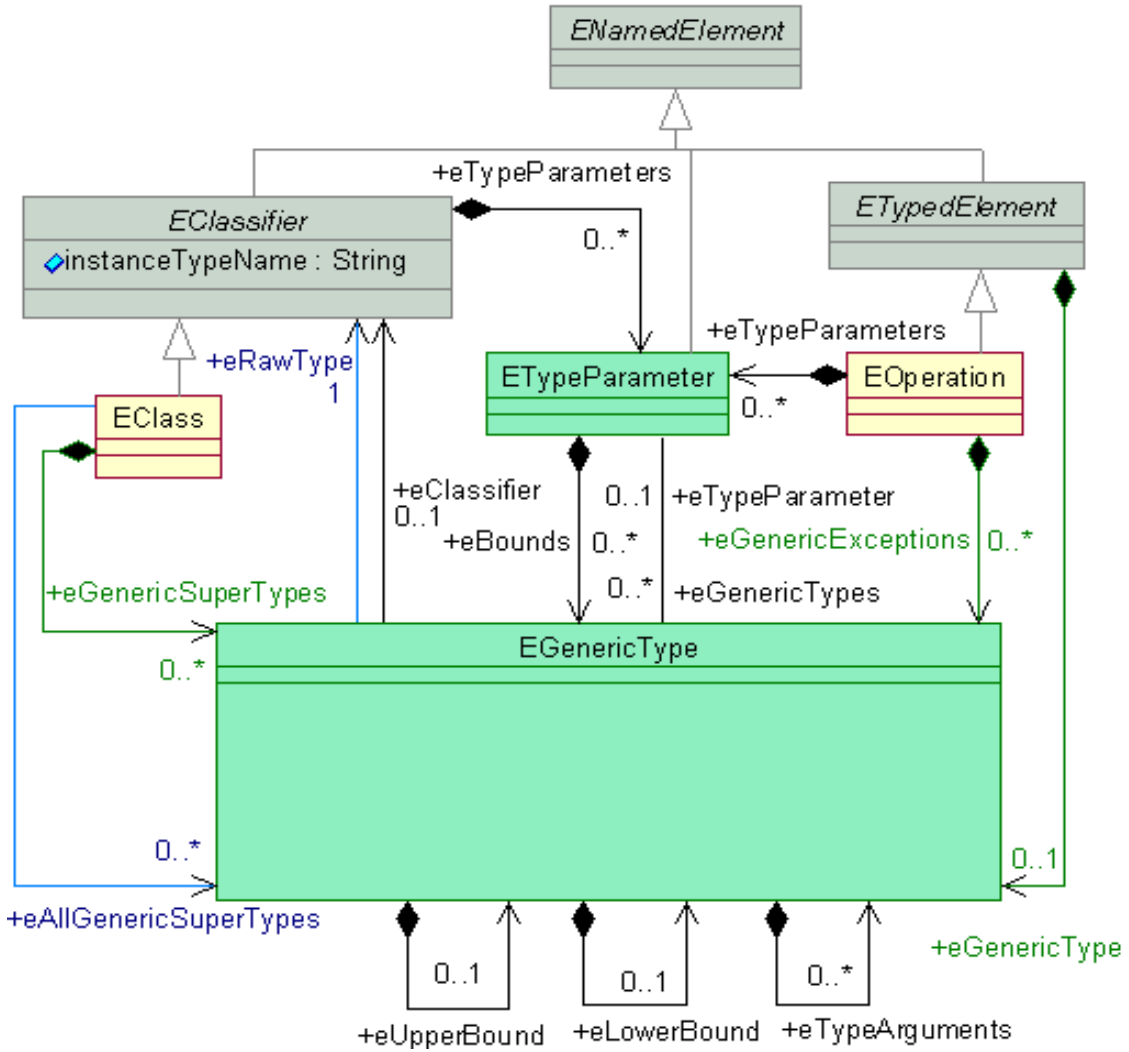


[MOF]
http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html



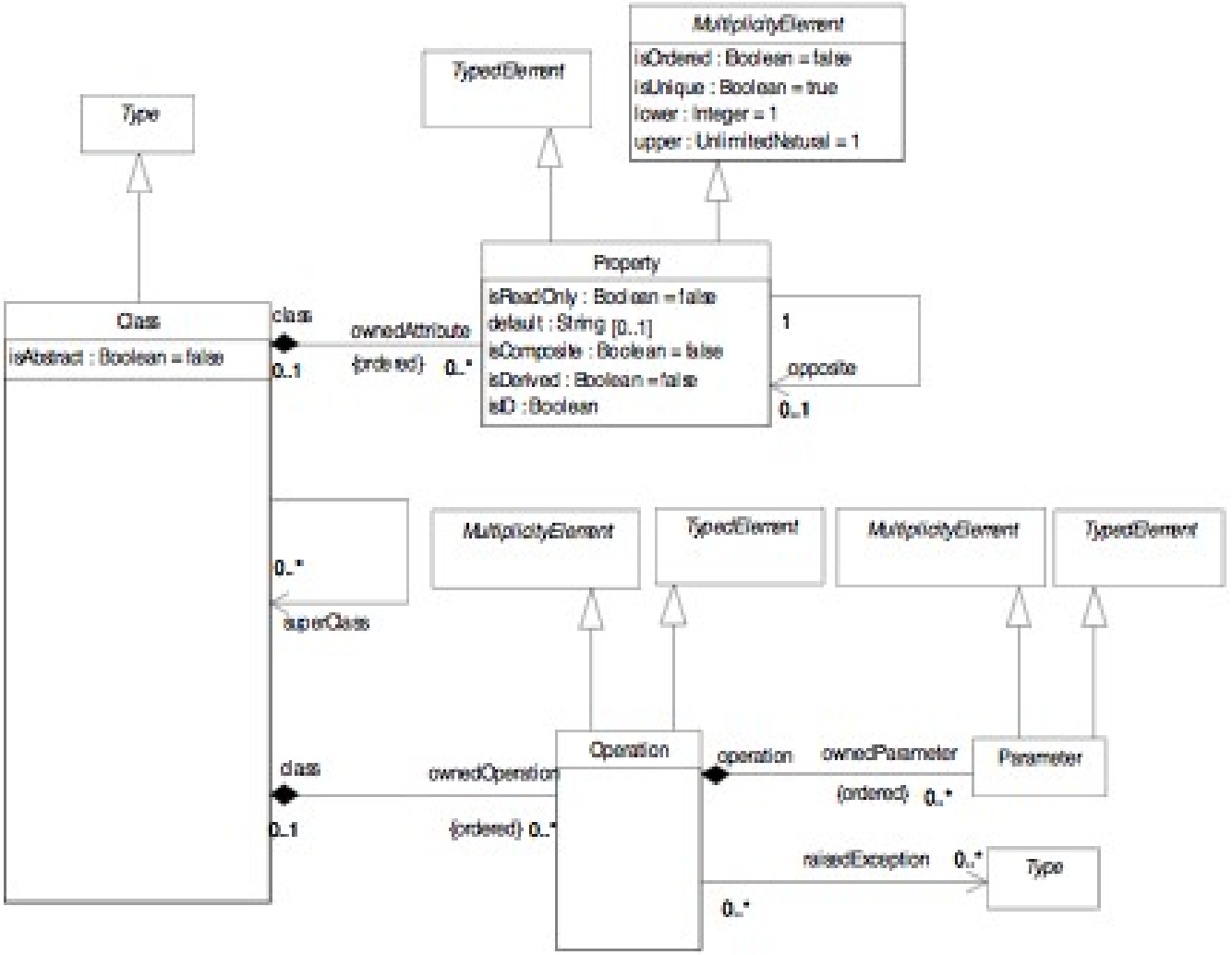
http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html

Generic Types



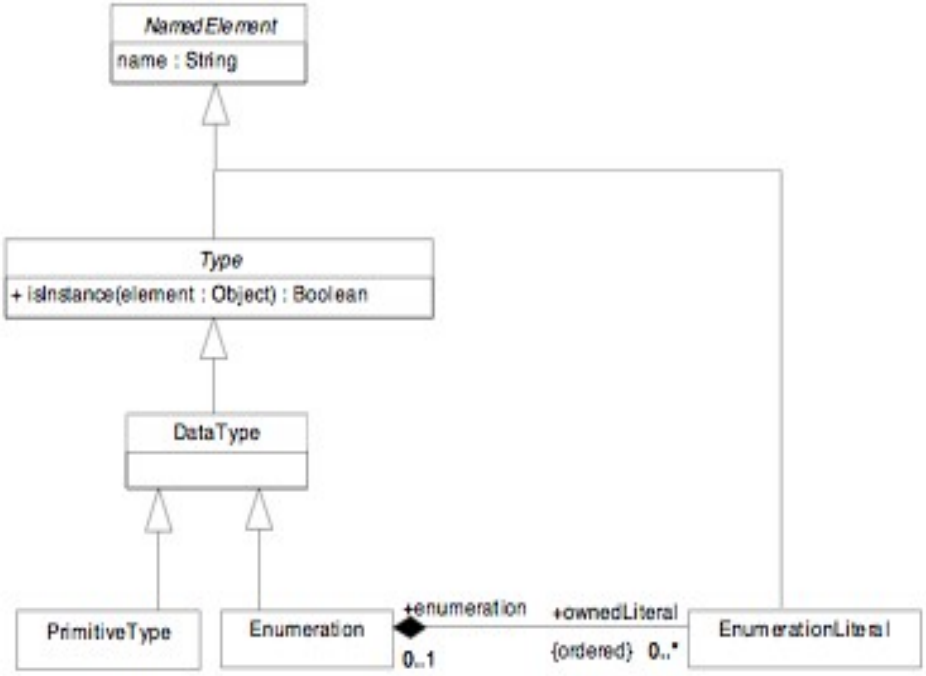
http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html

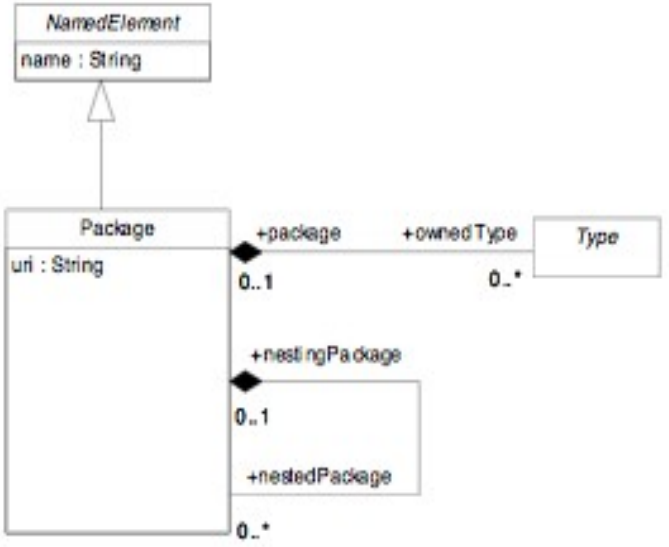
EMOF Classes in Detail



[MOF]

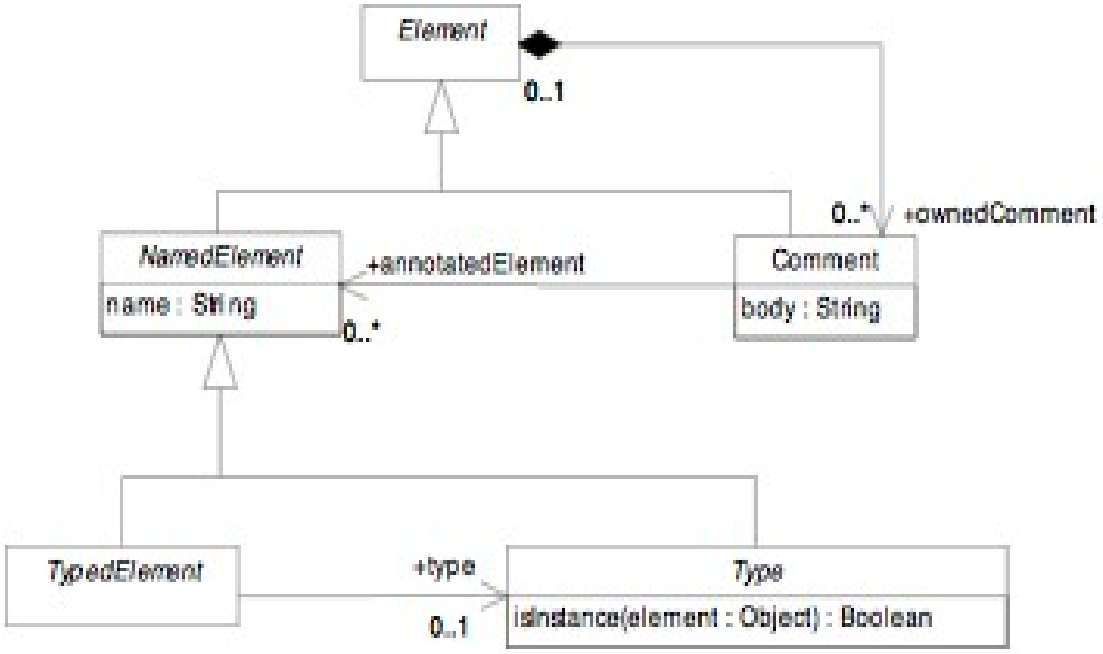
EMOF Data Types and Packages





[MOF]

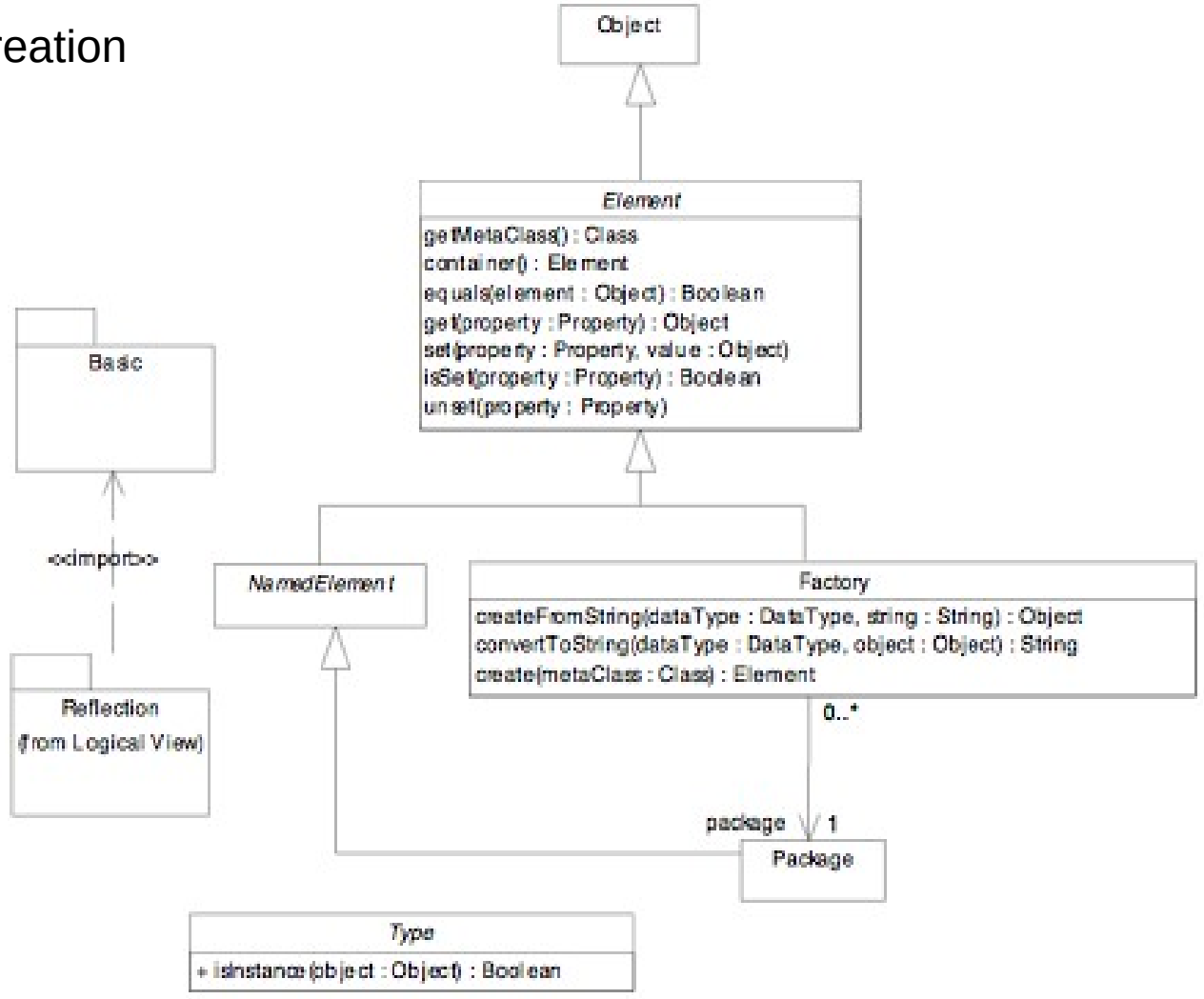
EMOF Types



[MOF]

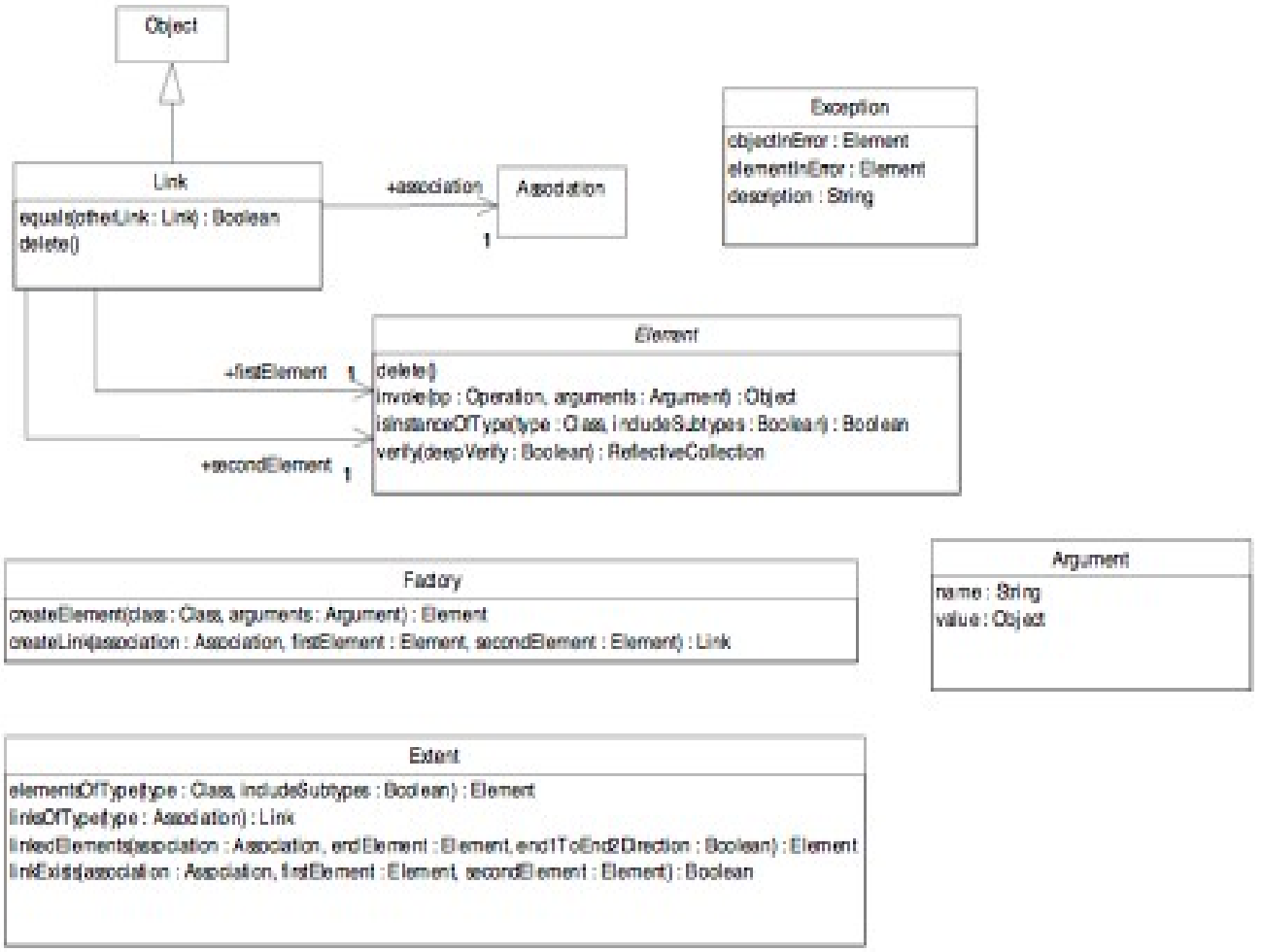
EMOF Reflection

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



[MOF]

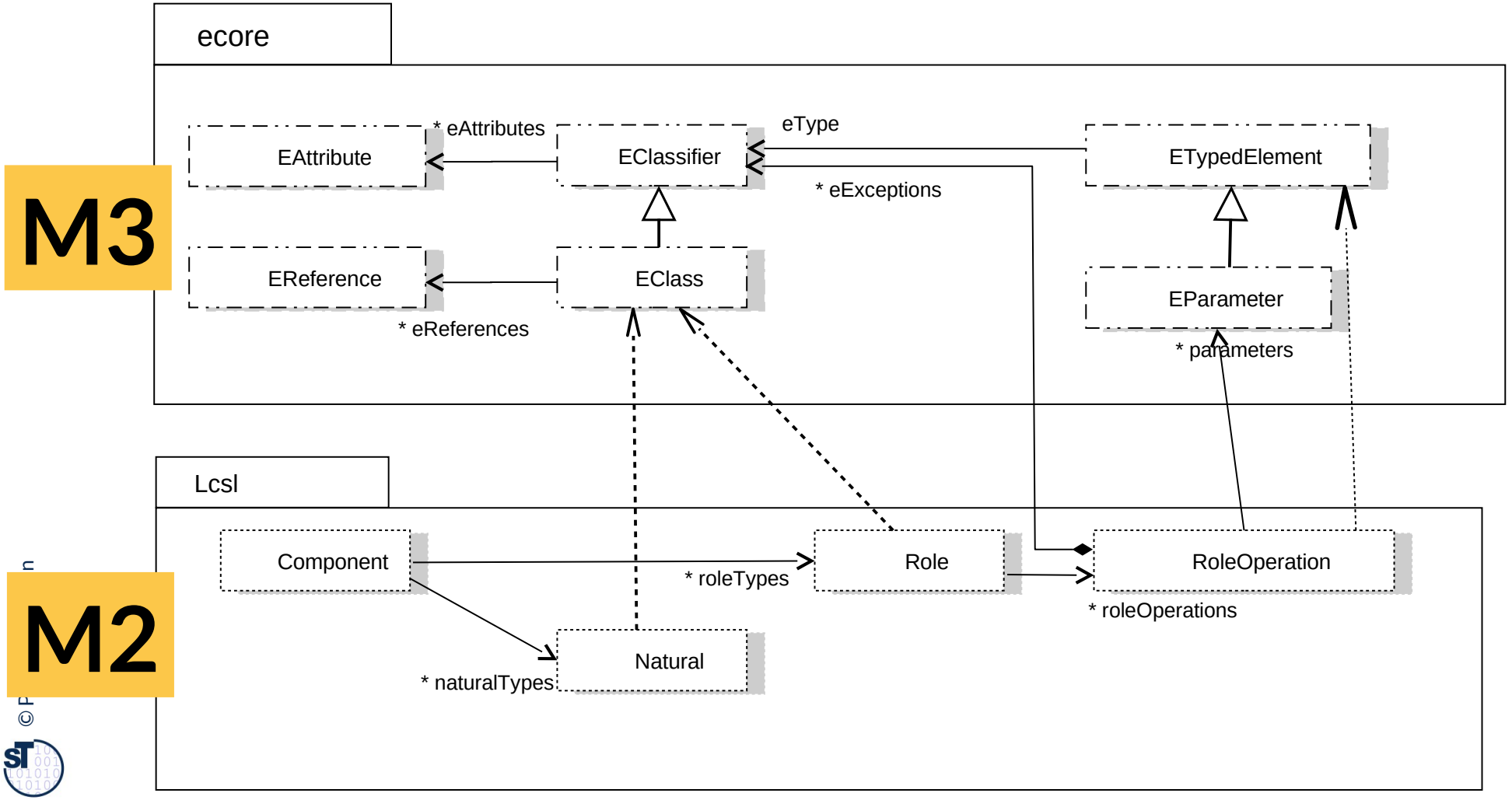
CMOF Reflection



[MOF]

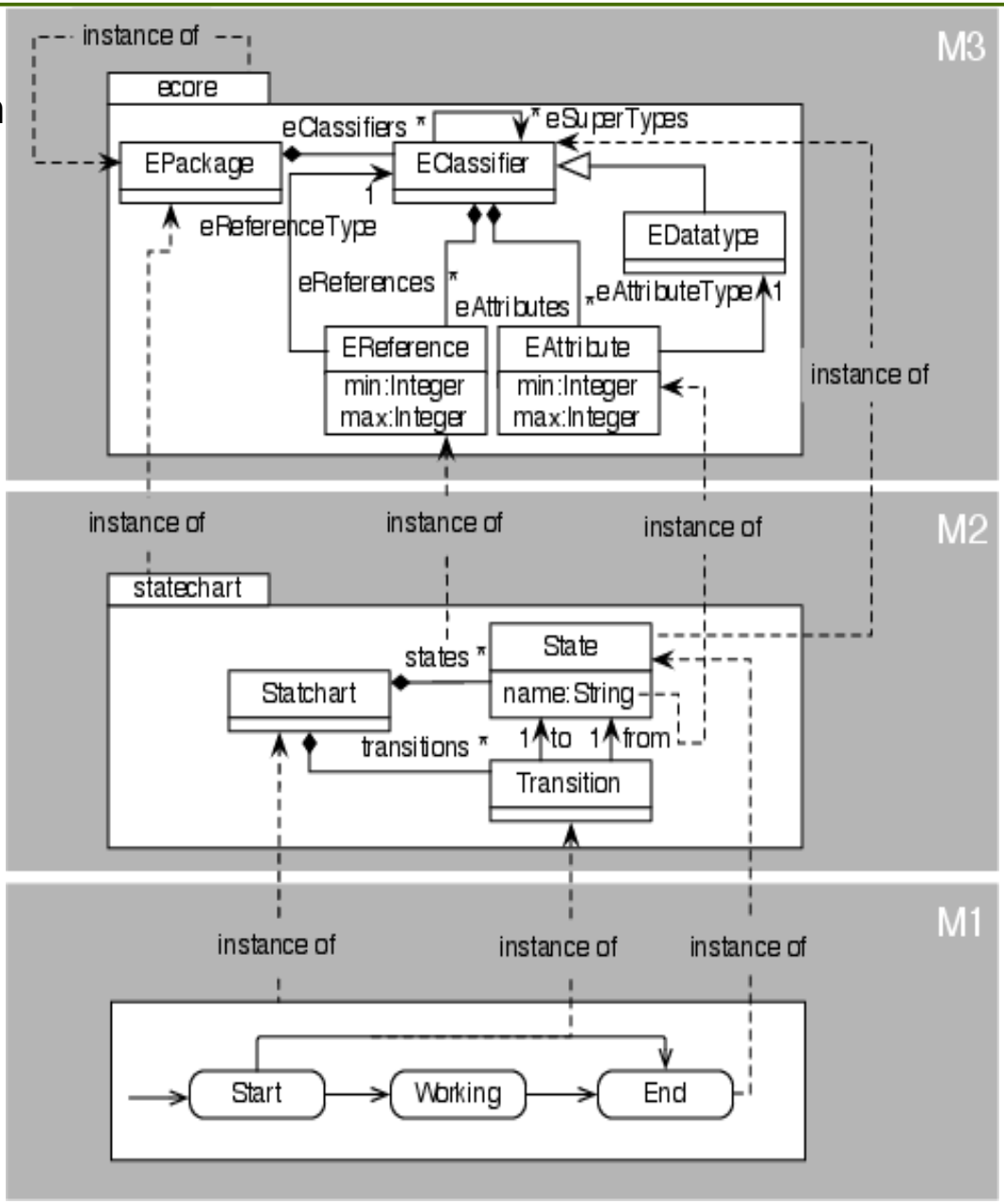
Ex.: Deriving a DSL from EMOF and its Implementation Eclipse ecore

- 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)



10.2.2 Lifting of a Metamodel to a Metametamodel

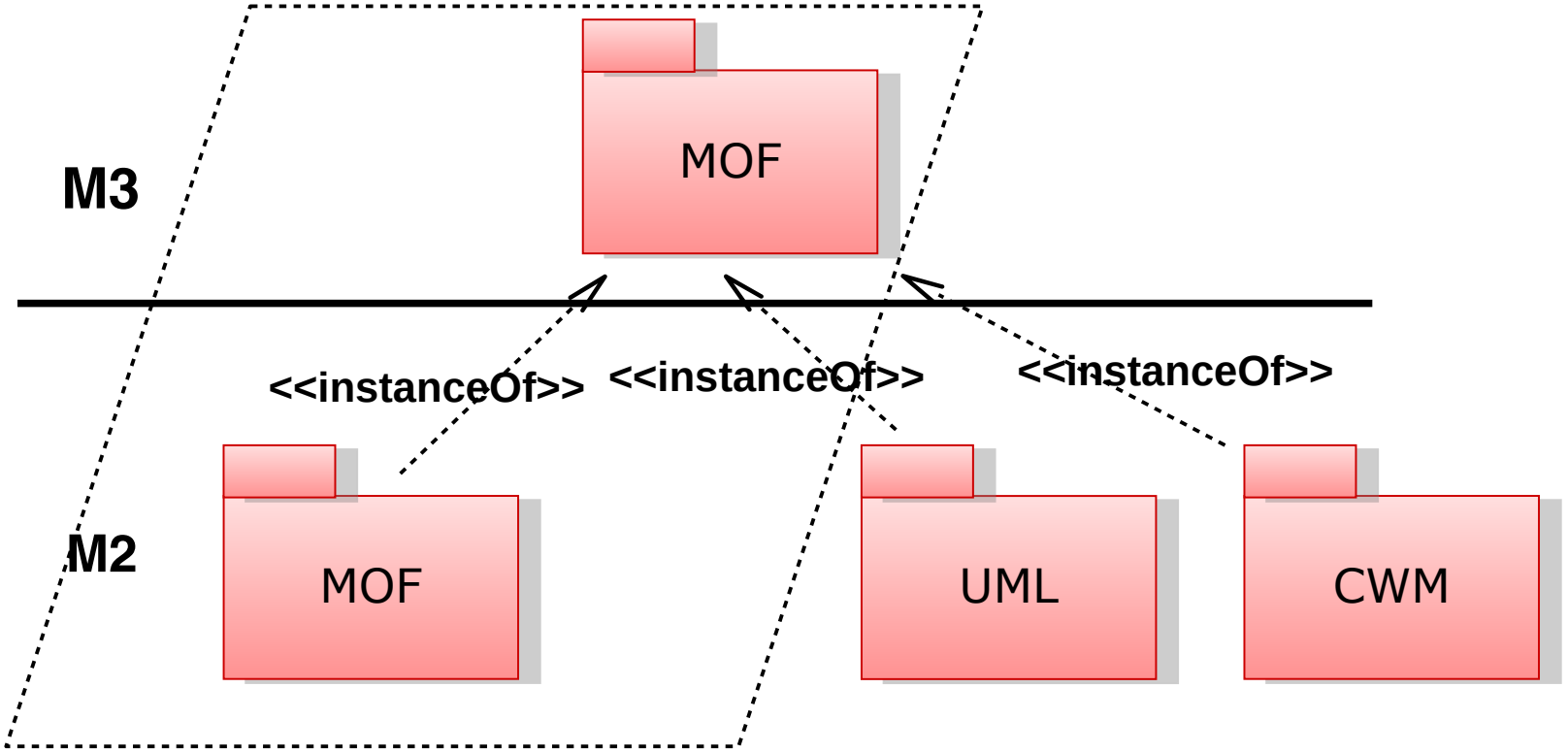
Lifting of Metamodels

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

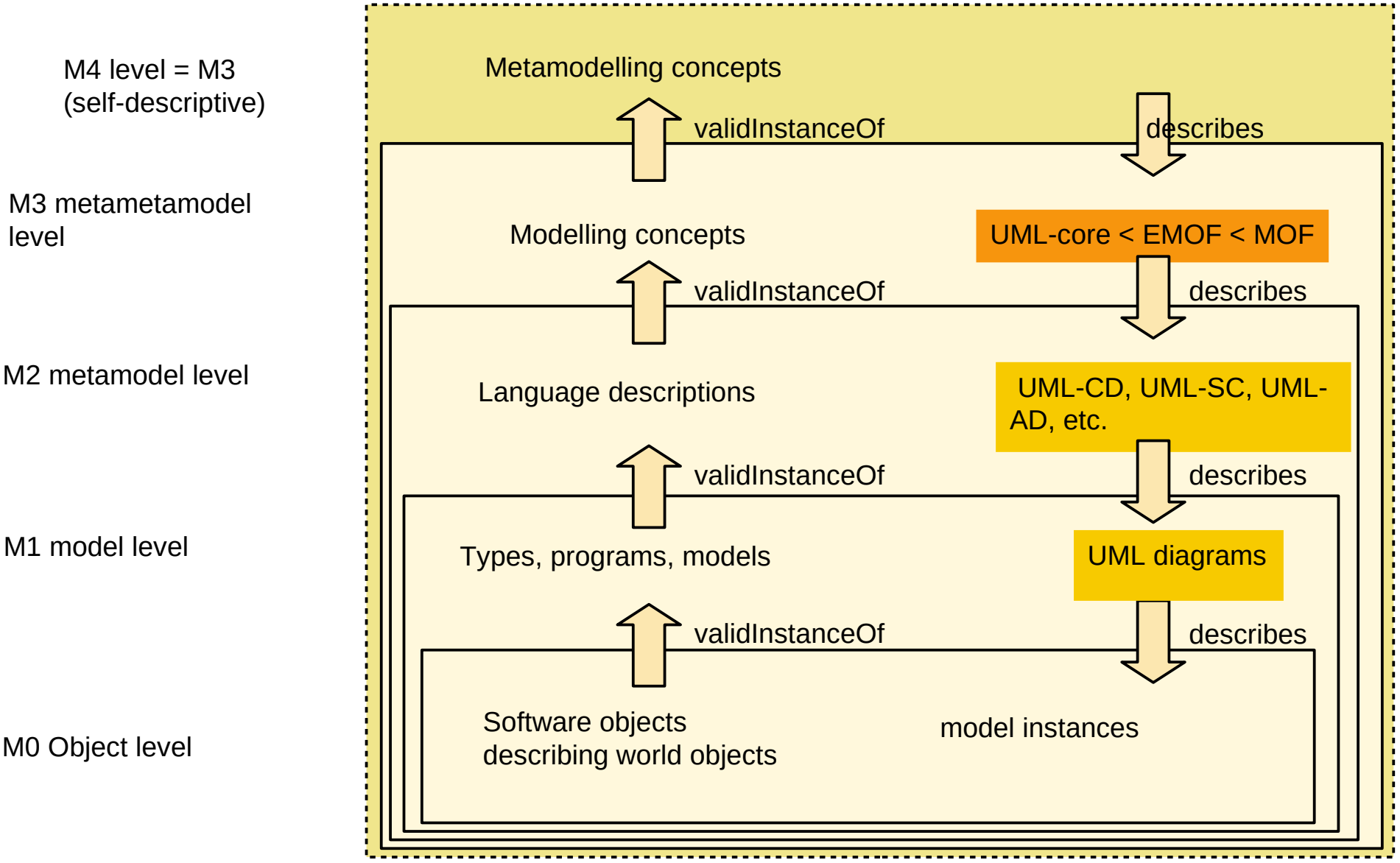
Self-Descriptive MOF

- 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

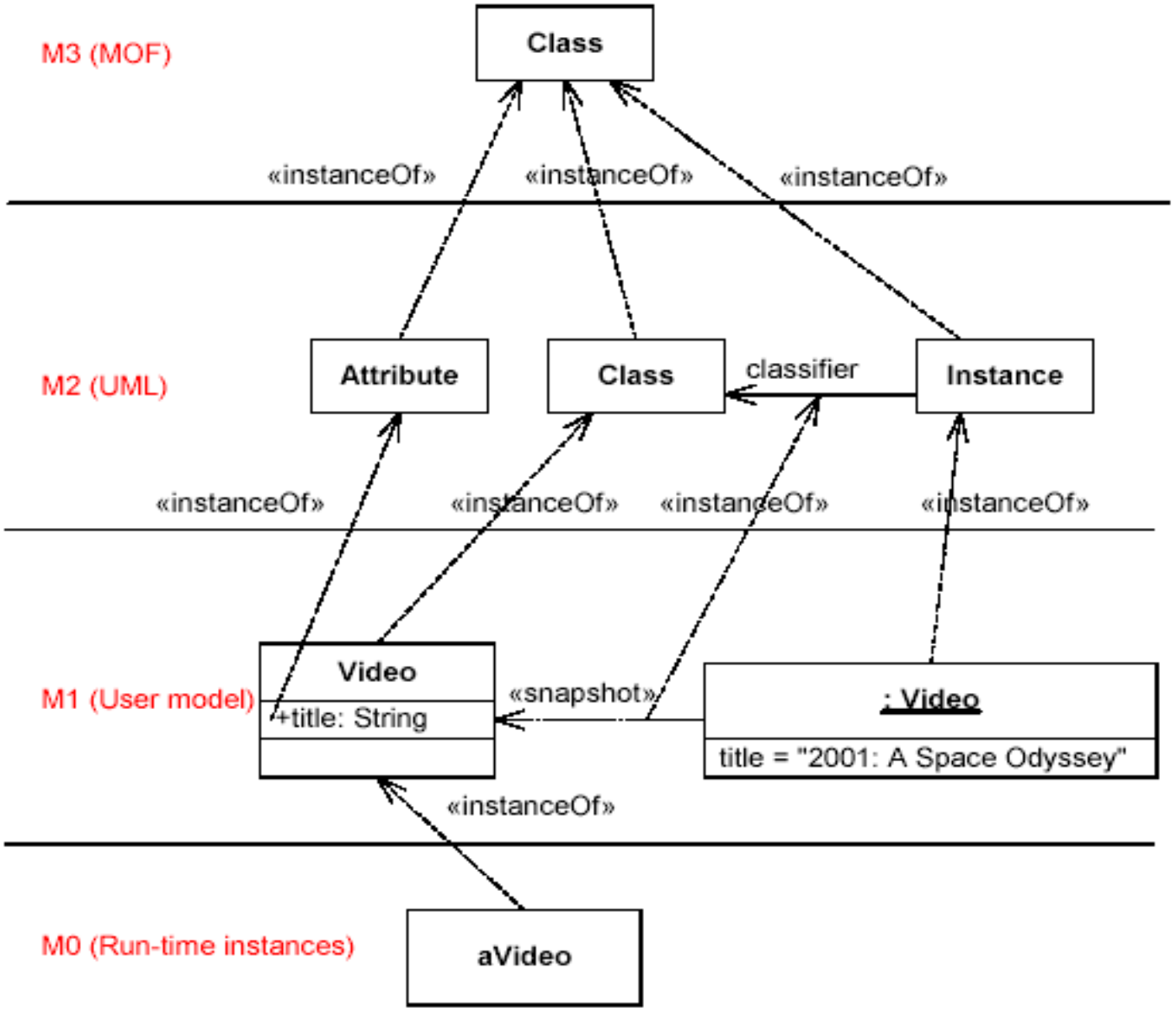


The UML-Core/MOF Metahierarchy

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

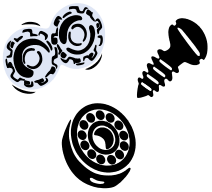


Ex.: MOF-Metahierarchy for UML



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

Metamodeling – Benefits



Constraints

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

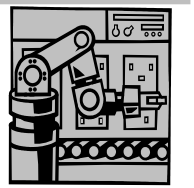
- (Meta-)Modeling of language constructs
- Definiton of language structure
- Domain specific semantics

Metamodel

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

Abstract Syntax

Transformation



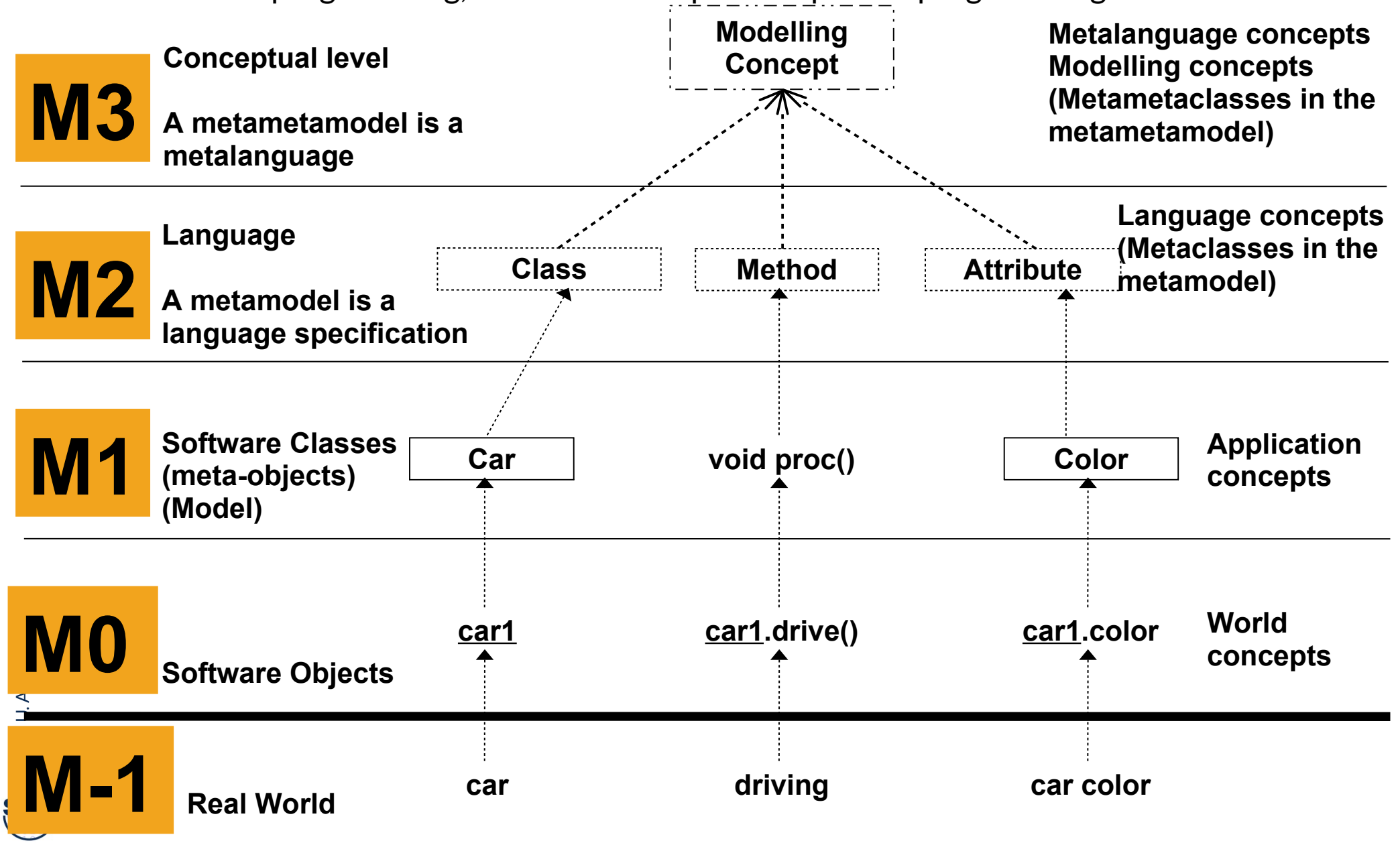
Model

[F. Klar, TU Darmstadt]

10.2.3 Metahierarchies for Metaprogramming

Metalevels in Programming Languages (The Metahierarchy for Metaprogramming)

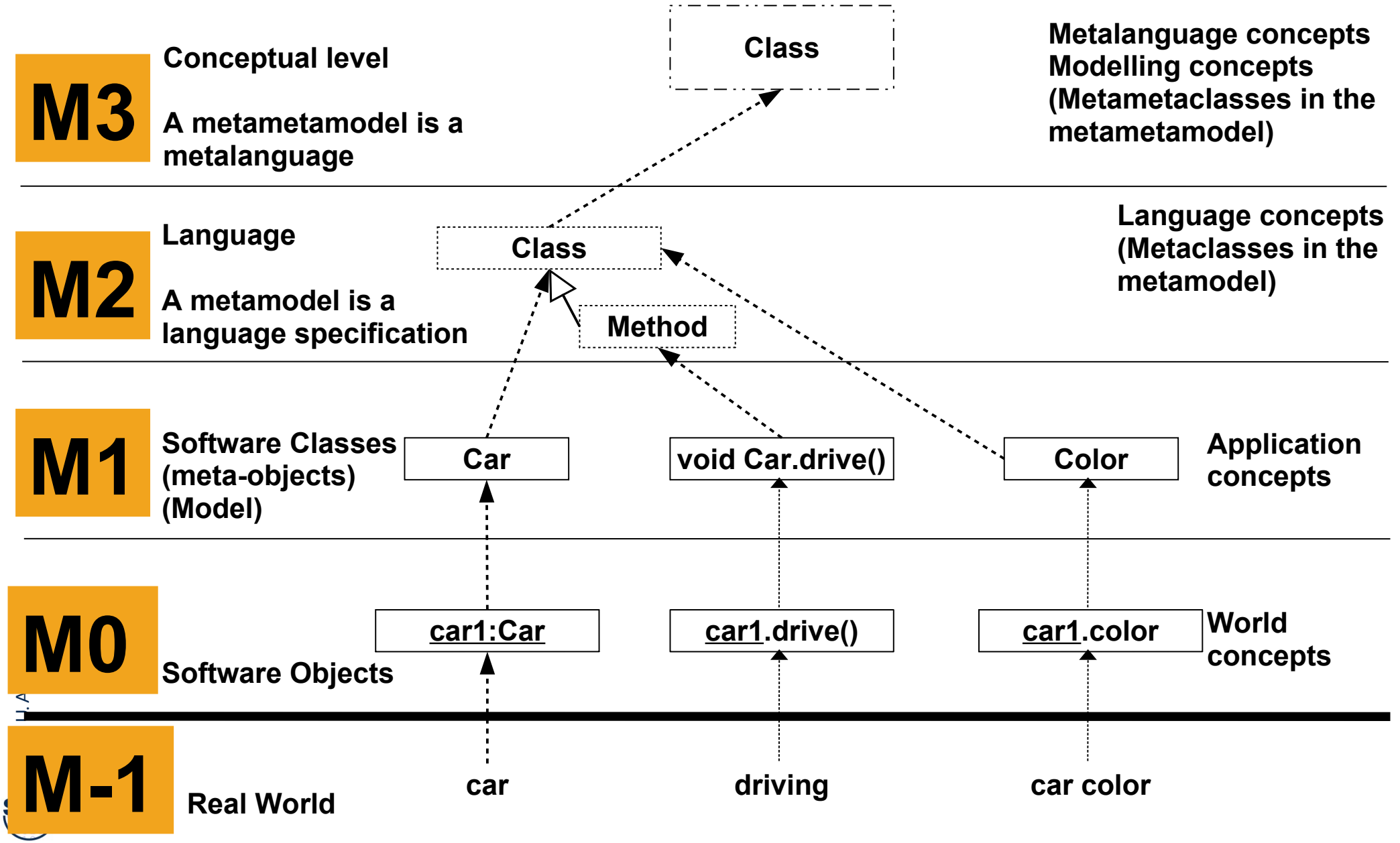
- In Metaprogramming, all meta*-concepts are open for programming



Excursion: Metaprogramming

- **Metaprograms (reflective programs)** generate code on the basis of a metamodel of their own language (self model)
- **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

Metalevels in Smalltalk



The End

- Compare MOF and EMOF. Why do many programmers like EMOF more than MOF?
- Explain the advantages that MOF sopports general associations.
- What is the purpose of a metametamodel?
- Would it make sense to use TAM on the M3 level, i.e., in the metametamodel?
- Explain why TAM stereotypes do not occur on M2.

Metaprograms

- **Metaprogramme** sind Programme, die Programme erzeugen oder transformieren.
  - Sie haben seit Lisp eine lange Tradition: Lisp erlaubt ungetypte Metaprogramme
- **Dynamische Metaprogramme** laufen ständig mit der Anwendung mit und regenerieren Teile neu.
  - Dynamische Metaprogramme sind allerdings laufzeitintensiv und verhindern eine statische Analyse der dynamisch produzierten Programme.
- **Introspektive Programme** inspizieren die Metadaten oder den Code anderer Programme und Komponenten und ziehen dadurch Schlüsse
- **Codegeneratoren** sind Metaprogramme, die mit Introspektion von Modellen oder Programmen neuen Code produzieren und alten Code nicht invalidieren
- **Statische Metaprogramme** produzieren ein Programm, in dem sie selbst nicht enthalten sind.
  - Sie funktionieren also als reiner Code-Generator.

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).