Modellbasierte Softwareentwicklung (MODSOFT)

Part II

Domain Specific Languages

Structure

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

{fischer,scheidge,blunk}@informatik.hu-berlin.de

LFE Systemanalyse, III.310

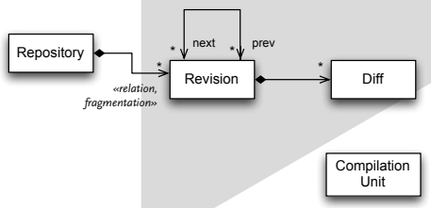
Agenda

prolog (1 VL)	**Introduction:** languages and their aspects, modeling vs. programming, meta-modeling and the 4 layer model
0. (2 VL)	**Eclipse/Plug-ins:** eclipse, plug-in model and plug-in description, features, *p2*-repositories, *RCP*s
➡ 1. (2 VL)	**Structure:** *Ecore*, *genmodel*, working with generated code, constraints with *Java* and *OCL*, *XML/XMI*
2. (3 VL)	**Notation:** Customizing the tree-editor, textural with *XText*, graphical with *GEF* and *GMF*
3. (4 VL)	**Semantics:** interpreters with Java, code-generation with *Java* and *XTend*, model-transformations with *Java* and *ATL*
epilog (2 VL)	**Tools:** persisting large models, model versioning and comparison, model evolution and co-adaption, modular languages with *XBase*, *Meta Programming System* (MPS)

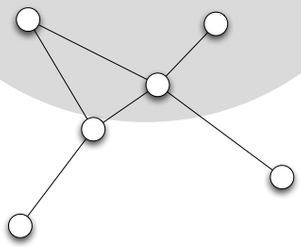
Previously on MODSOFT

Language Aspects

notation

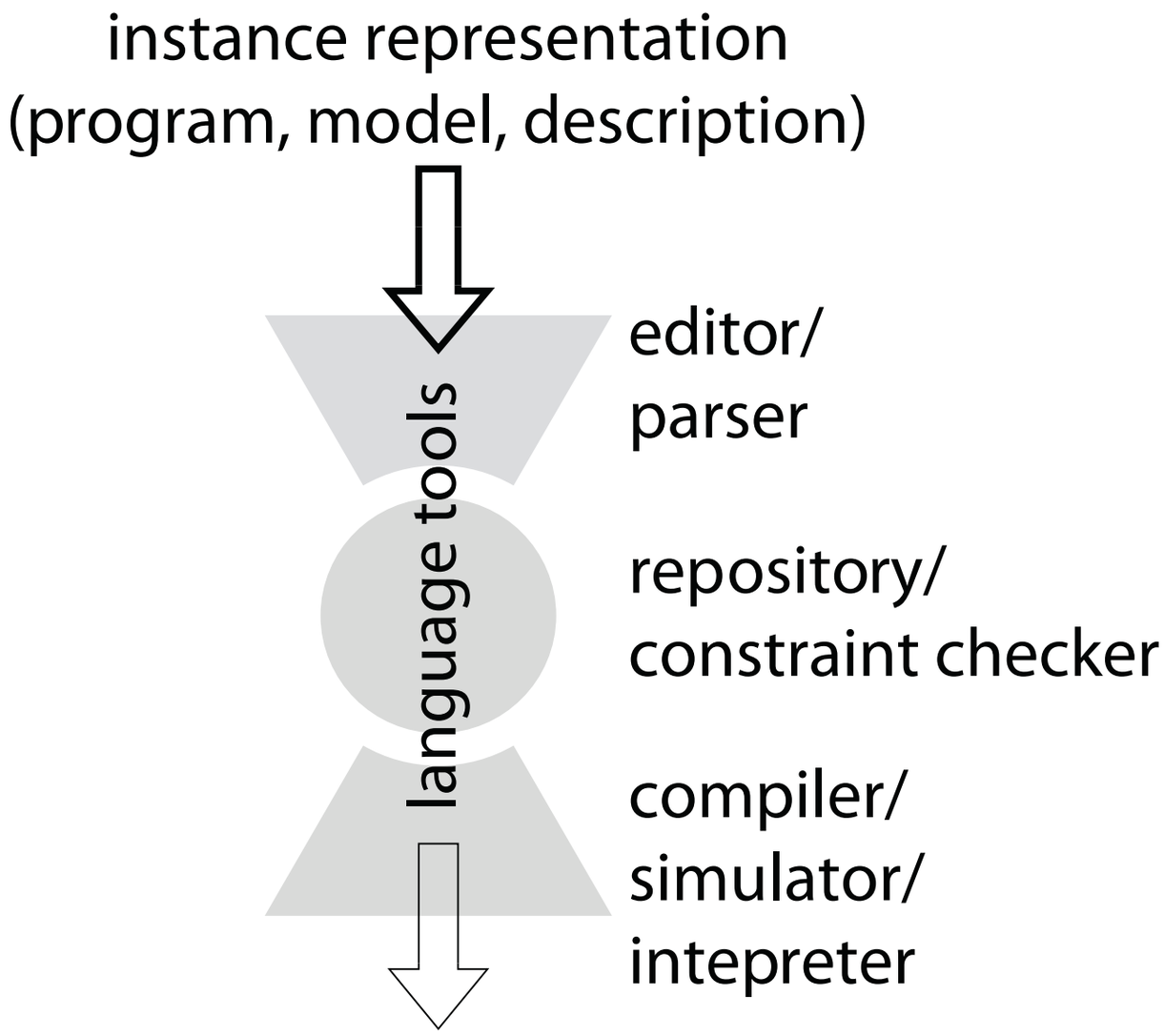


structure

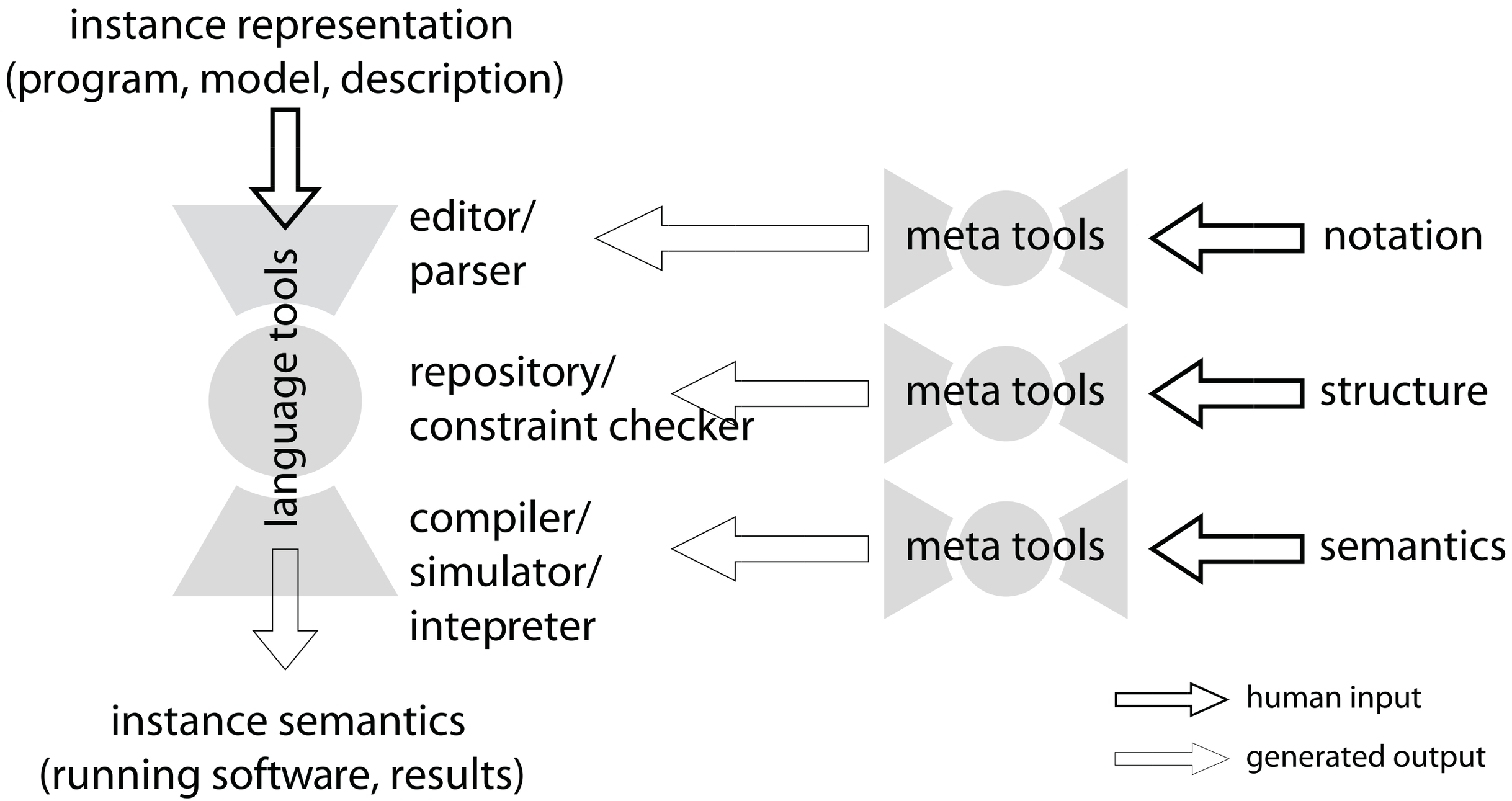


semantics

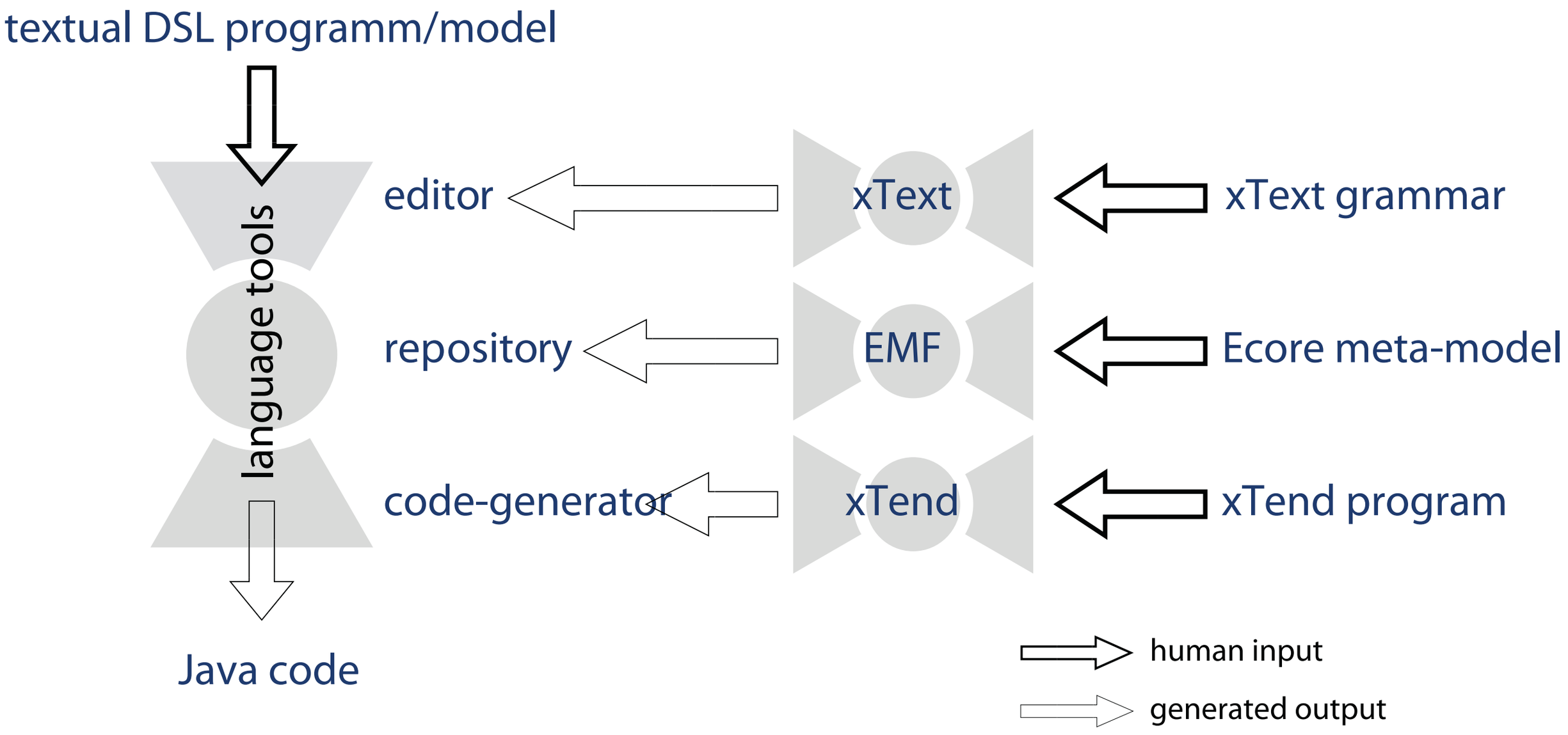
Meta-Languages



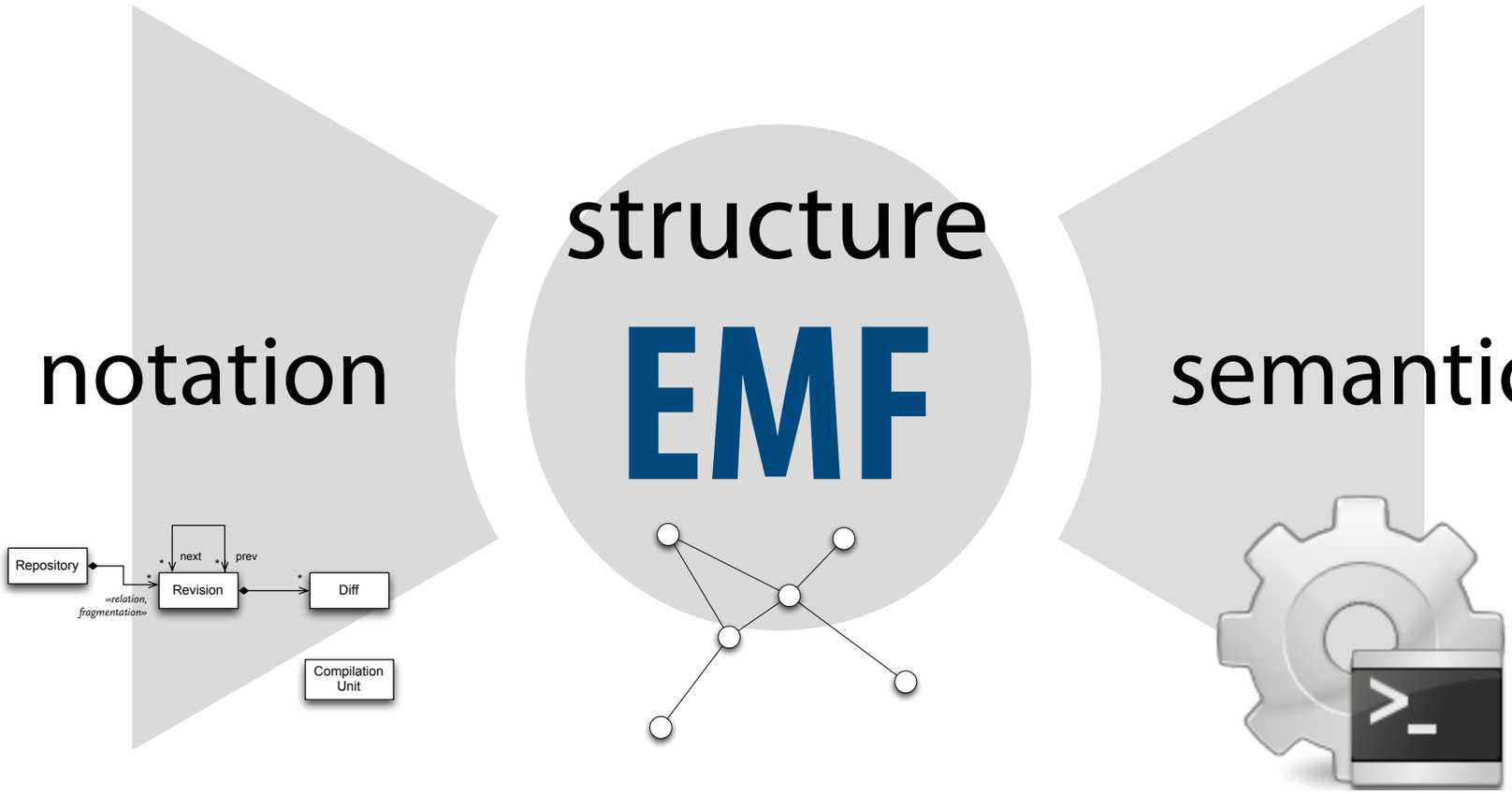
Meta-Languages



Concrete Meta-Languages



Eclipse Modeling Framework



Meta-Modeling

Meta-Modeling Definition

- There is a wide range of possible definitions...
  - a meta-model defines a set of models
  - a meta-model is a structural model that defines a set of structures
  - what a structure is depends on the meta-modeling formalism (i.e. the meta-meta-model): graphs, trees, terms, algebras, etc.

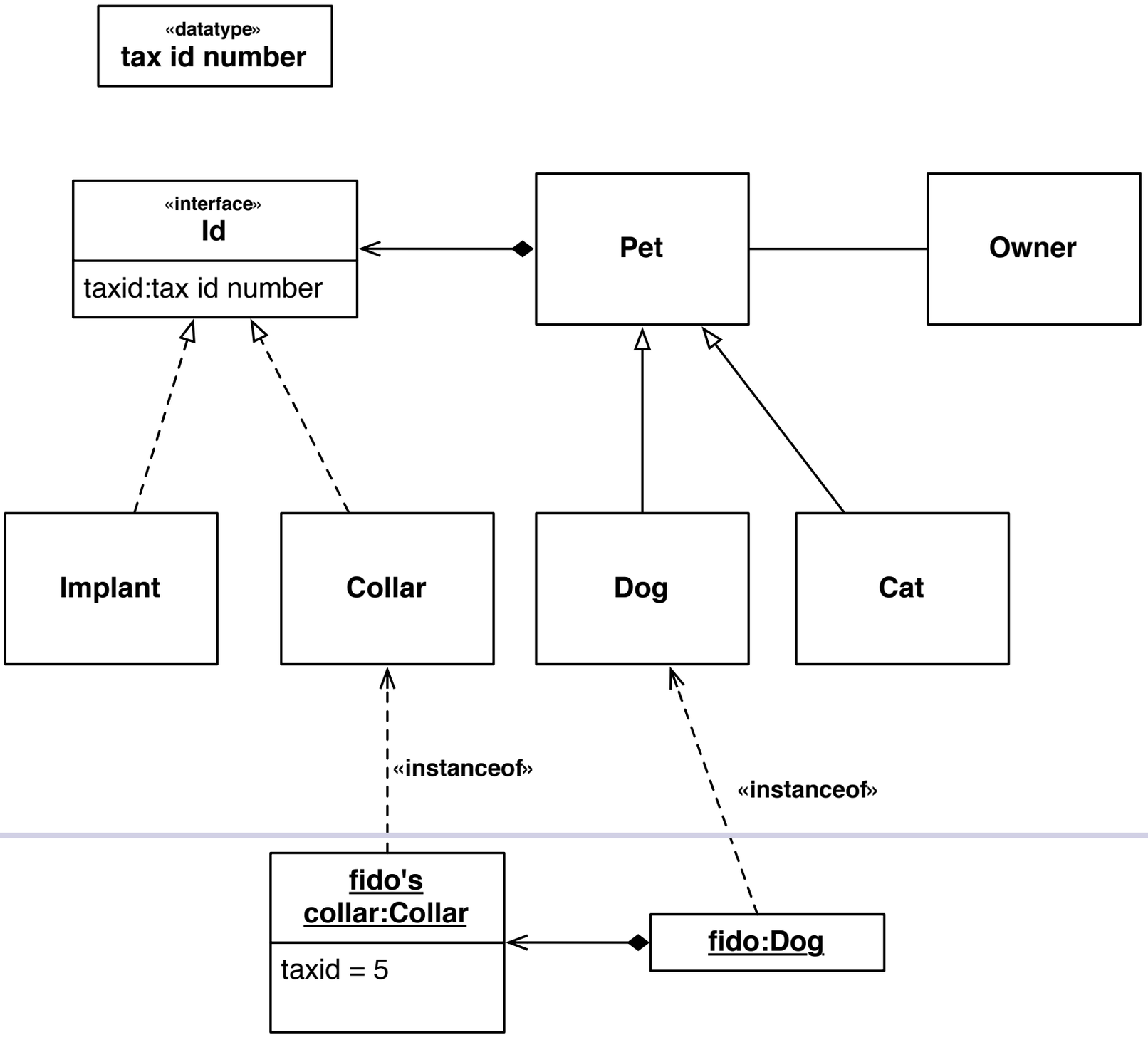
Examples for Meta-Modeling Formalisms

- Object oriented meta-modeling (e.g. based on MOF)
- Chompsky hierarchy of grammars/languages
- Formal automata
- XML
- Relational databases

Structure Modeling

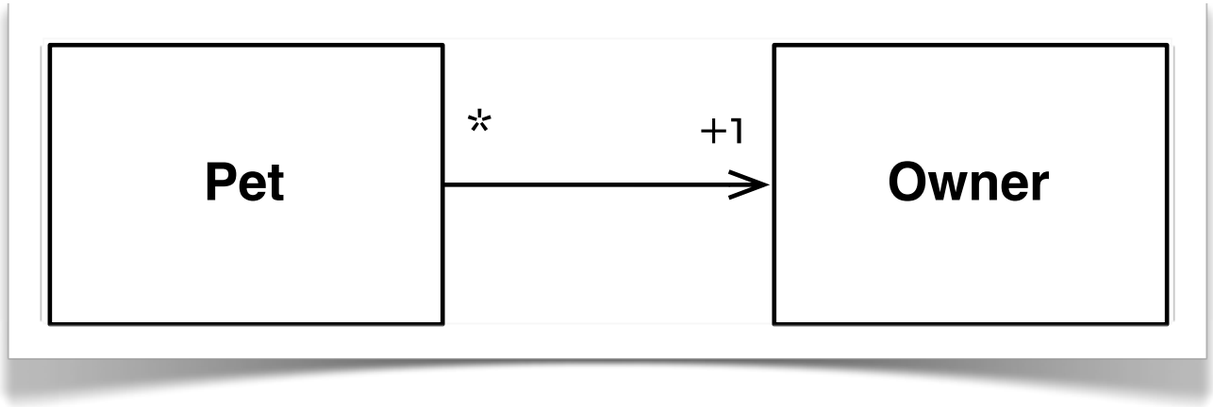
- Entities
  - Classifier, Classes, Datatypes
  - Objects, Values
- Relations
  - Association (incl. aggregation, composition)
  - Generalization / Specialization
  - Declaration / Implementation (Realization)
  - Classification / Instantiation
  - ...

Structure Modeling



Association

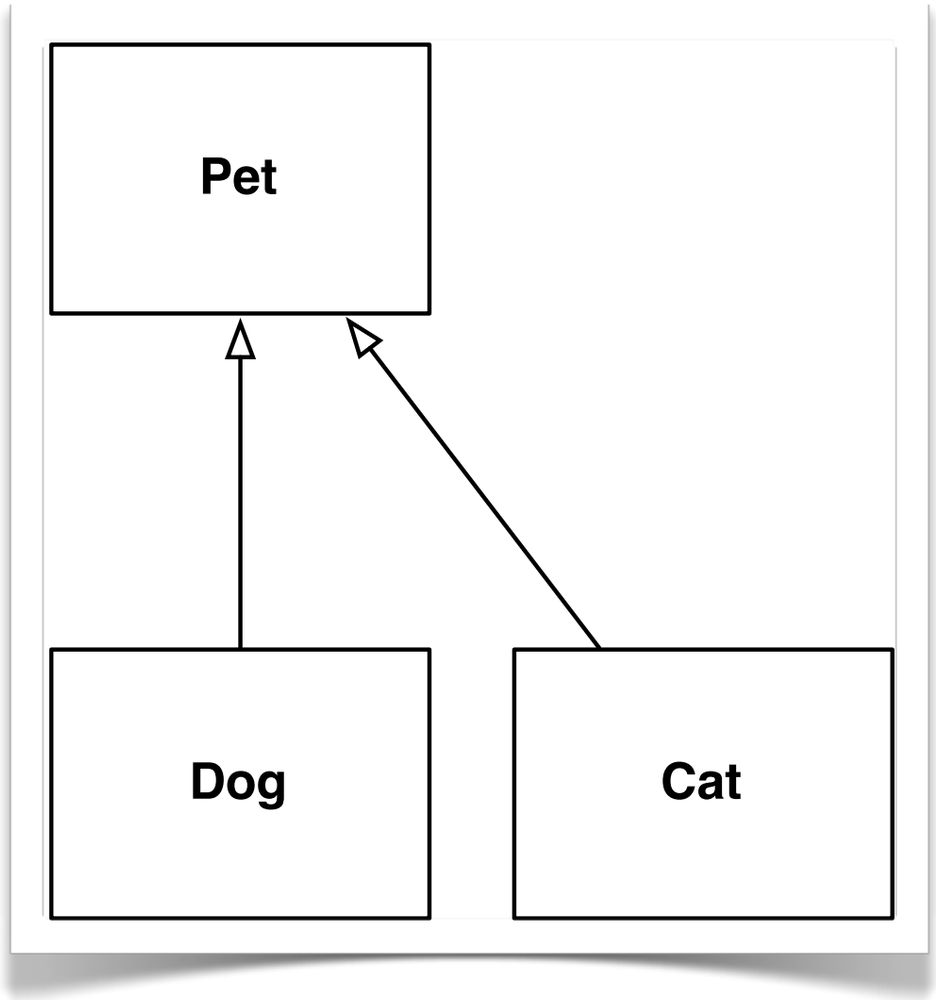
- between two entities on the same level of abstraction
- aggregation and composition as special kinds of association
- many properties
  - navigation
  - multiplicities
  - class
  - visibility



- can be seen as aggregation of one or two classifier features, called association ends or references

Generalization / Specialization

- Relates two entities of the same kind on different levels of abstraction but on the same meta-layer
- only reasonable for classifiers
- Classifiers can be defined extrinsically or intrinsically.
- Specialization means less elements, stricter rules.

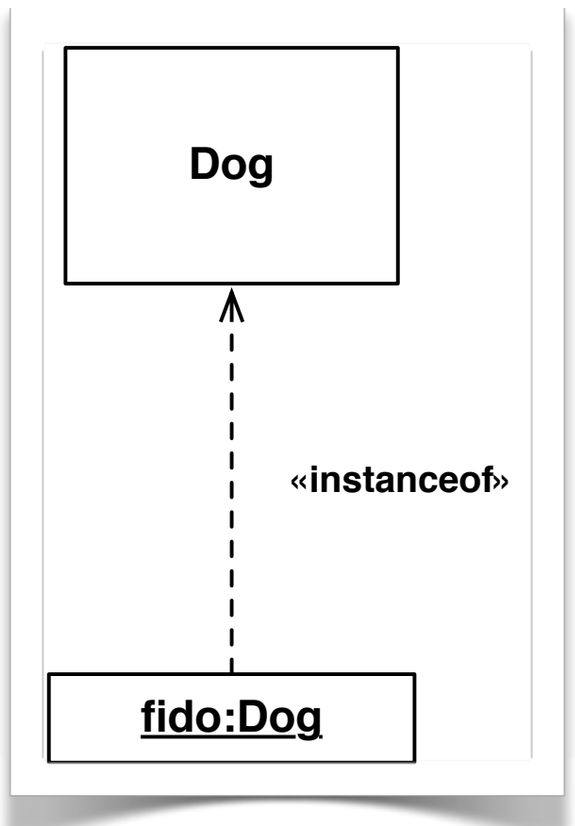


Declaration / Implementation (Realization)

- Relates two entities of the different kind on different levels of abstraction but on the same meta-layer
- Only important in the context of behavior/semantic modeling

Classification / Instantiation

- Relates two entities of different kind on different levels of abstraction and on different meta-layer.
- Typical entity kind pairs are: class/object, datatype/value, feature/property, etc.
- Class and datatype are also generalized under the term classifier.
- defines meta-layers
- the essential meta-modeling relation
- extends to comprising features



Instantiation: Class / Object

- most important instance of instantiation in meta-modeling
- A class constitutes an intrinsic definition for a set of objects
  - name, package
  - defines structural features (a.k.a contained attributes and corresponding association ends)
  - inherits structural features
- An object instantiates a class (or classes) and its (their) structural features.
- Objects have id semantics: two objects with the same structure (same attribute values and same liked objects) can still be two distinct objects.

Instantiation: Datatype / Value

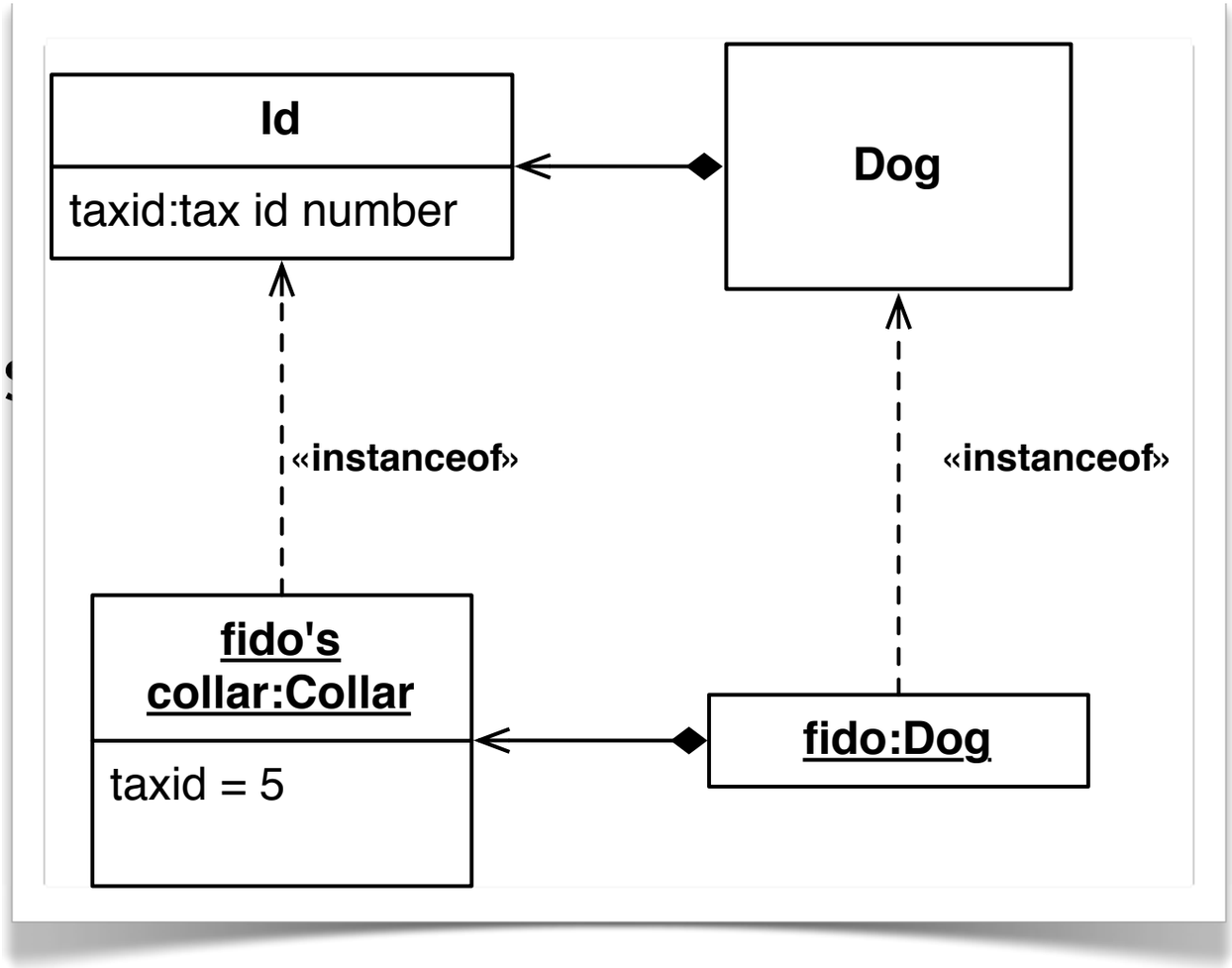
- Datatype defines a set of values. The type of definition depends on meta-language, platform, etc. Also known as abstract type definition.
- Values have value semantics: two values that look the same are the same.

Instantiation: Attribute, Reference/ Property, Link

- Instantiation extends to class features.
- A structural feature defines
  - name
  - type (usually a classifier)
  - multiplicity, visibility
- A structural feature is instantiated through sets (multiplicity!) of properties and links (i.e. value sets)

Instantiation: Attribute, Reference/ Property, Link

- Instantiation extends to class
- A structural feature defines
  - name
  - type (usually a classifier)
  - multiplicity, visibility

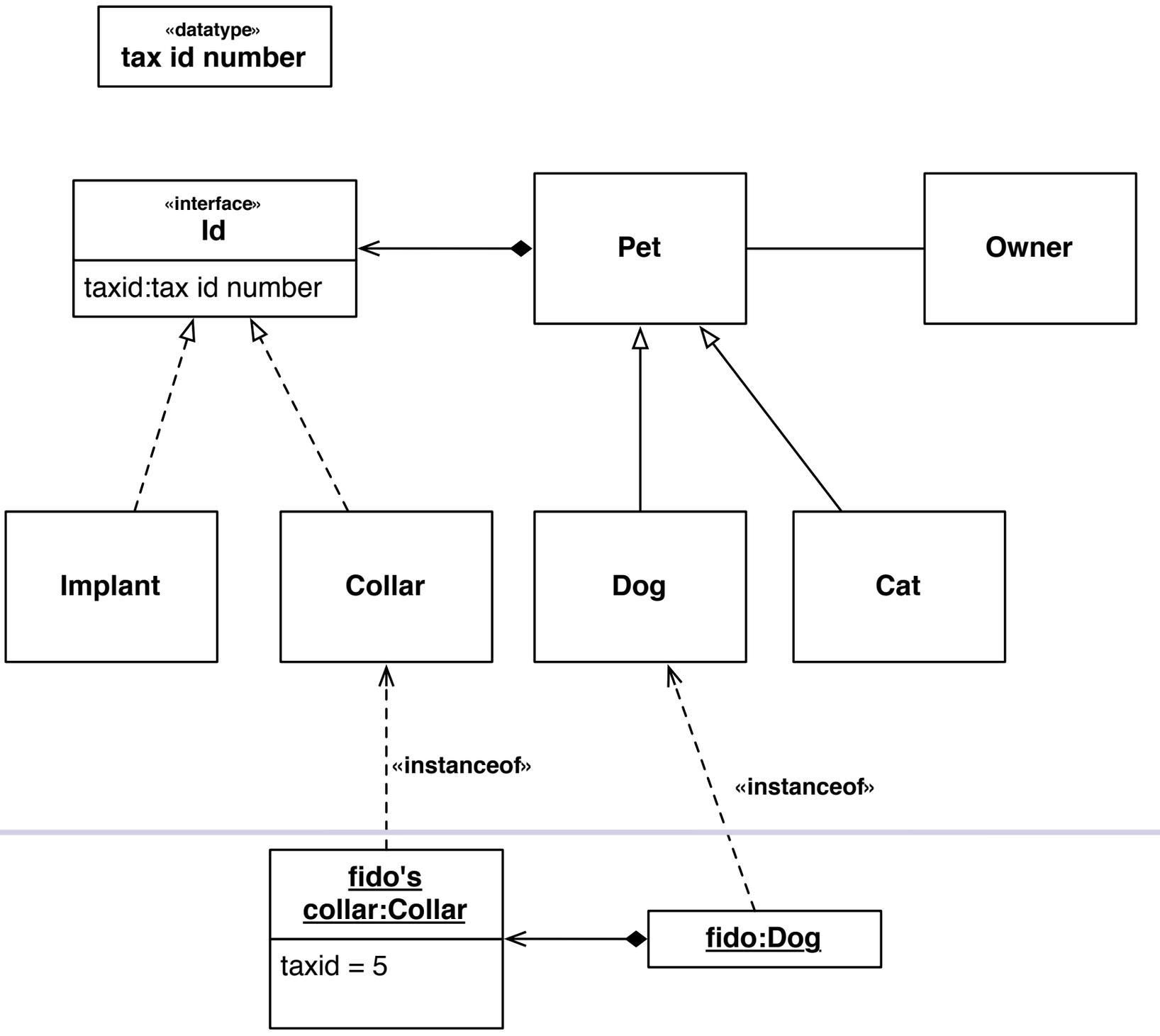


- A structural feature is instantiated through sets (multiplicity!) of properties and links (i.e. value sets)

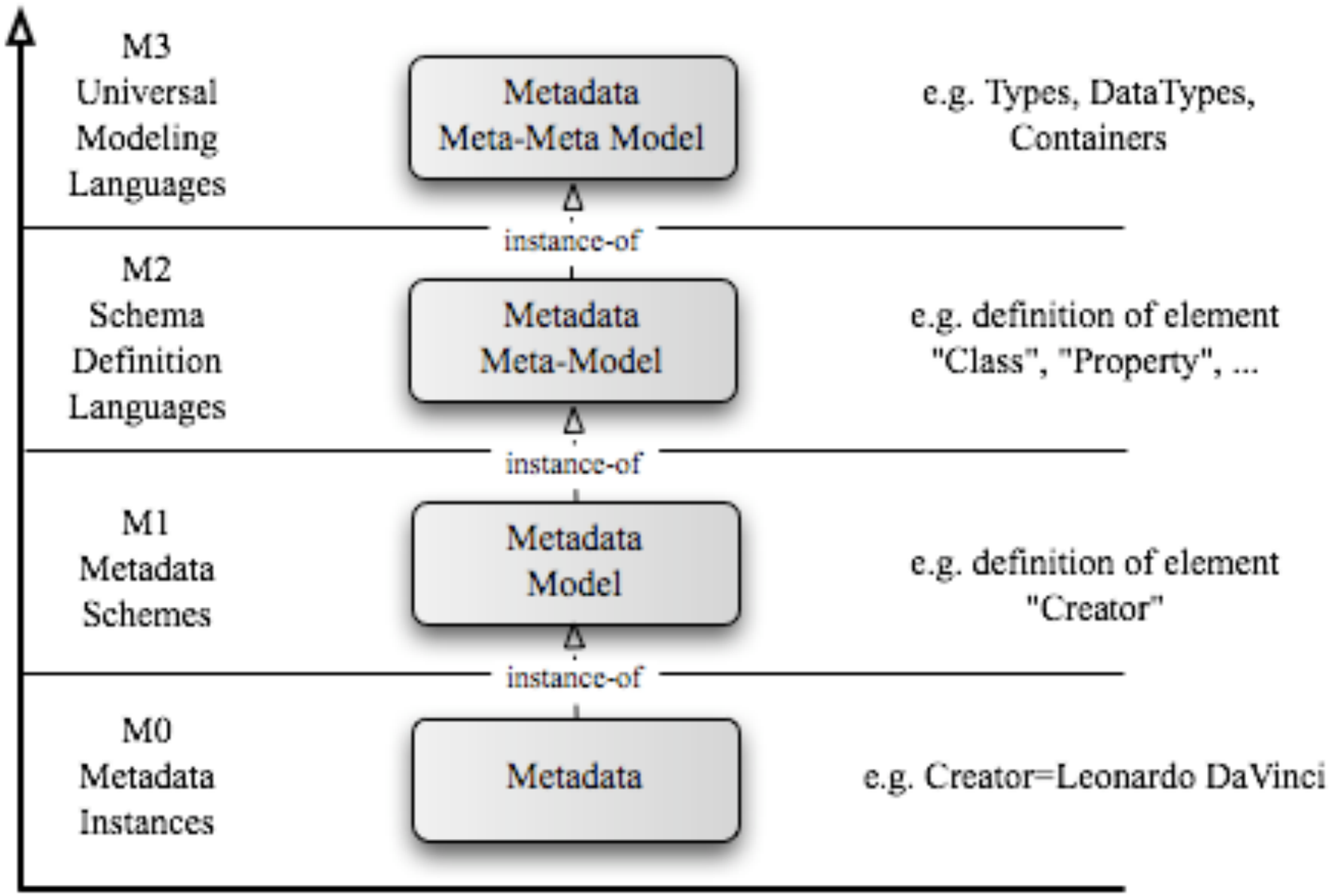
Multi-Level Meta-Modeling

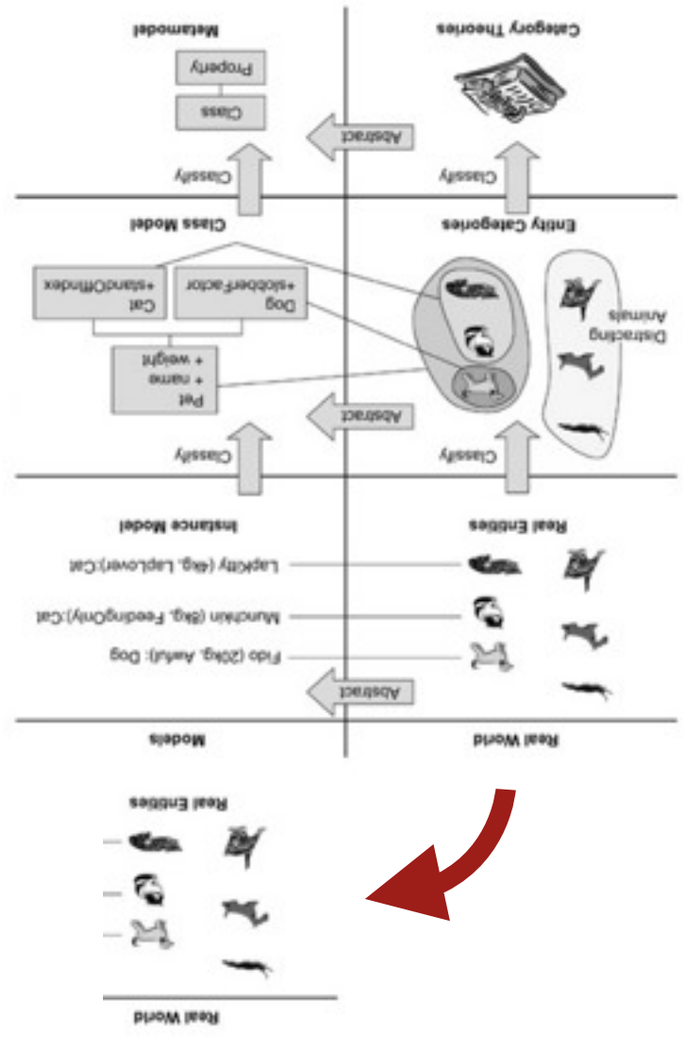
- Colin Atkinson, Thomas Kühne: *The Essence of Multi-Level Meta-Modeling*, UML 2001 (later MODELs)

Two Layer Object Oriented Structure Modeling



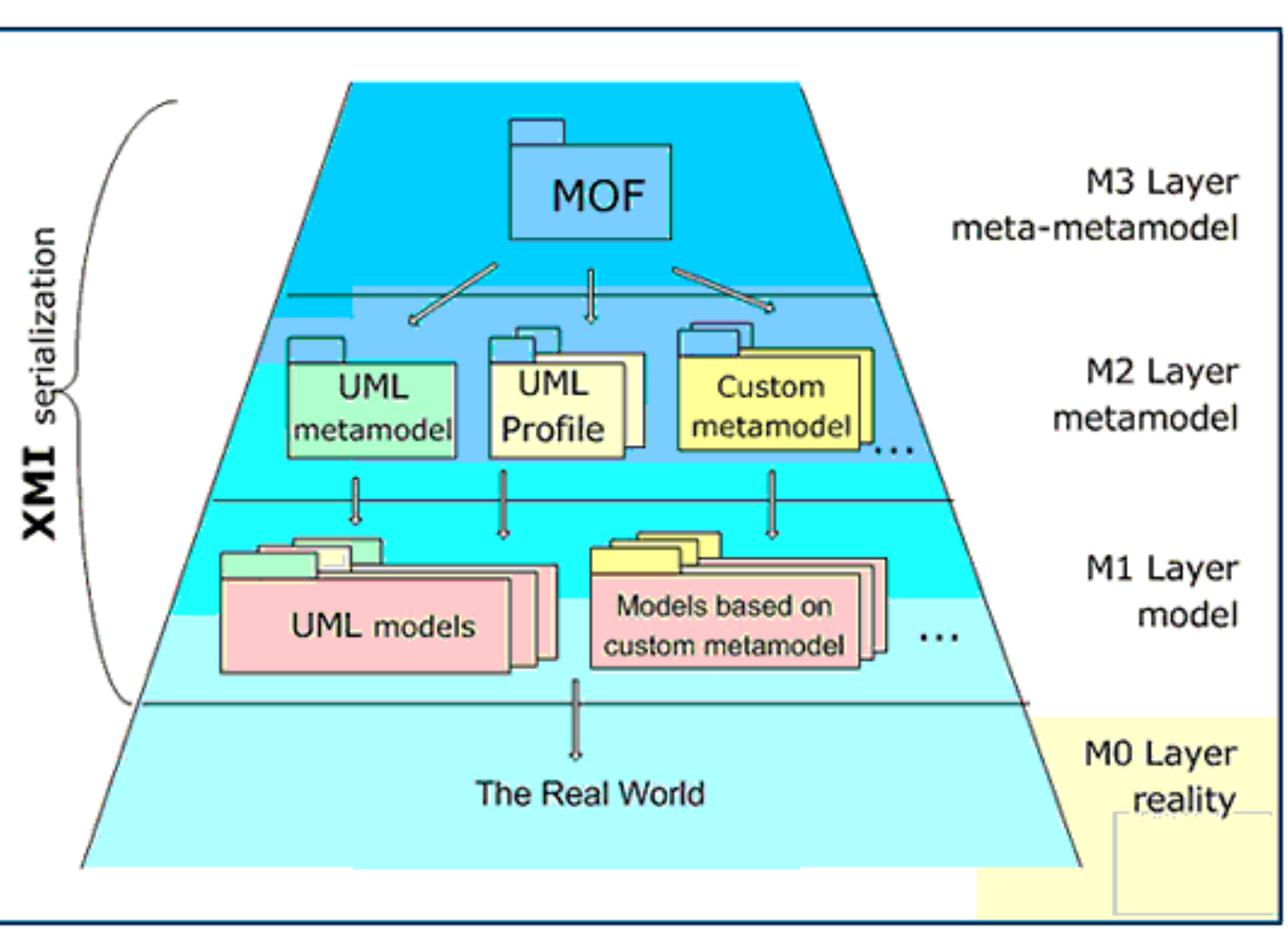
Meta-Modeling – M4 Model

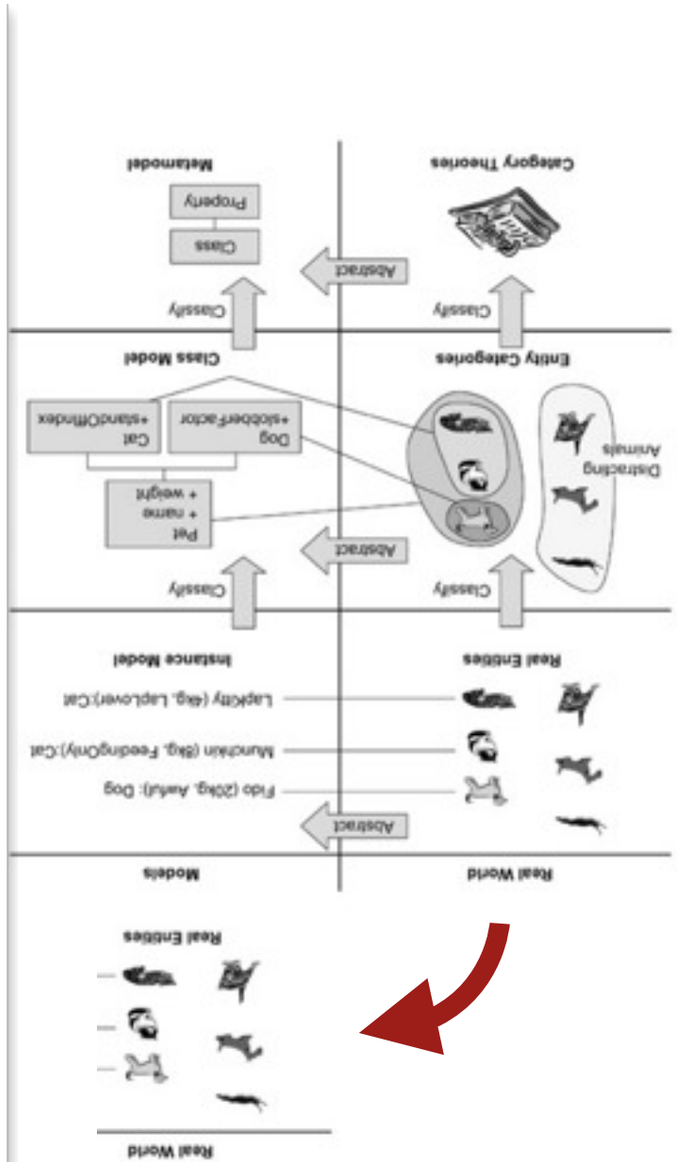




Object Management Group (OMG): *Meta Object Facility (MOF)*

Meta-Modeling – M4 Model





Object Management Group (OMG): *Meta Object Facility (MOF)*

Ambiguous Instantiation – “Dog” DSL

meta-meta-model (M3)

meta-model (M2)

model (M1)

real things (M0)

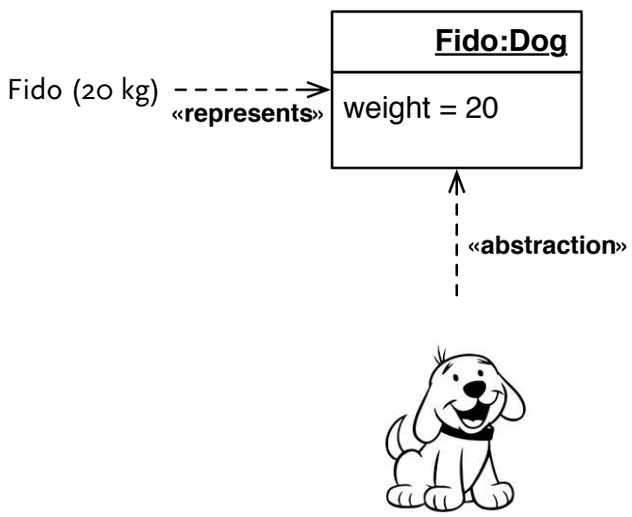
Ambiguous Instantiation – “Dog” DSL

meta-meta-model (M3)

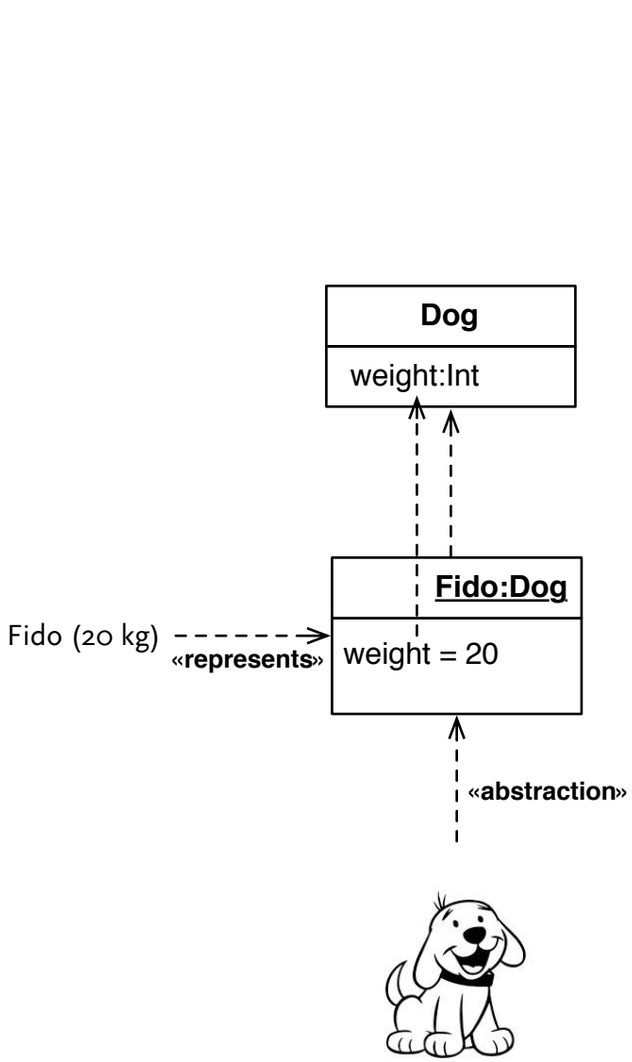
meta-model (M2)

model (M1)

real things (M0)



Ambiguous Instantiation – “Dog” DSL



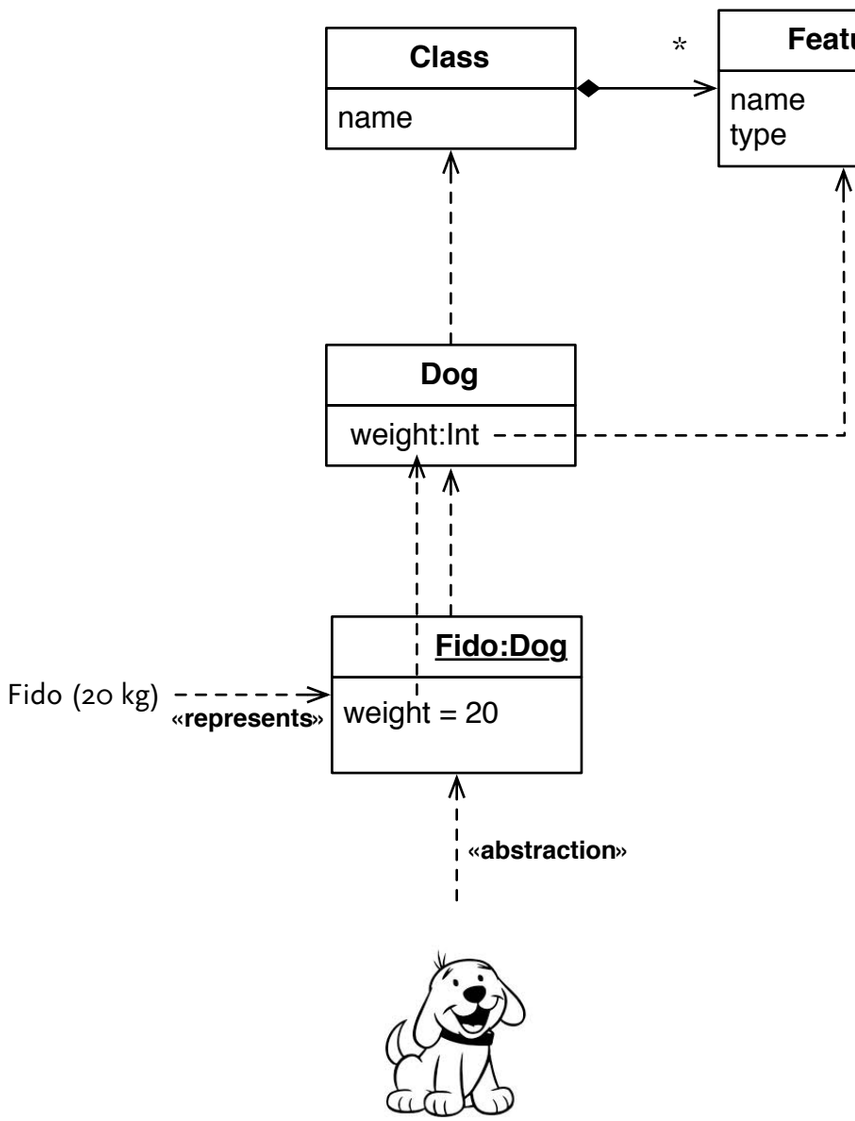
meta-meta-model (M3)

meta-model (M2)

model (M1)

real things (M0)

Ambiguous Instantiation – “Dog” DSL



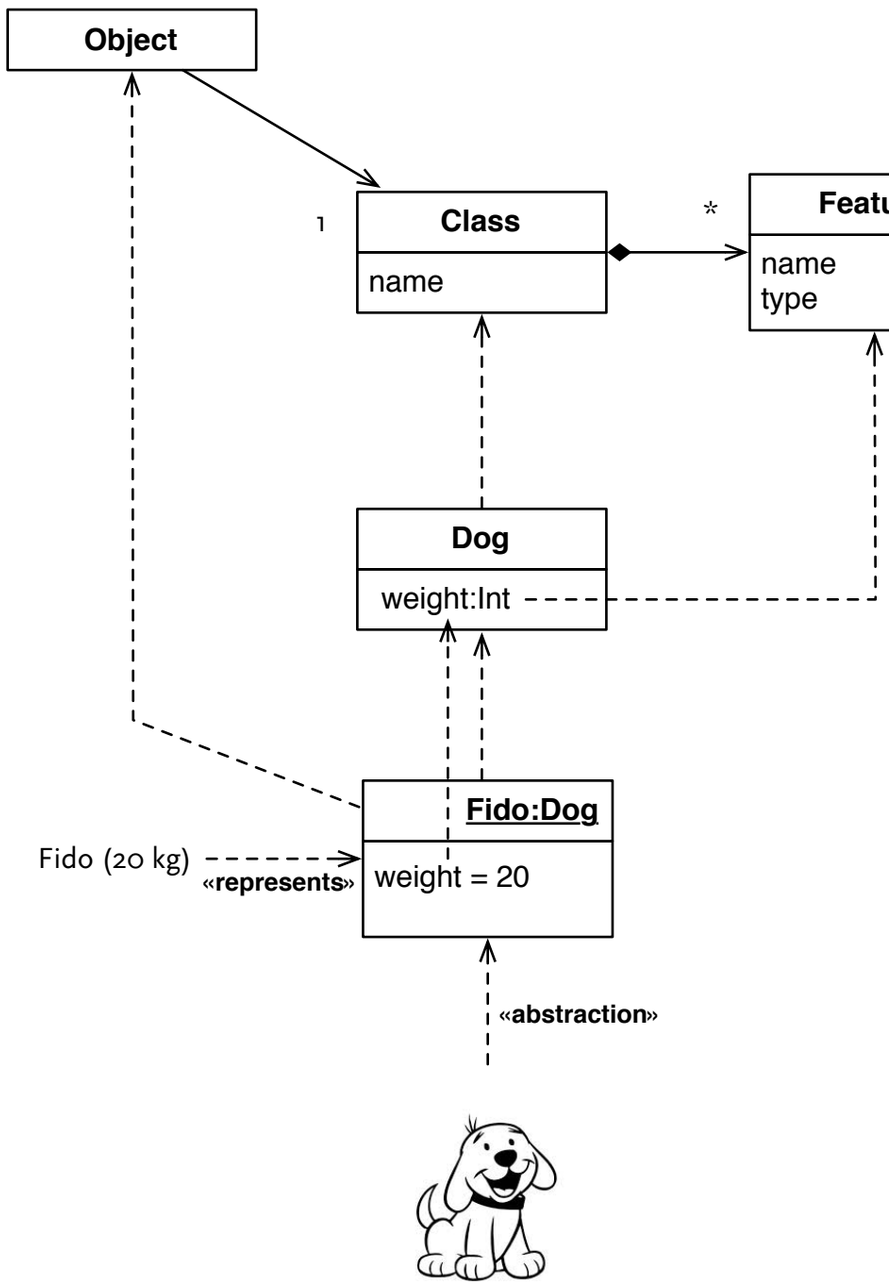
meta-meta-model (M3)

meta-model (M2)

model (M1)

real things (M0)

Ambiguous Instantiation – “Dog” DSL



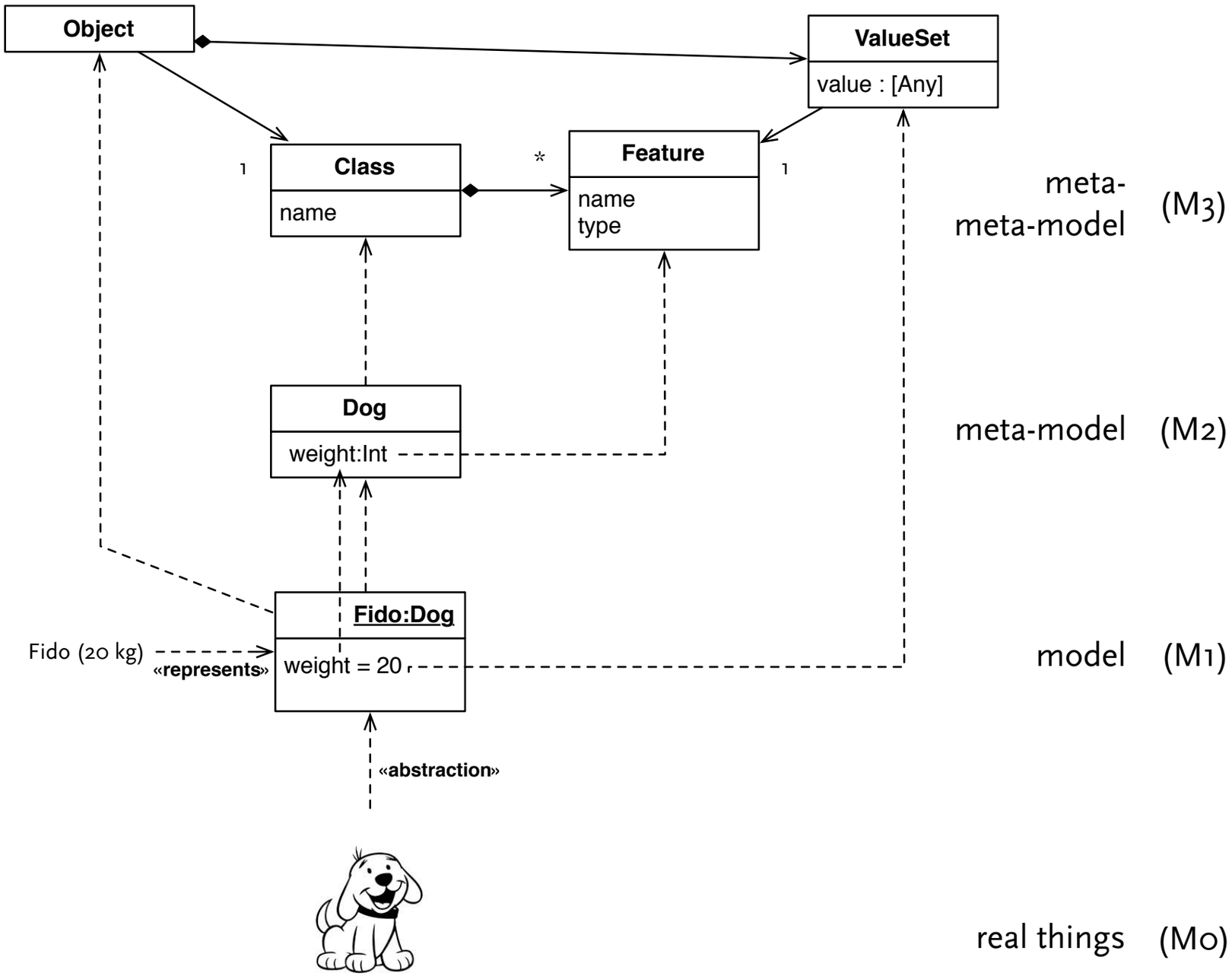
meta-meta-model (M3)

meta-model (M2)

model (M1)

real things (M0)

Ambiguous Instantiation – “Dog” DSL



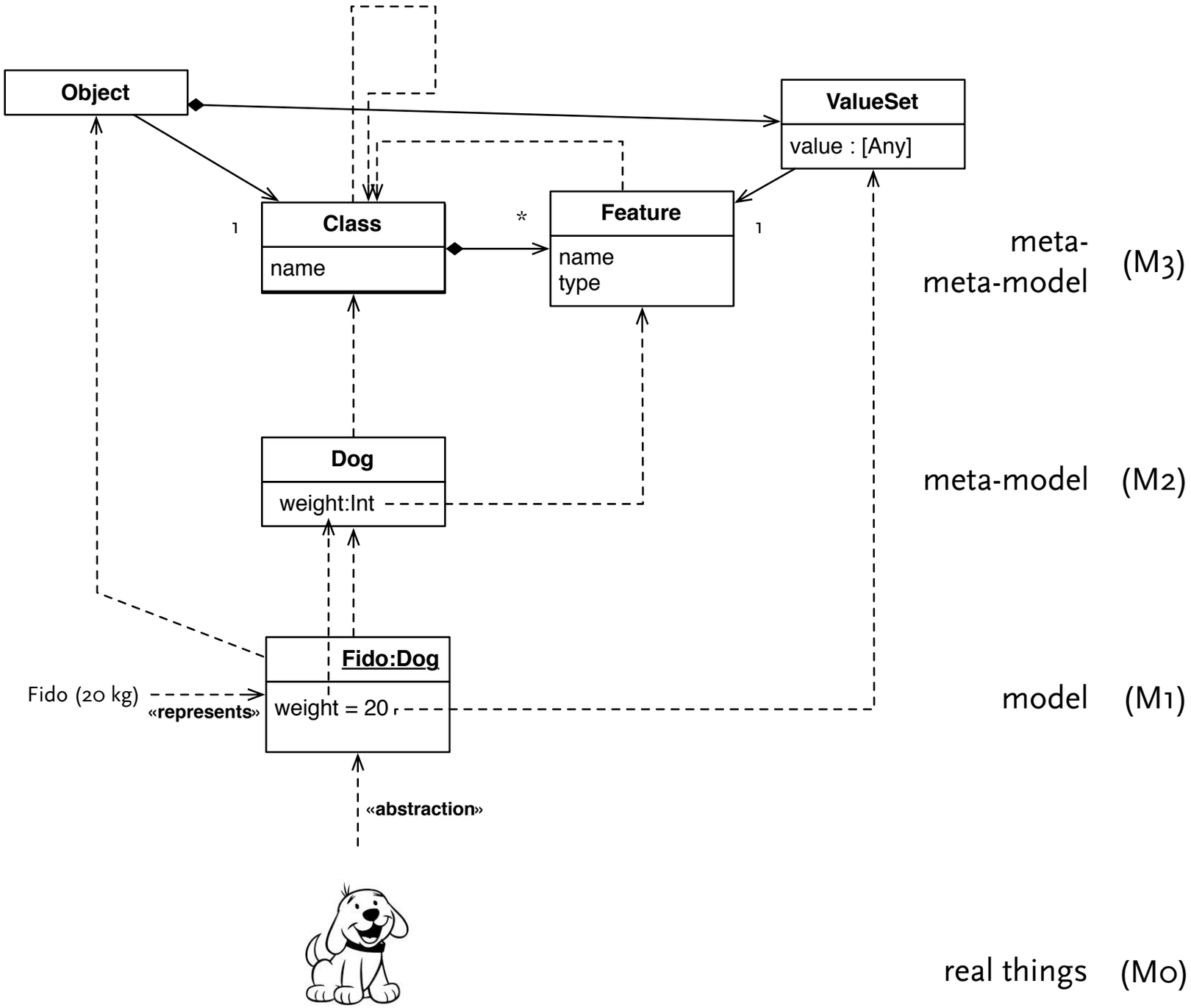
meta-meta-model (M3)

meta-model (M2)

model (M1)

real things (M0)

Ambiguous Instantiation – “Dog” DSL



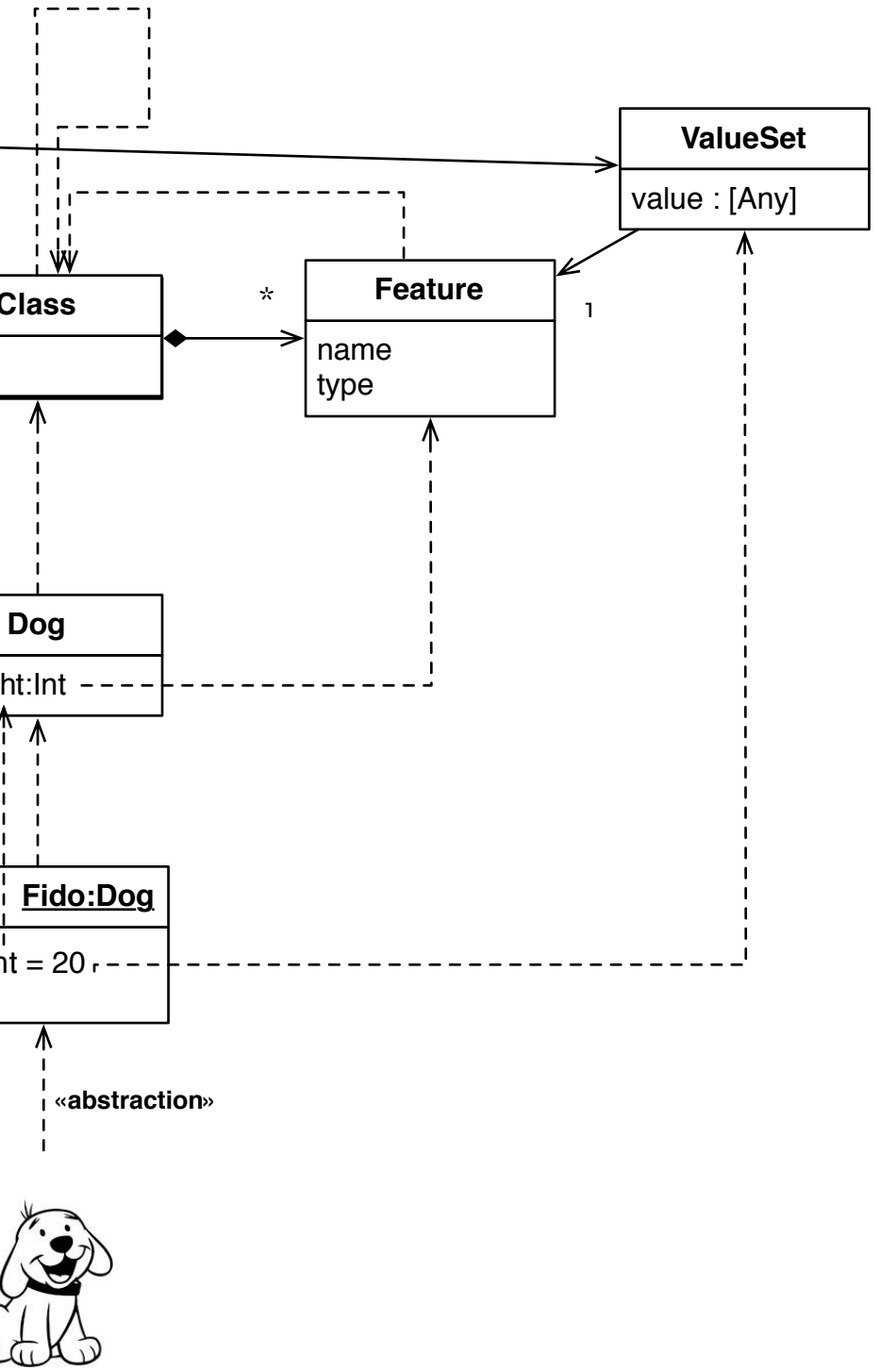
meta-meta-model (M3)

meta-model (M2)

model (M1)

real things (M0)

Replication of Concepts – “Dog” UML Model



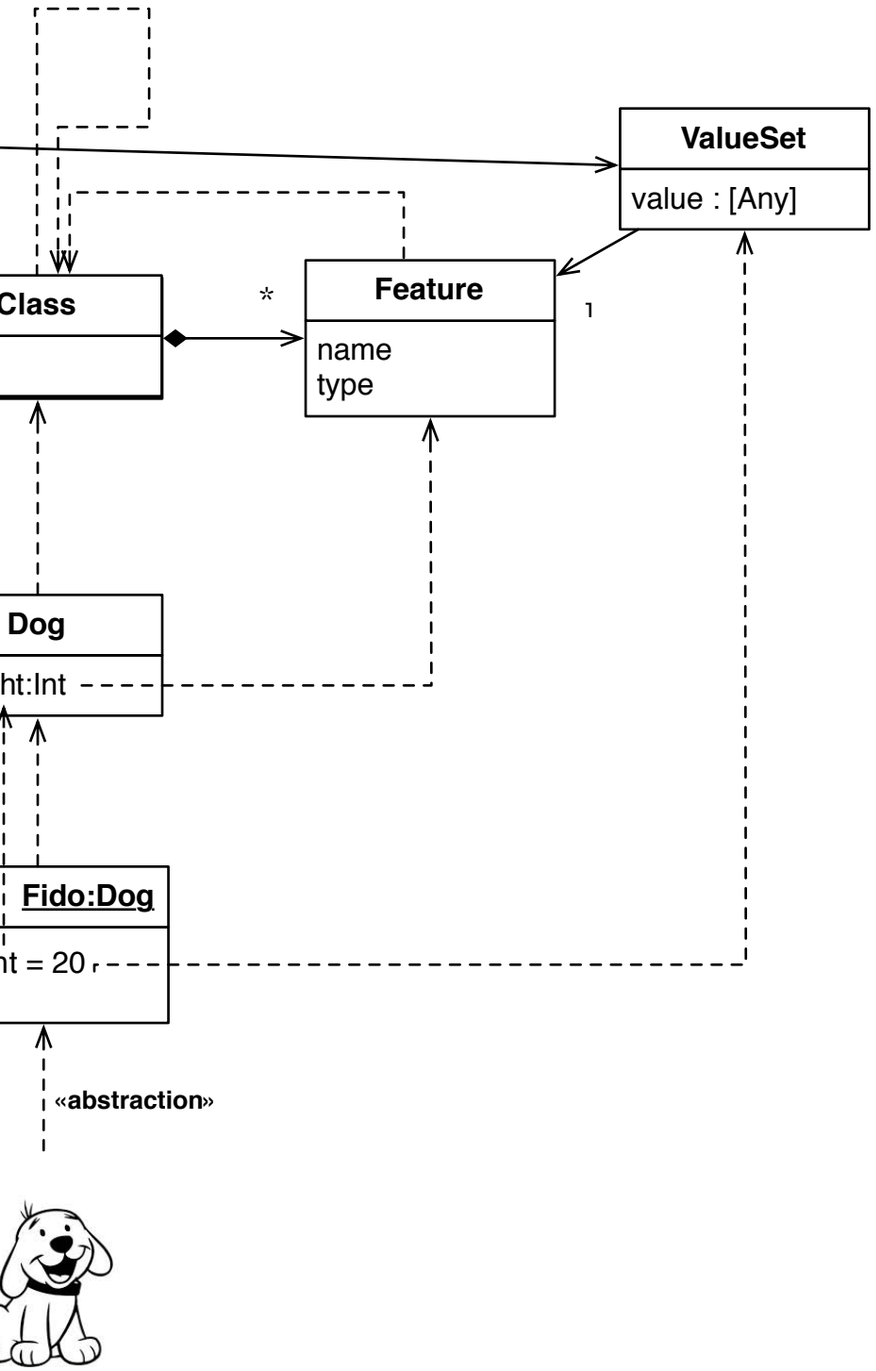
(M3) MOF

(M2) UML

(M1) UML model

(M0) real things

Replication of Concepts – “Dog” UML Model

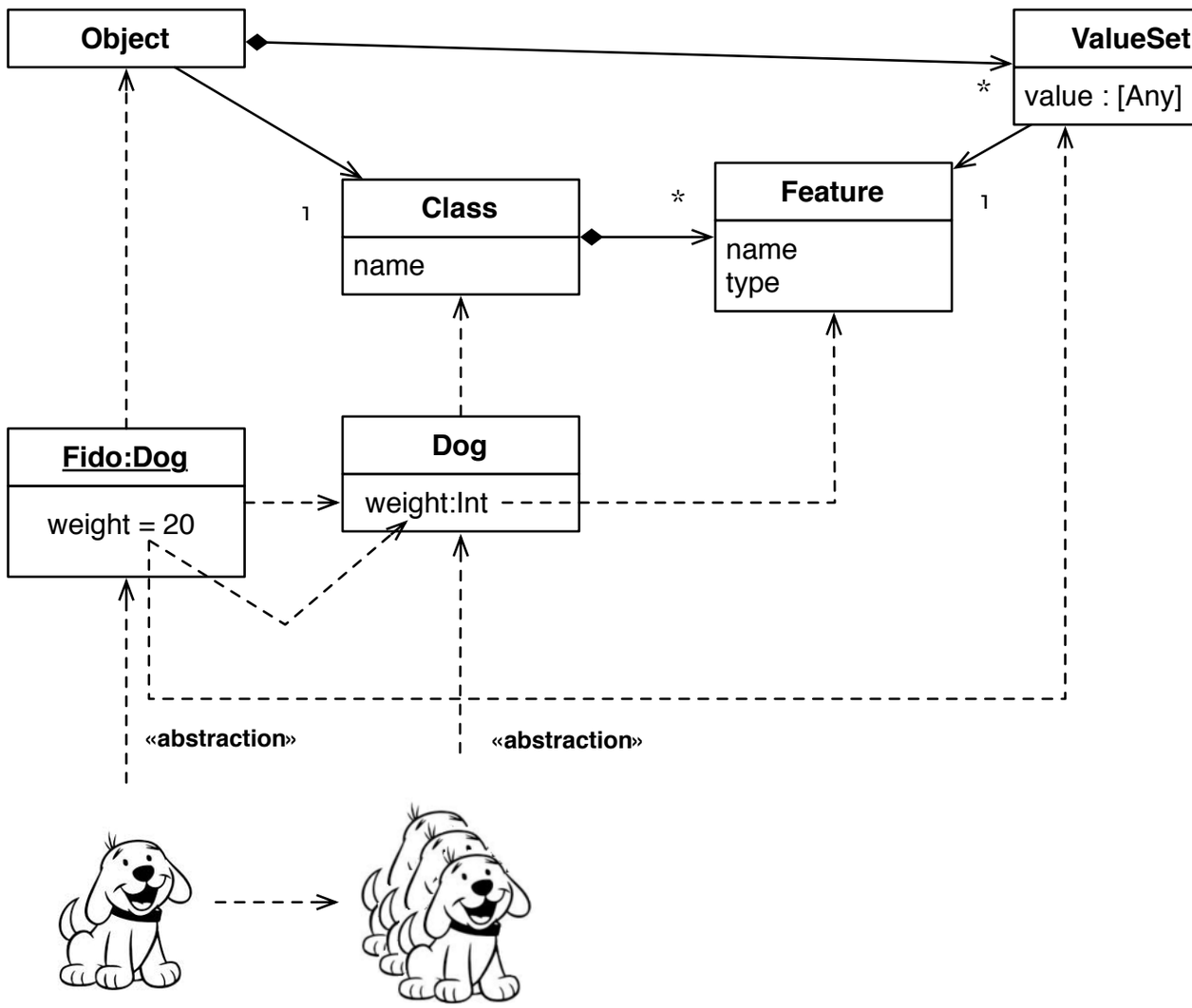


(M3) MOF

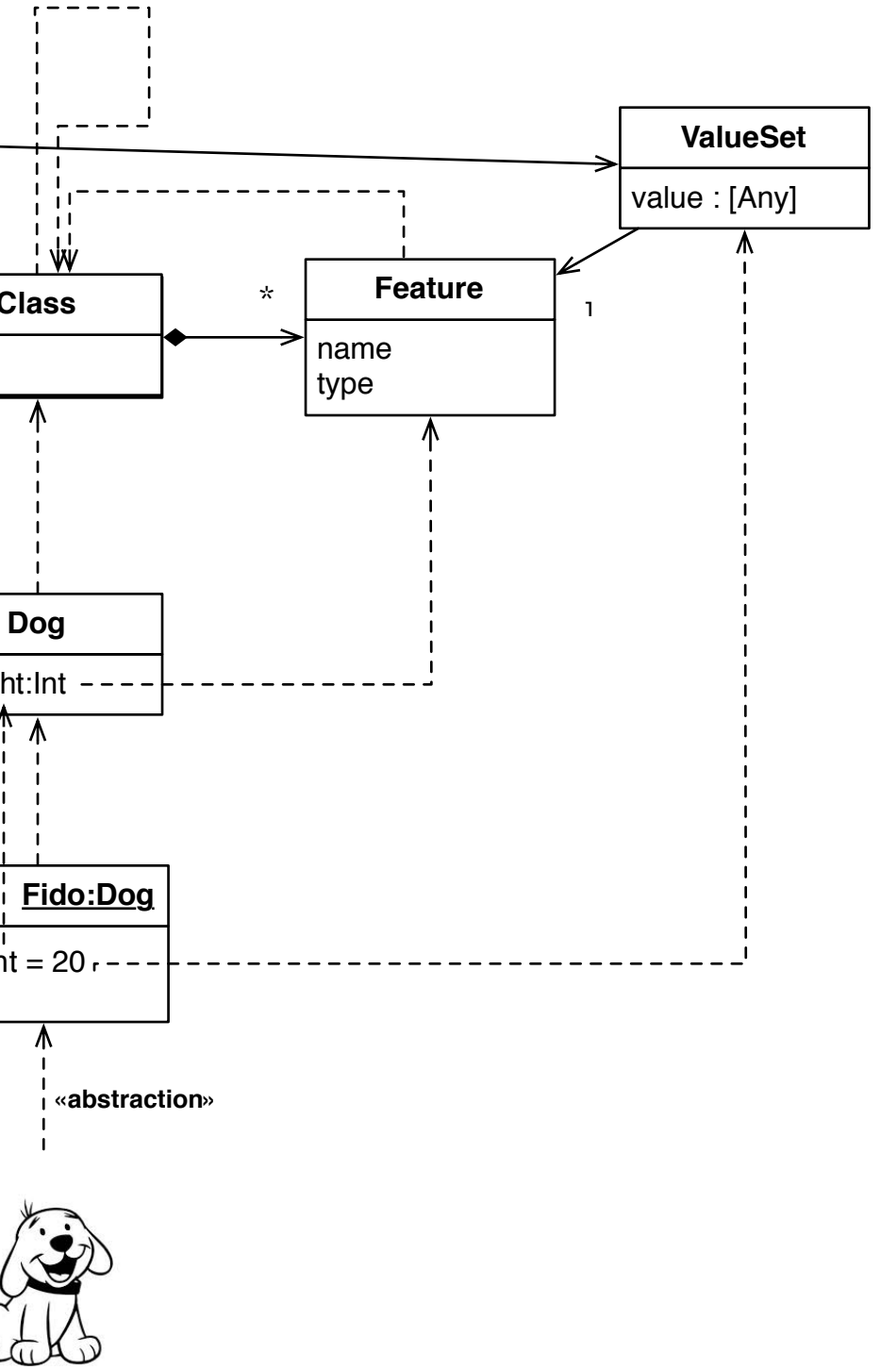
(M2) UML

(M1) UML model

(M0) real things



Replication of Concepts – “Dog” UML Model

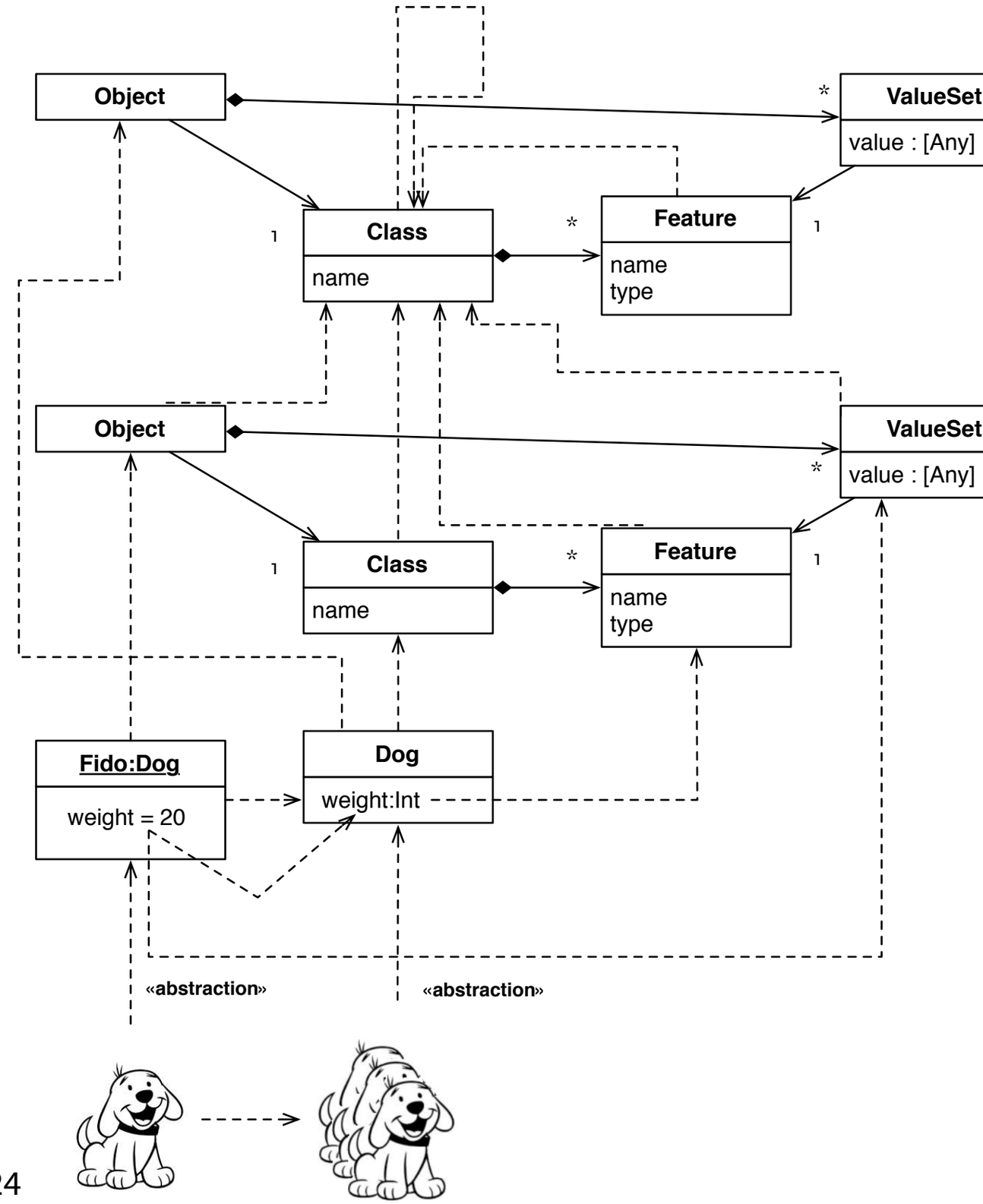


(M3) MOF

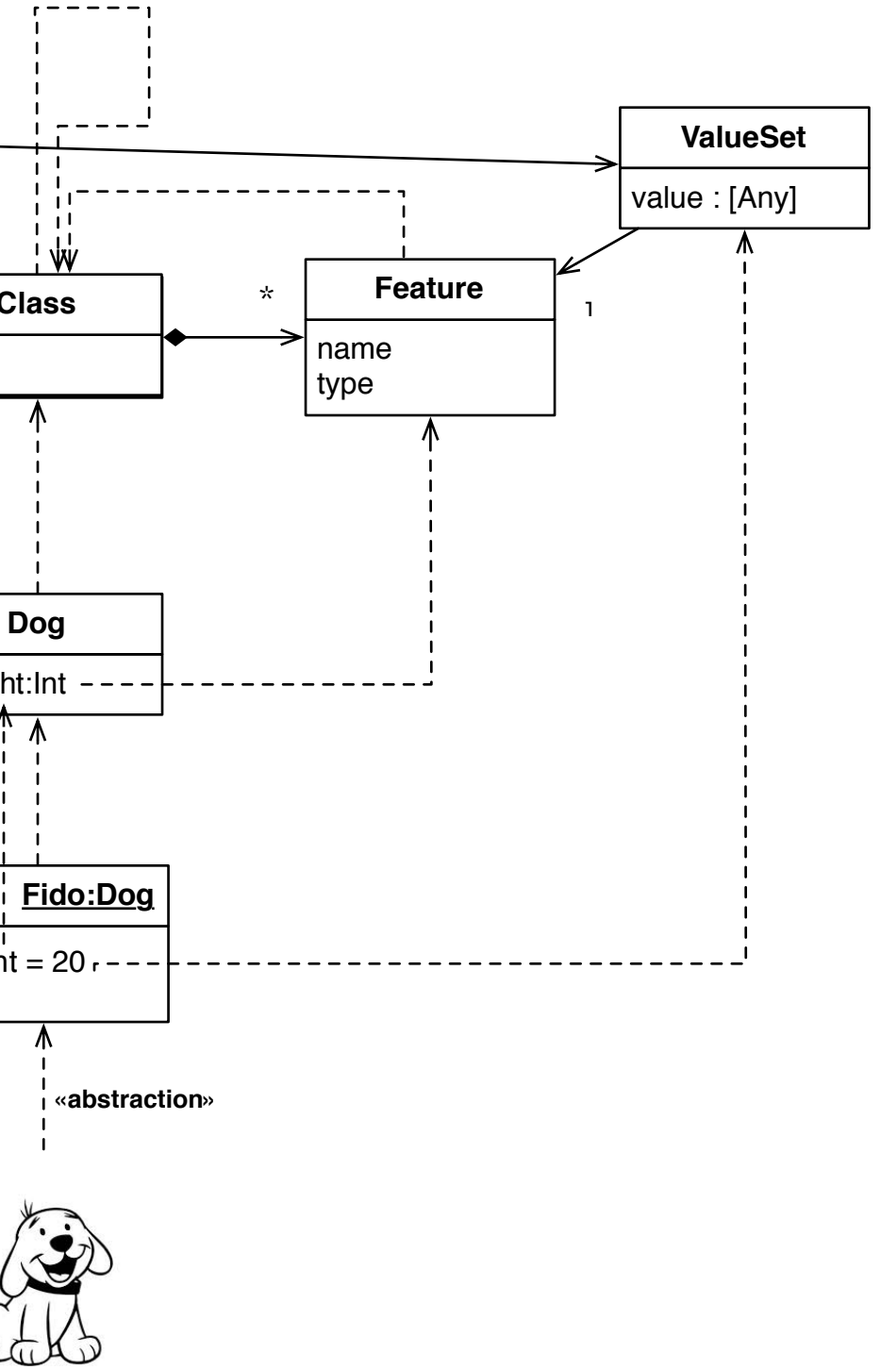
(M2) UML

(M1) UML model

(M0) real things



Replication of Concepts – “Dog” UML Model

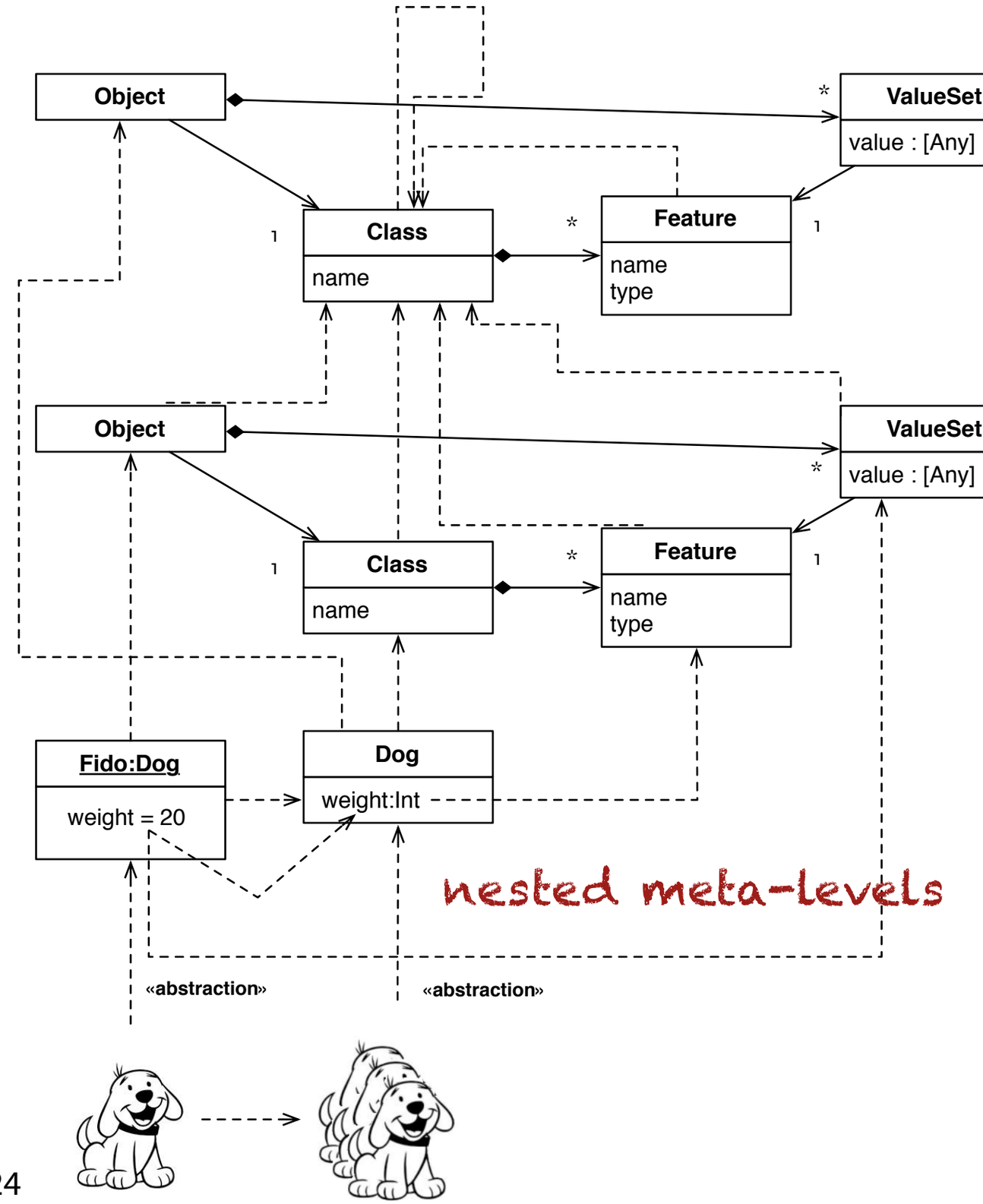


(M3) MOF

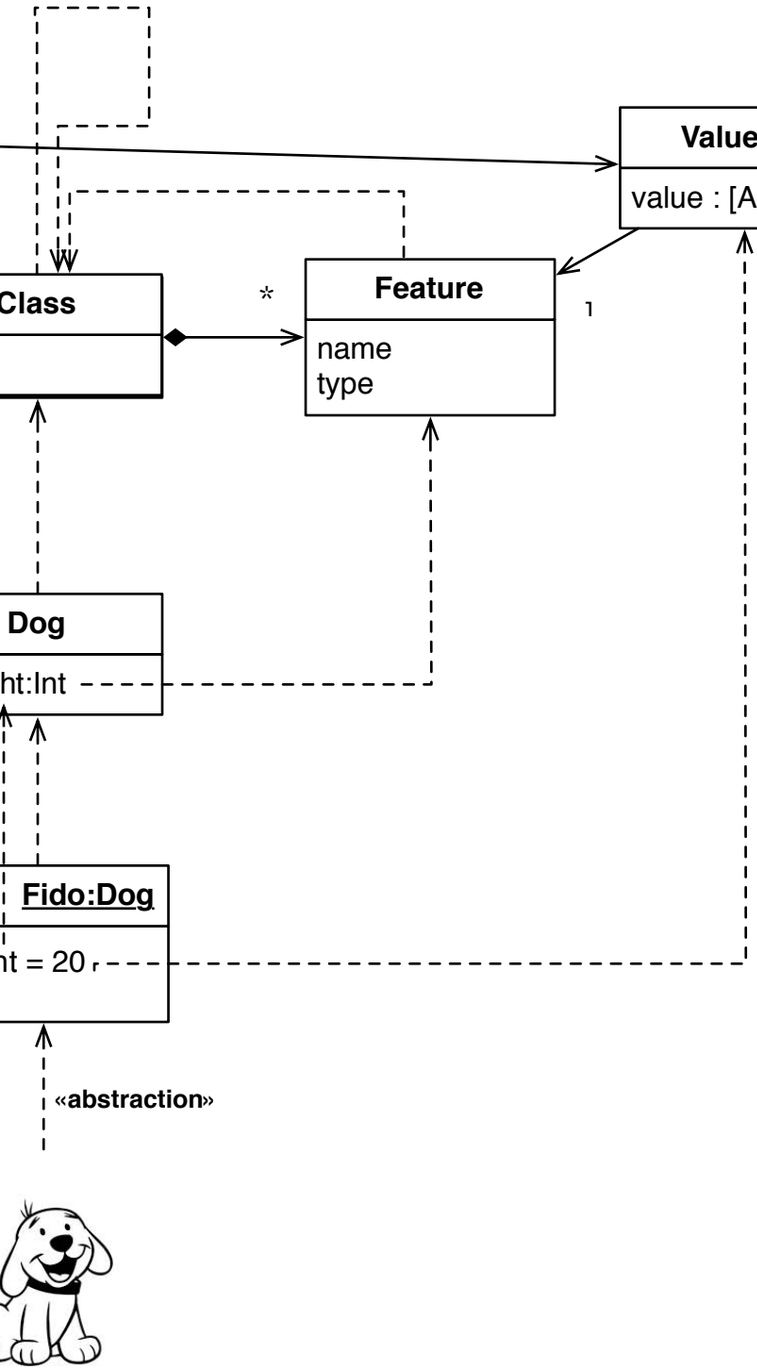
(M2) UML

(M1) UML model

(M0) real things



Another Example – “Dog”Java Programm



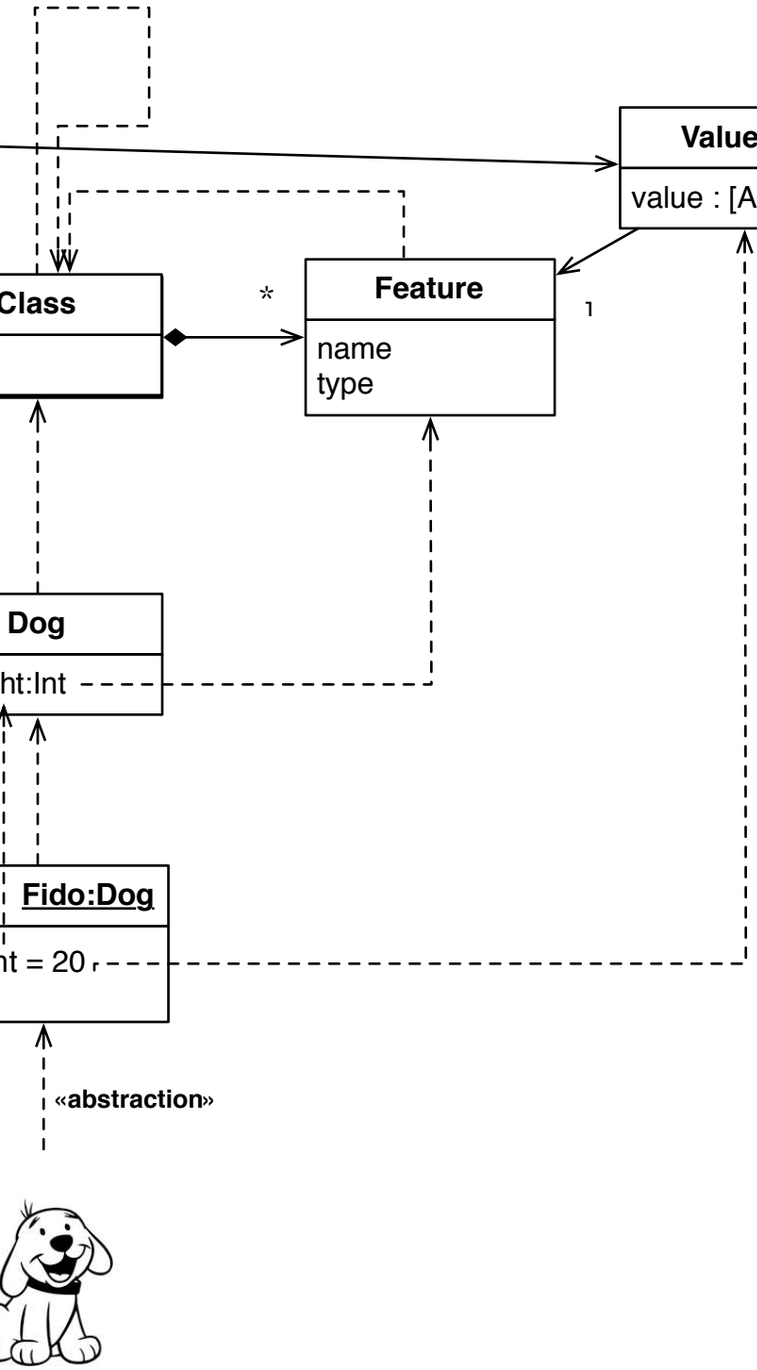
(M3) ?

(M2) Java Spec.

(M1) Java Prog.

(M0) real things / JVM

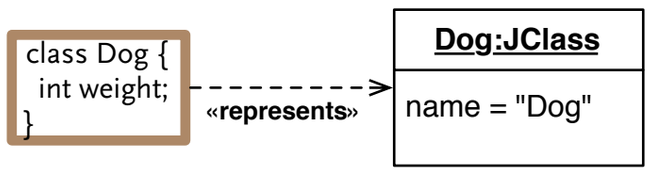
Another Example – “Dog”Java Programm



(M3) ?

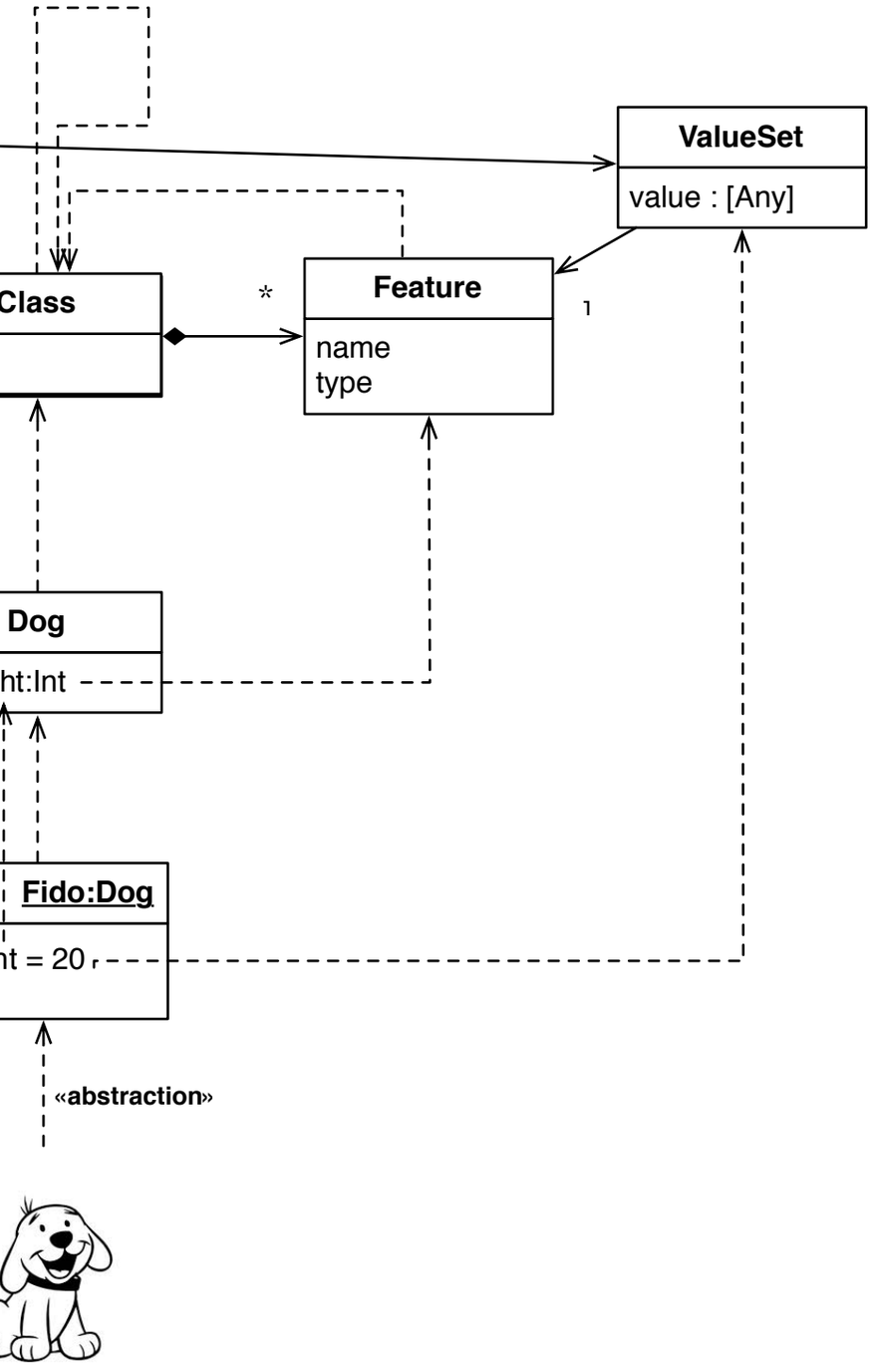
(M2) Java Spec.

(M1) Java Prog.



(M0) real things / JVM

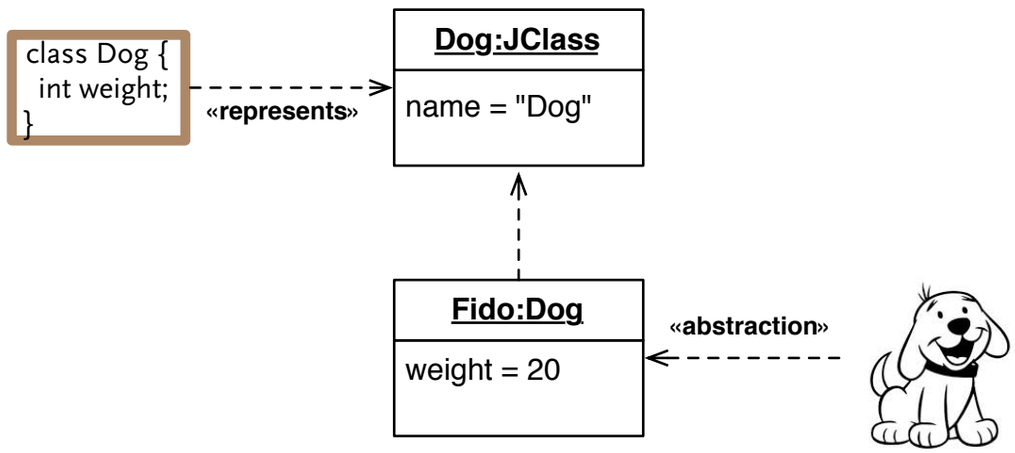
Another Example – “Dog”Java Programm



(M3) ?

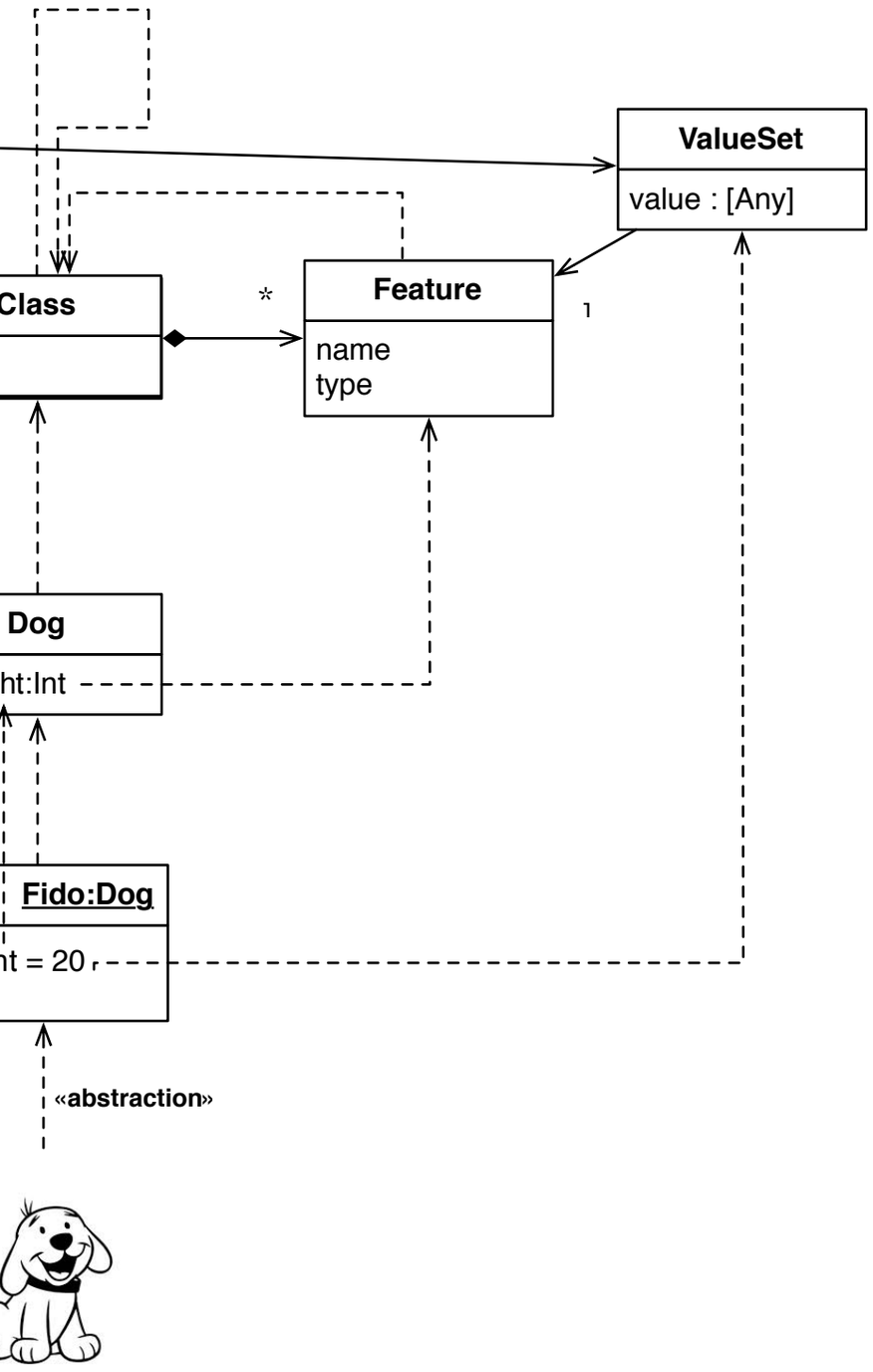
(M2) Java Spec.

(M1) Java Prog.



(M0) real things / JVM

Another Example – “Dog”Java Programm

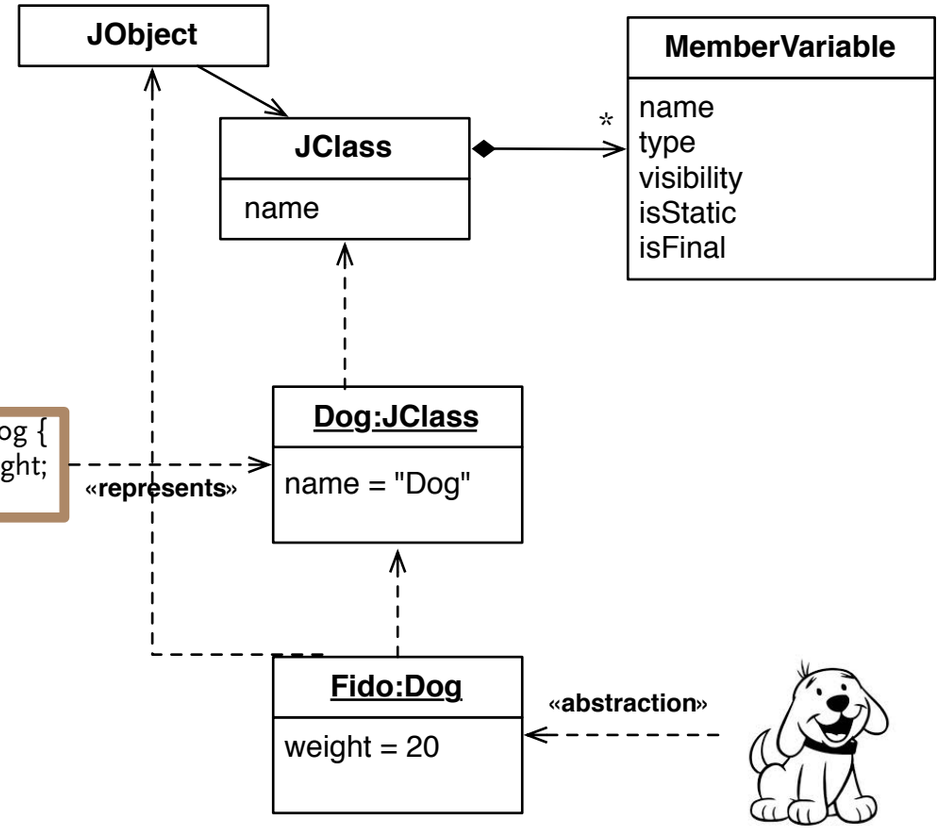


(M3) ?

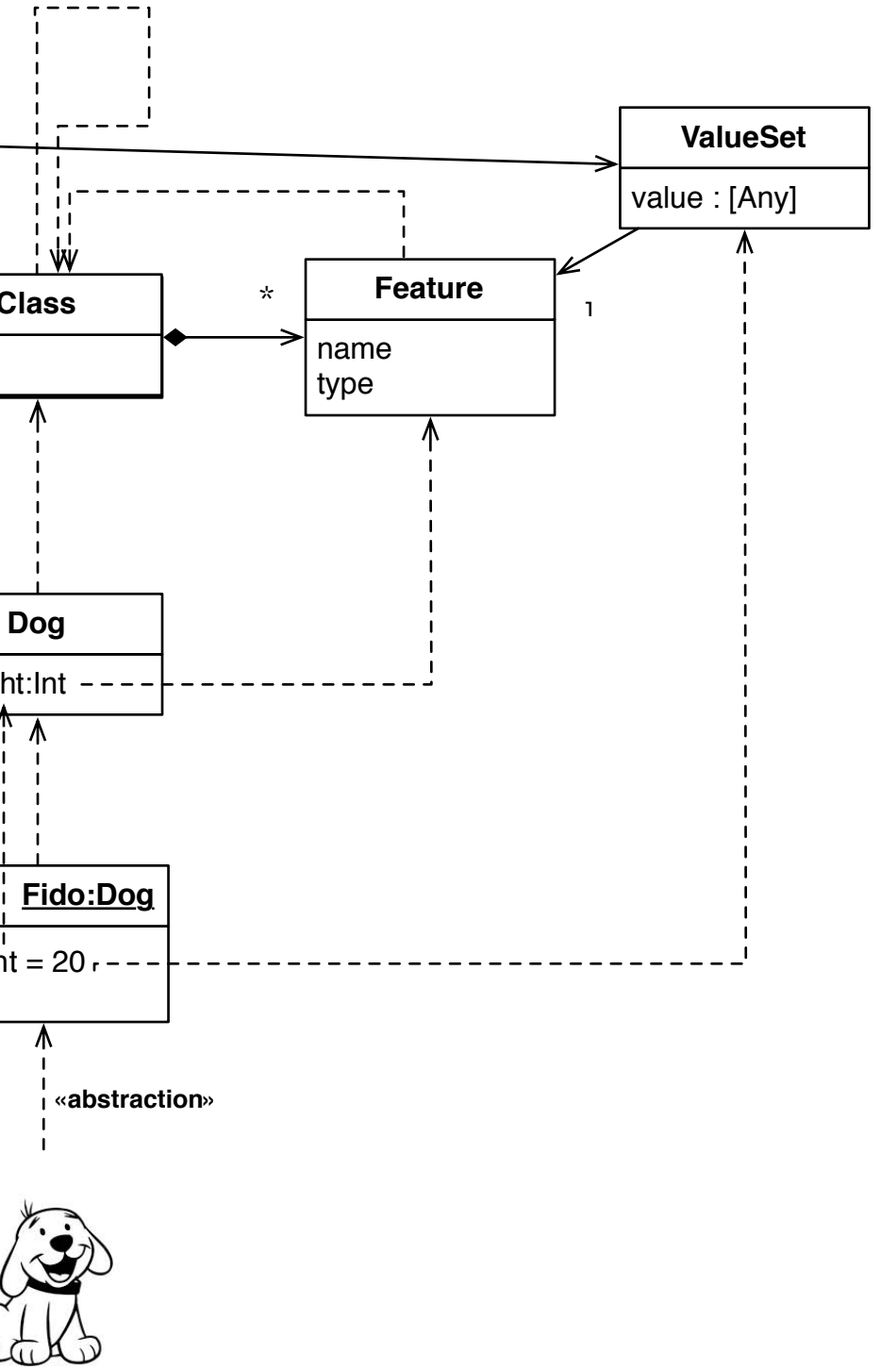
(M2) Java Spec.

(M1) Java Prog.

(M0) real things / JVM



Another Example – “Dog”Java Programm

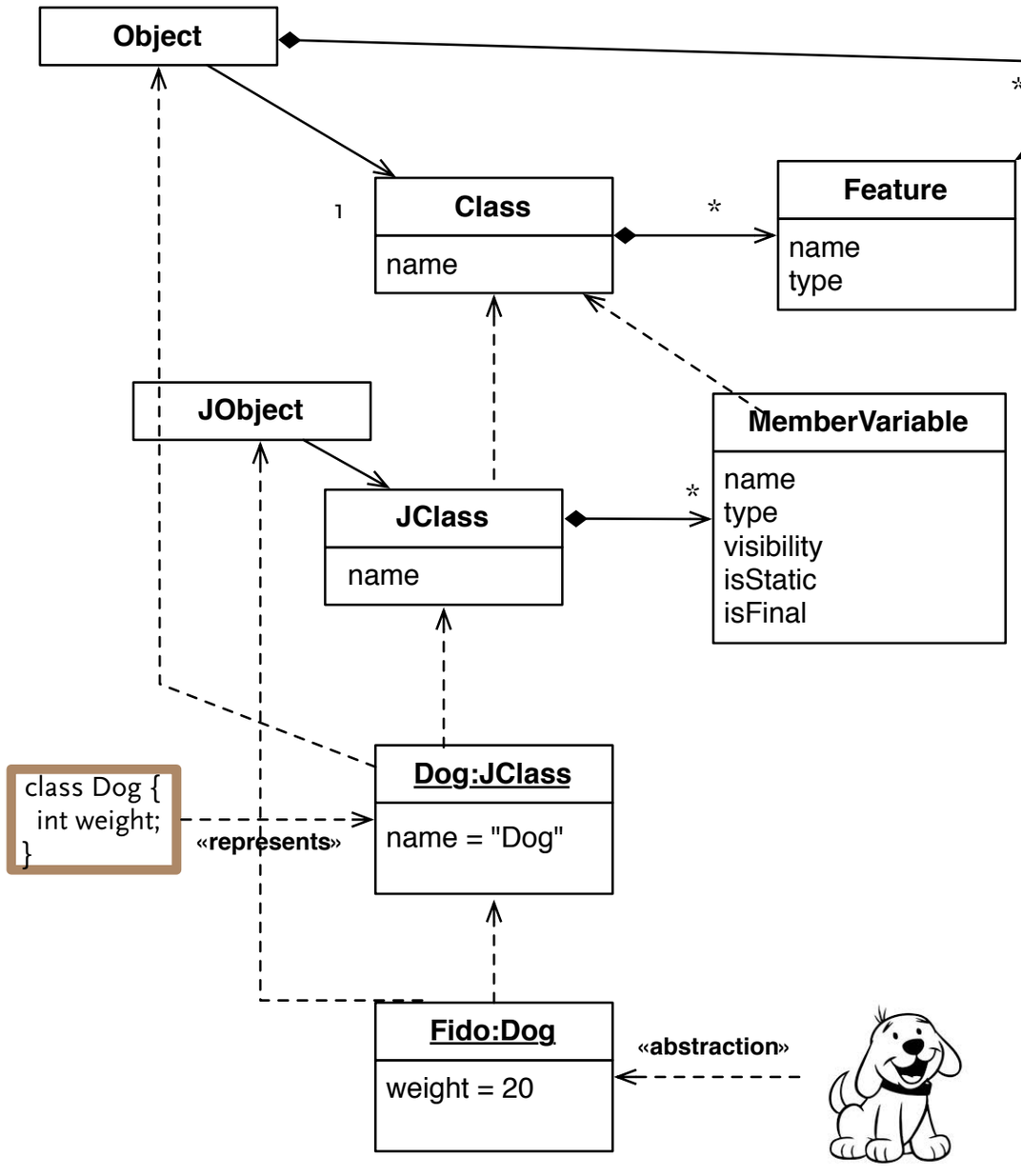


(M3) ?

(M2) Java Spec.

(M1) Java Prog.

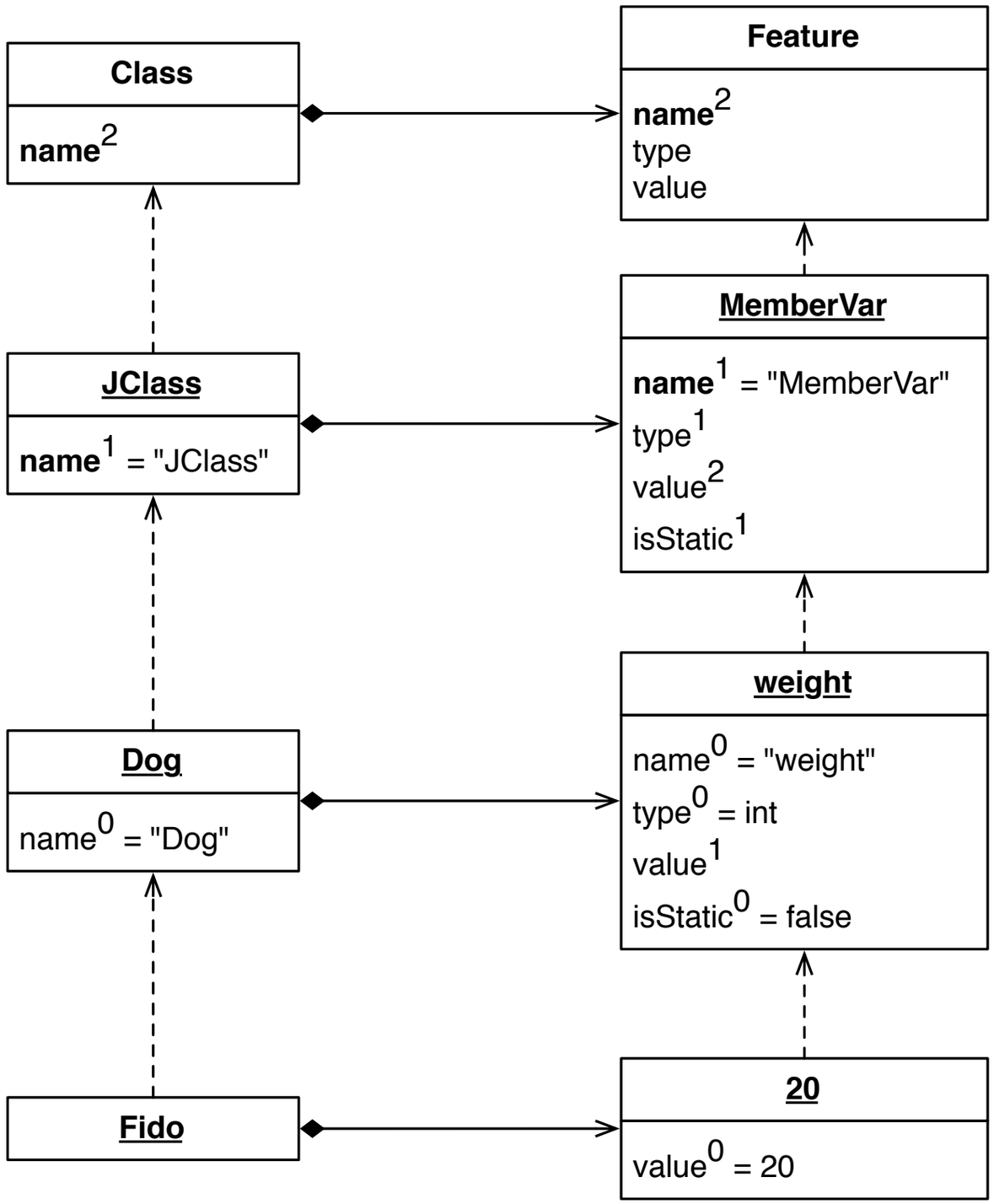
(M0) real things / JVM



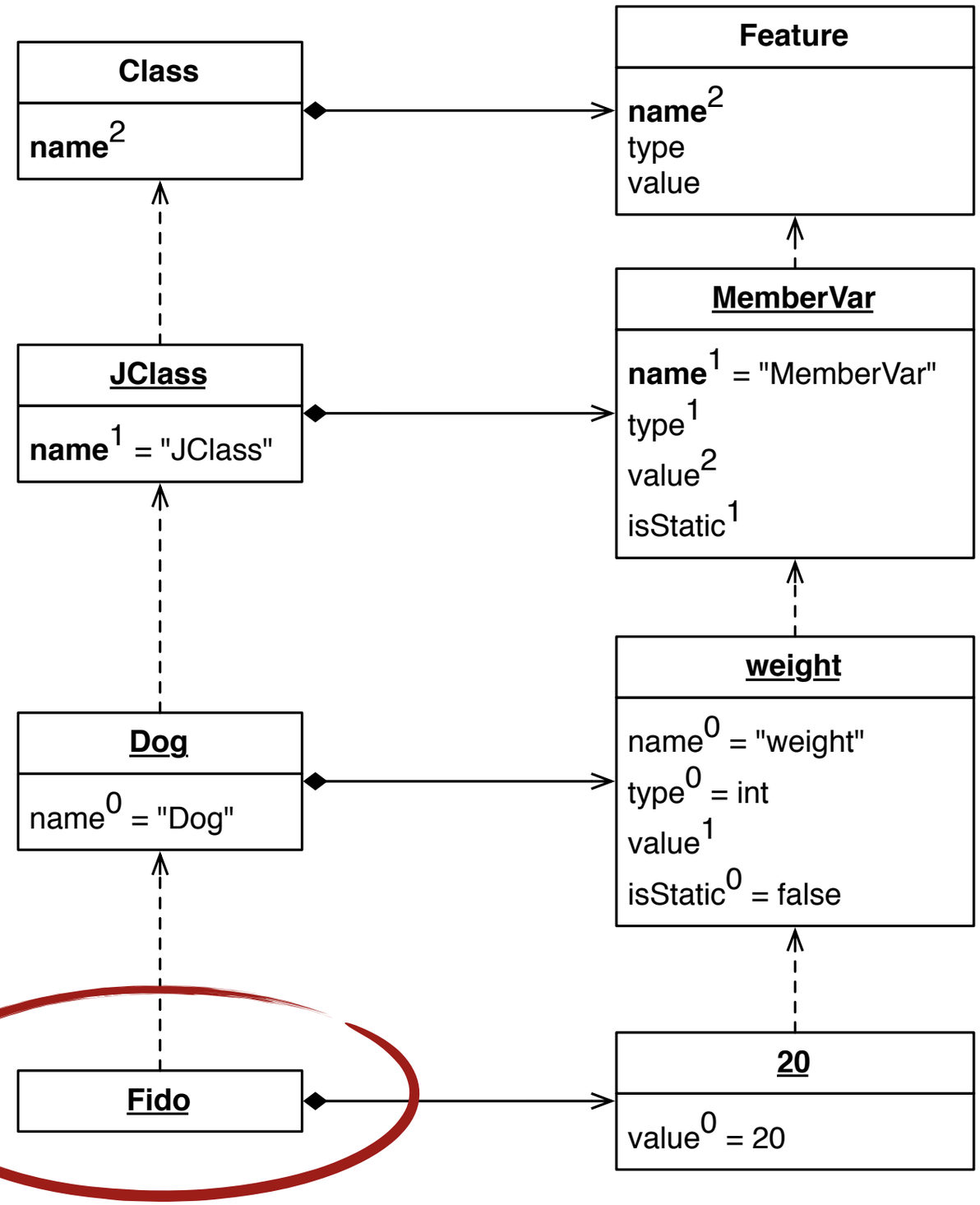
Shallow and Deep Instantiation

- What we used so far is Shallow Instantiation -> Instantiation reached only into the layer below!
- **Deep Instantiation** is a class of instantiation mechanisms that allow define instantiation of several layers to overcome *ambiguous instantiation* and *replication of concepts*.
  - Potency
  - Single and Dual Fields

Deep Instantiation with Potency° and Single/**Dual** Fields



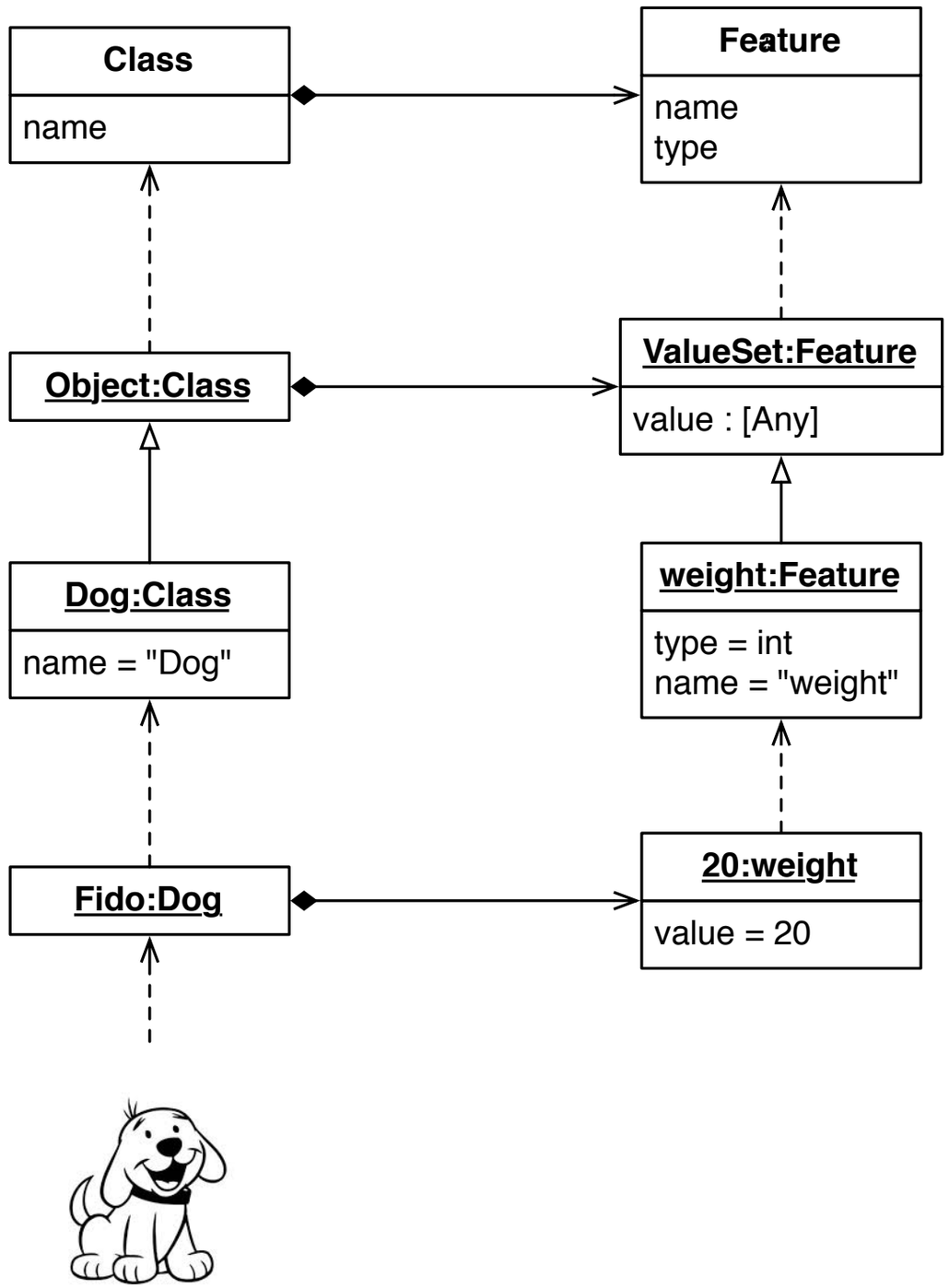
Deep Instantiation with Potency° and Single/**Dual** Fields



"Wo is Fido gespeichert?"

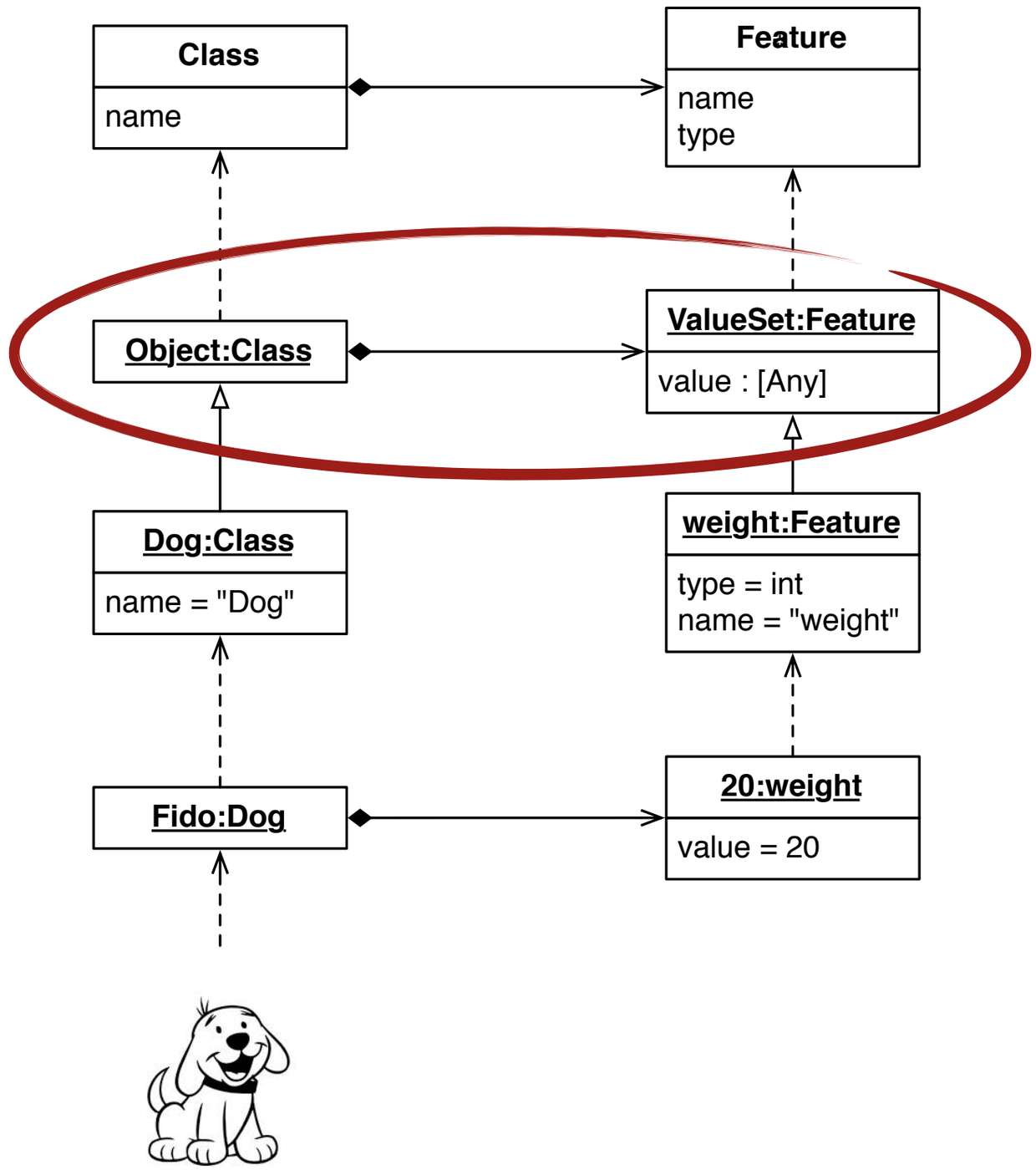
Deep Instantiation with Power Types

- Inheritance of object characteristics instead of instantiation



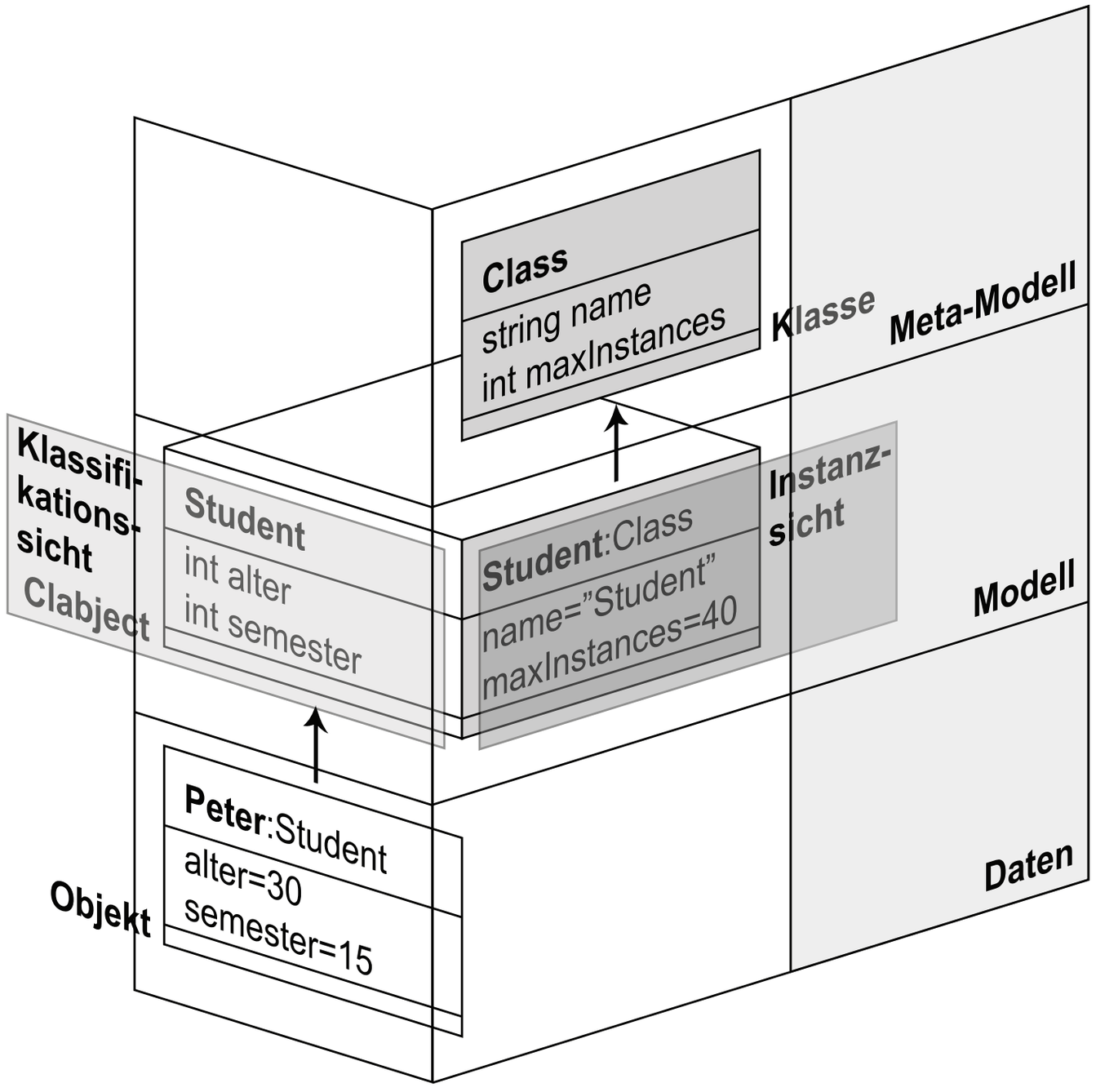
Deep Instantiation with Power Types

- Inheritance of object characteristics instead of instantiation



Clabject

- Model elements are both Objects and Types (Classes) [1]
- Clabject is an illustration of this duality [2]



1. Jim Odell: *Power Types*, Journal of Object-Oriented Programming, 1994
2. Colin Atkinson, Thomas Kühne: *Meta-Modeling Distributed Environments*, Enterprise Distributed Computing, 1997

Examples for Meta-Modeling Formalisms

	OO-Meta-Modeling	**XML**	**Rel. DBs.**	**Grammars**	**Automata**
M3	MOF	Schema-Schema	Rel. Alg.	Math	Math
M2	UML	Schema	E.-R.-M.	Grammar	Automata
M1	Model	XML	DB	Program	Word
M0	Things	Data	Data	Thing	Things

Summary

- Multi- vs. Dual-Level Modeling
- Ambiguous Instantiation and Replication of Concepts
- Deep-Instantiation
- Clabject
- Multi-Level Meta-Modeling Formalisms

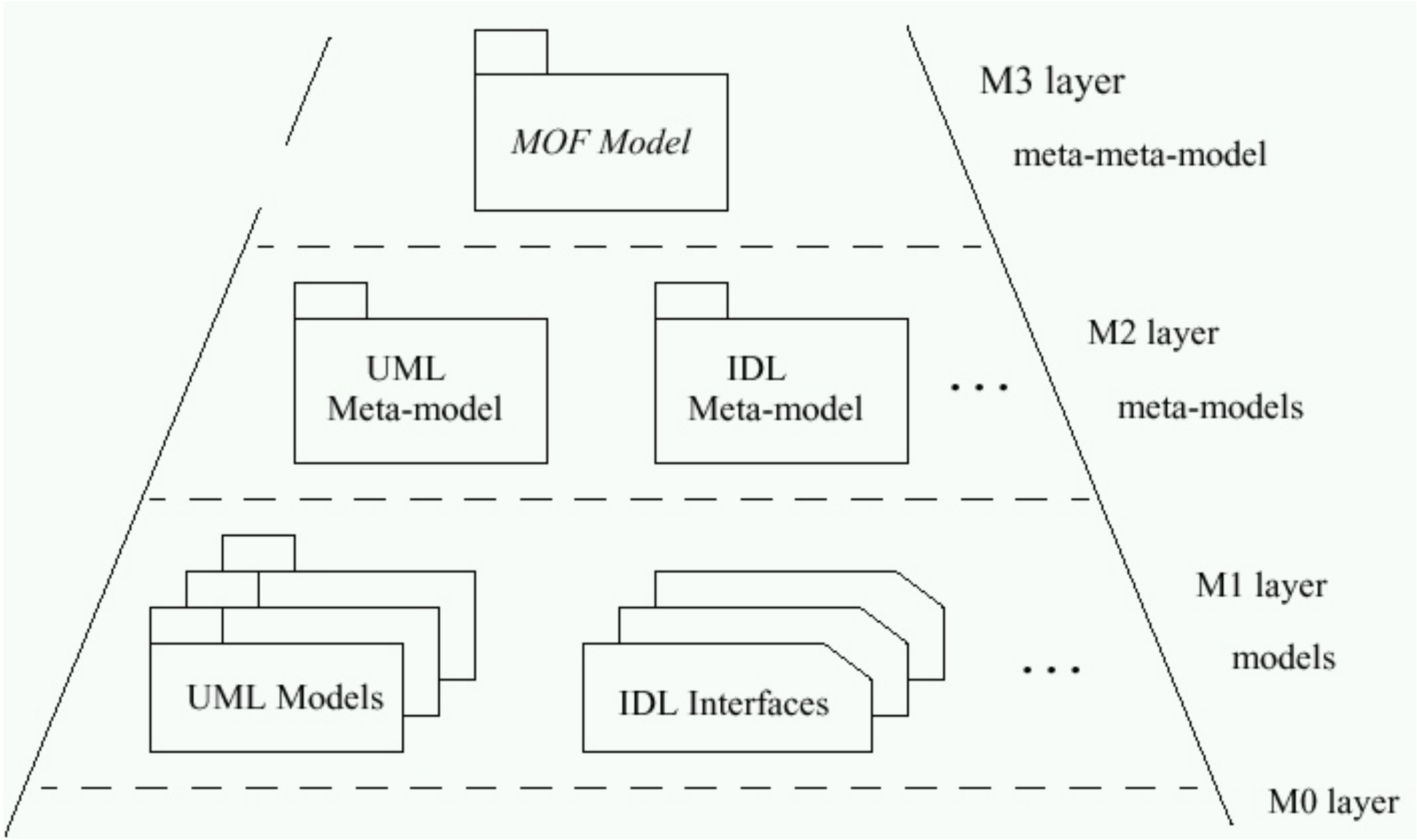
Meta Object Facility 1.x

- Hajime Horiuchi: *A Tutorial on Metamodel Standardization*, Metadata Down Under, 2008

What is MOF (1.x)

- The Meta-Object Facility (MOF) is the OMG's adopted technology for defining metadata and representing it as CORBA objects using UML.
- The MOF 1.3 specification was finalized in September 1999 (OMG document ad/99-09-05).
- The MOF 1.4 specification was submitted to ISO/IEC/JTC1 SC32 by PAS process in 2004 . It has been approved by PAS DIS ballot last December in 2004. (ISO/IEC DIS 19502).
- A MOF metamodel defines the abstract syntax of the metadata in the MOF representation of a model.
- The MOF model itself describes the abstract syntax for representing

MOF in M4


MOF Model
M3 layer
meta-meta-model
UML
Meta-model
IDL
Meta-model
M2 layer
meta-models
UML Models
IDL Interfaces
M1 layer
models
M0 layer

The MOF Model

- The MOF Model is based on the concepts of object relationship modeling.
- The three kinds of building blocks for a meta-information model are
  - Objects (described by MOF Classes),
  - Links that connect objects (described by MOF Associations),
  - Data values (described by CORBA IDL types).
- Instances of these constructs are organized as MOF Packages. CORBA(IDL)

Core portion of MOF

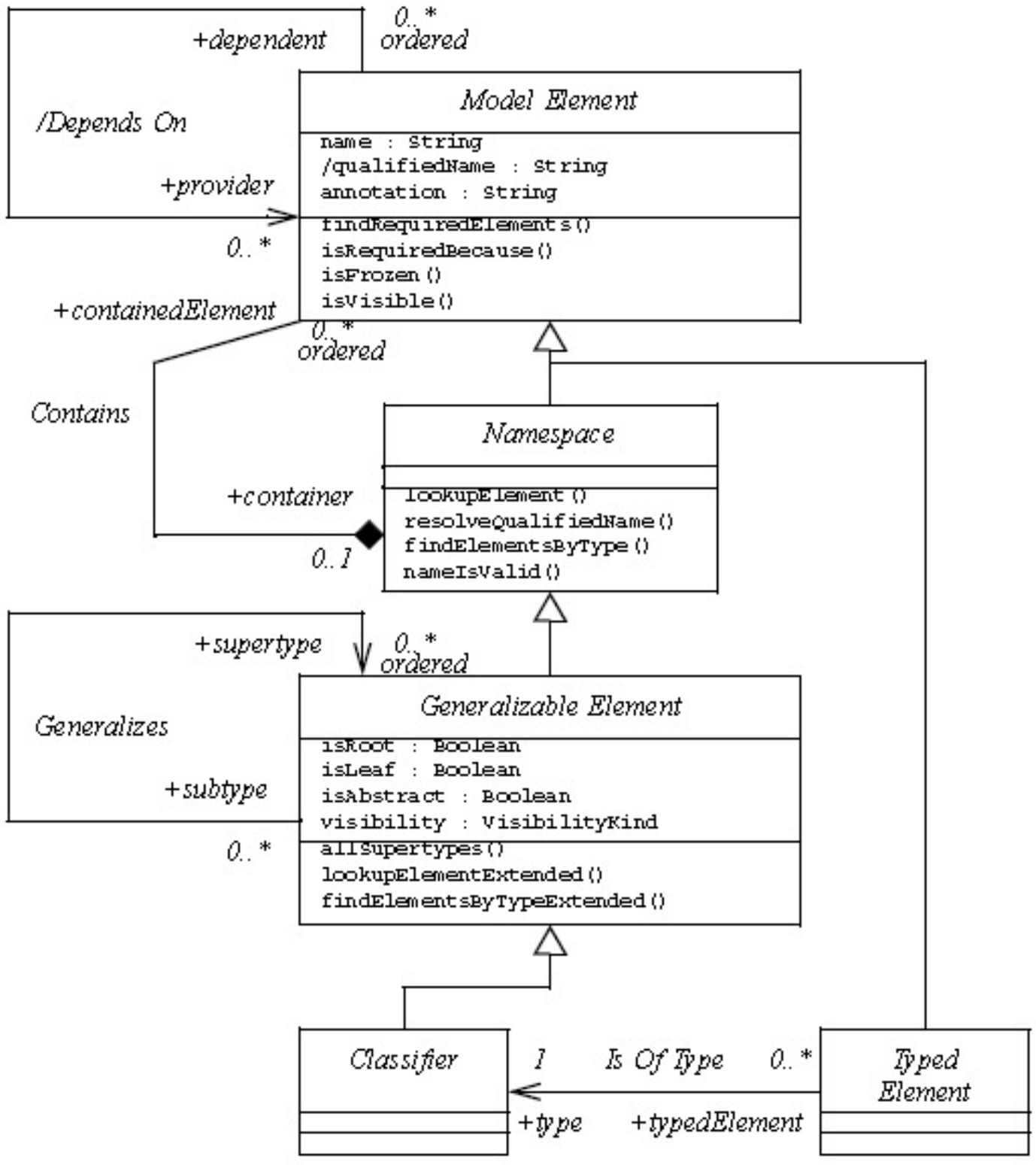


Figure 7.2 - The Key Abstractions of the MOF Model

Feature in MOF

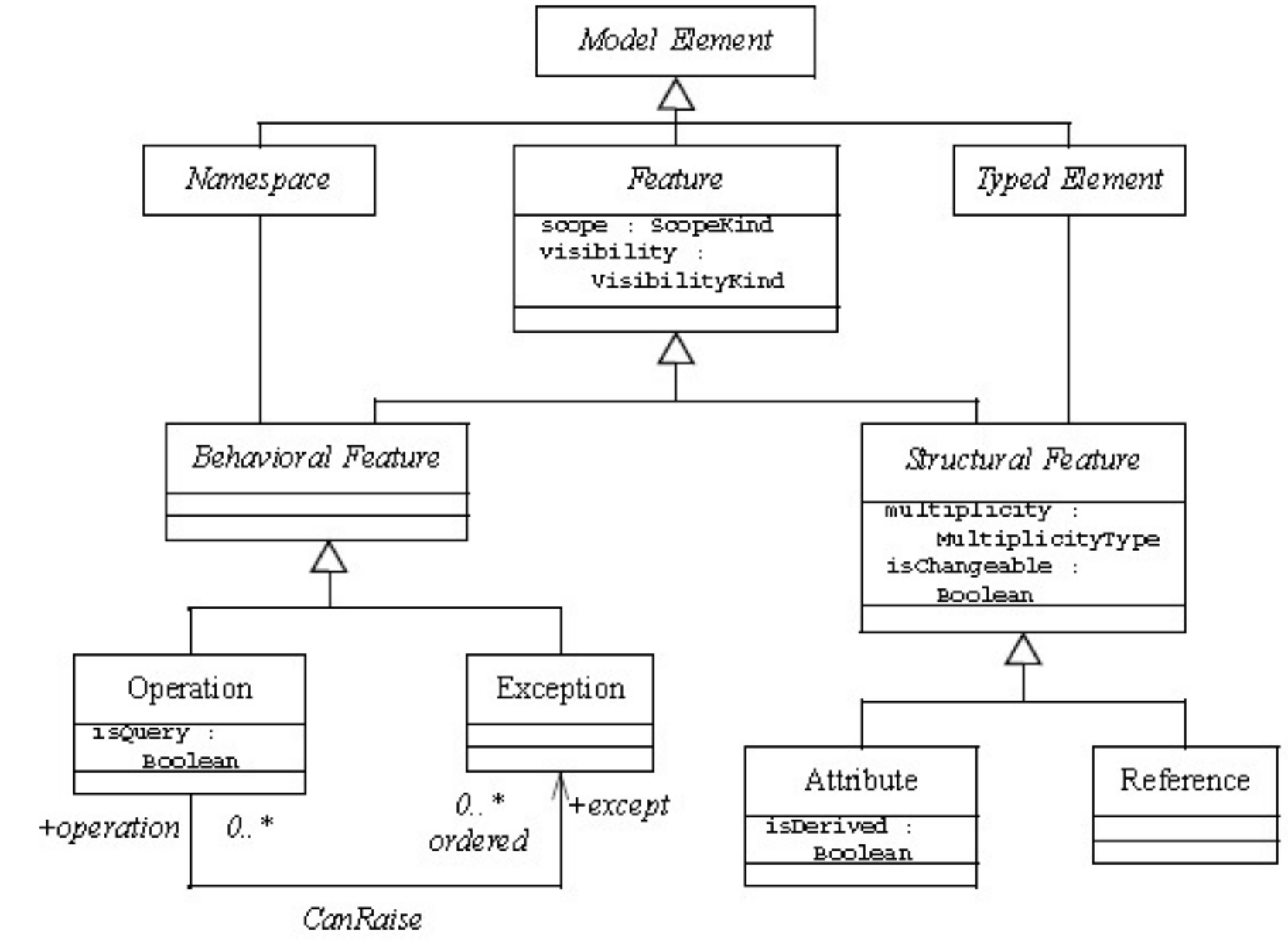


Figure 7.5 - Feature Classes of the MOF Model

TypedElement

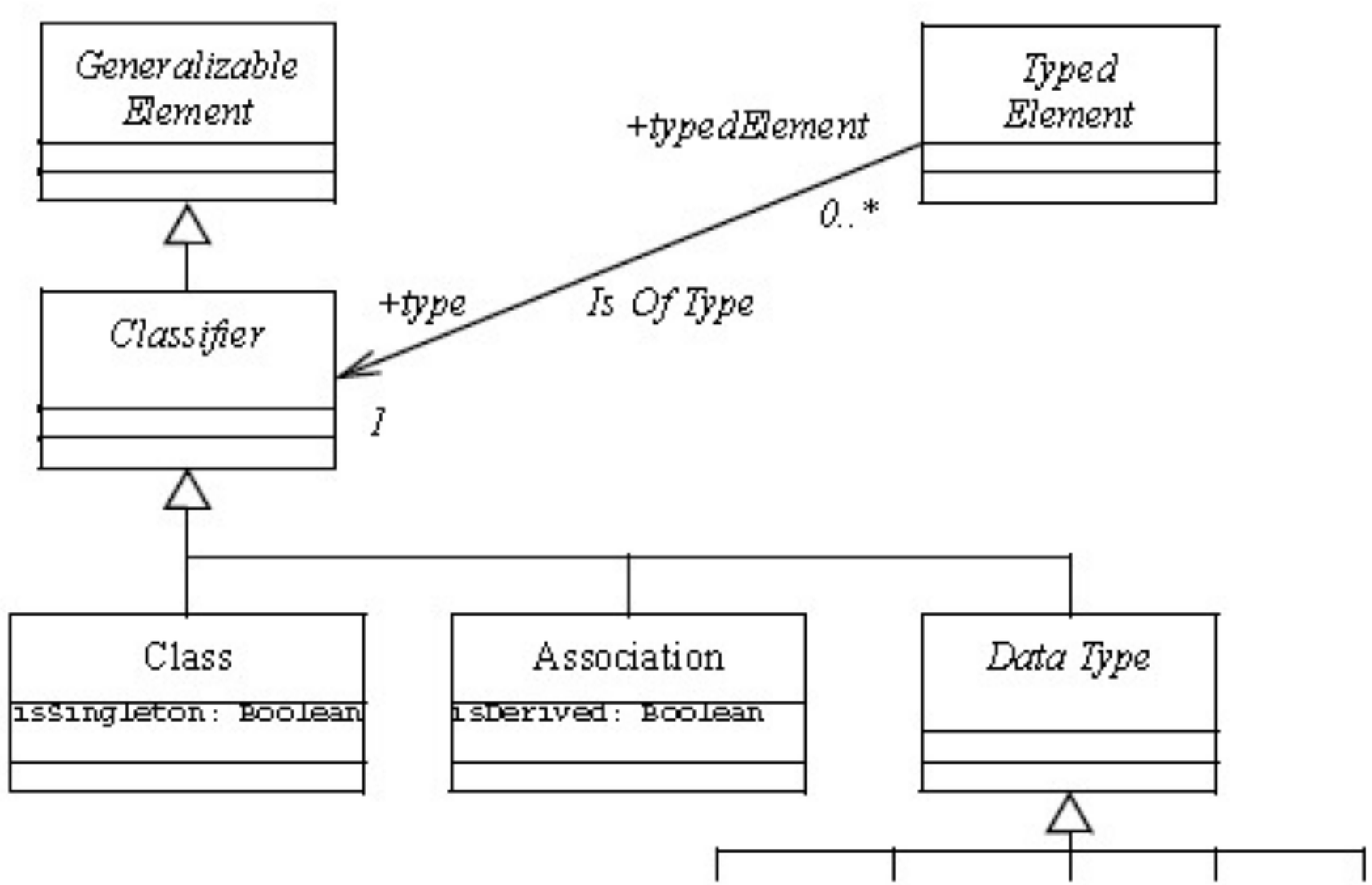


Figure 7.3 - MOF Model Classifiers

Data Type

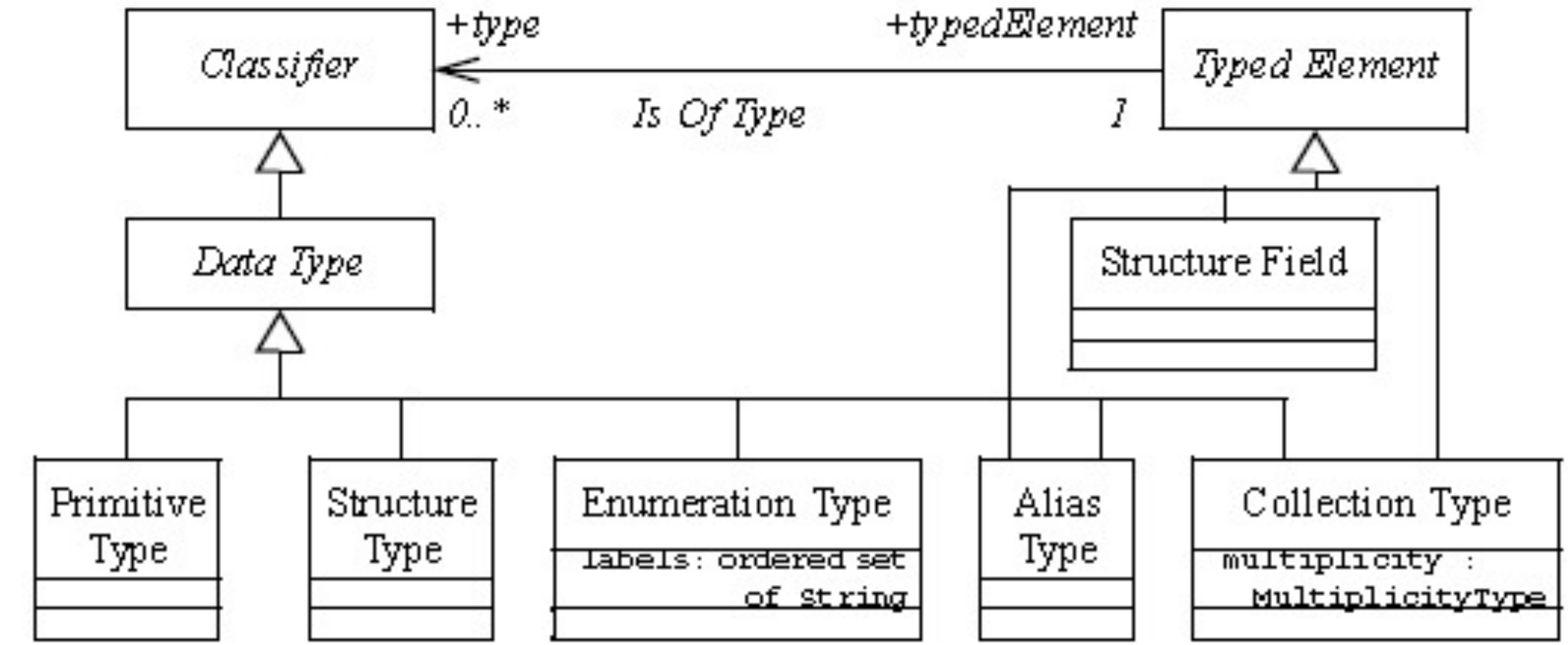


Figure 7.4 - MOF Data Type Elements

Association

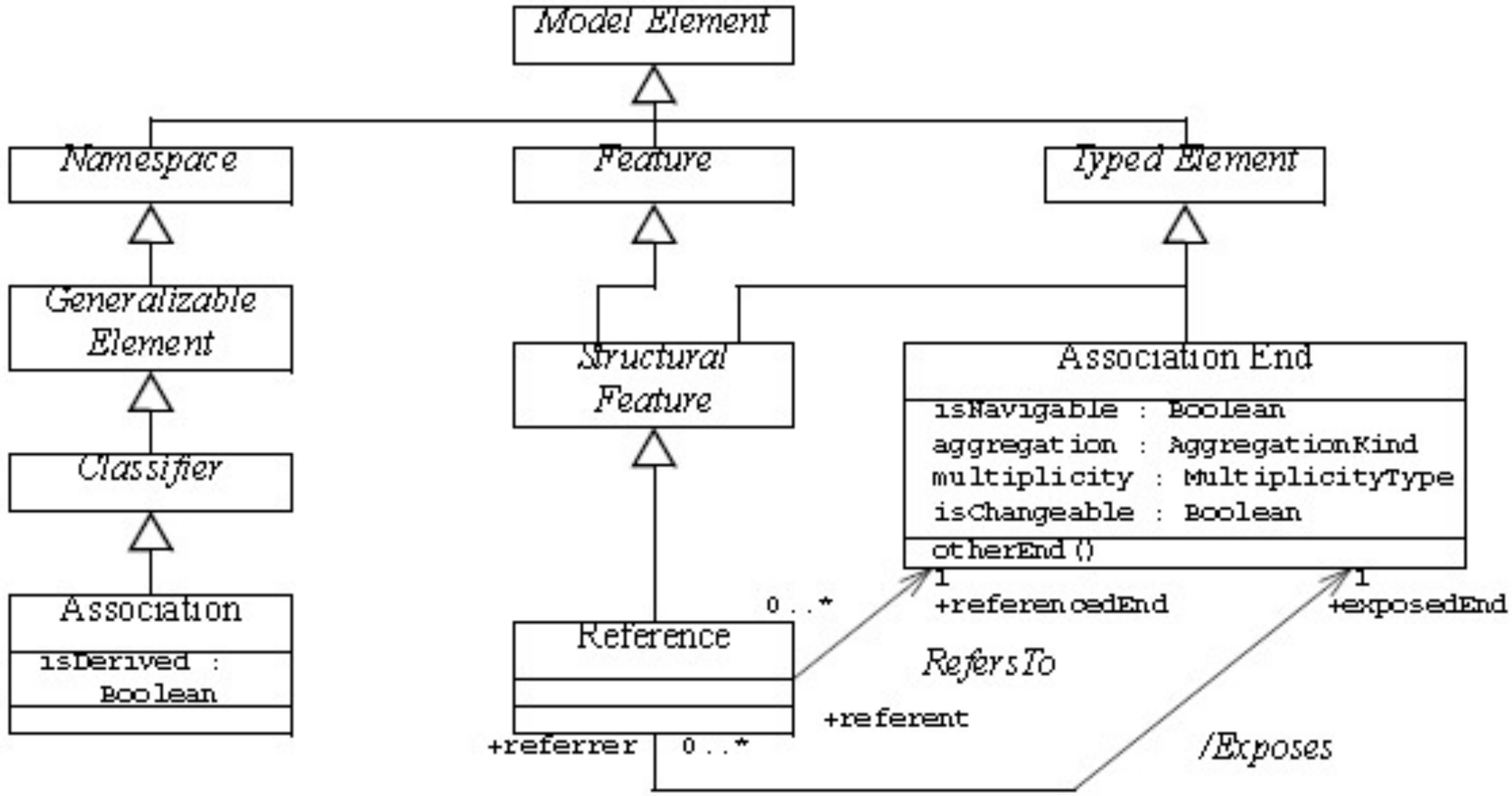


Figure 7.6 - MOF Model Elements for Associations

MOF, CORBA, XMI

- MOF
  - Is a generic framework for describing and representing meta-information in an CORBA-based environment.
  - Defines an IDL mapping which allows models expressed using MOF Model constructs to be translated into interfaces CORBA-based meta-information services.
- CORBA(IDL)
  - Generate IDLs from metamodel written by MOF
  - IDLs are interfaces for the repository to store metamodels.
  - Generate IDLs from UML metamodel
- XMI （XML metamodel Interchange)

Meta Object Facility 2.0

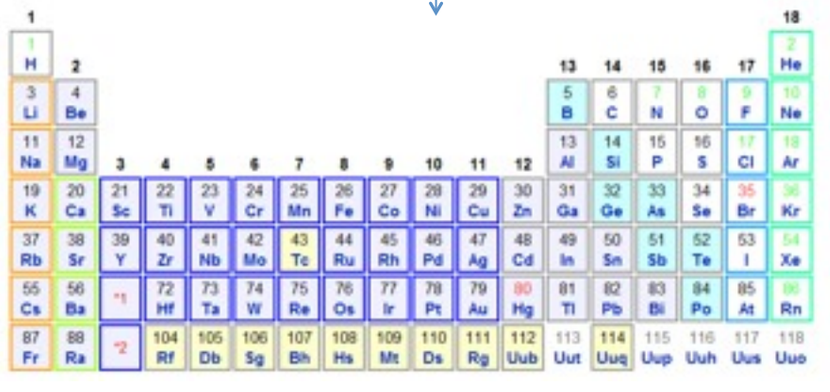
- Hajime Horiuchi: *A Tutorial on Metamodel Standardization*, Metadata Down Under, 2008

MOF/UML 2.0 – Don't call it "Wine"

Wine

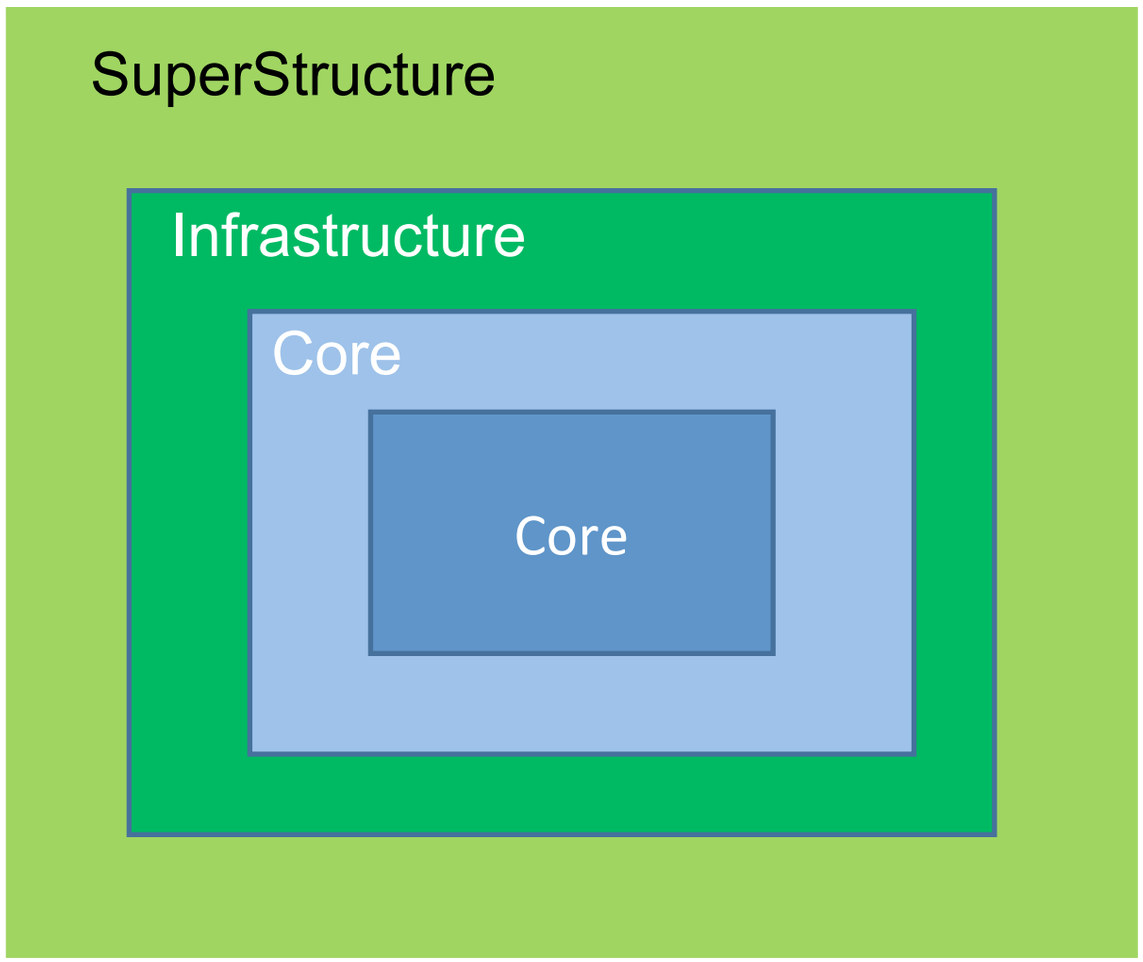
Water + alcohol + grape essence + mold +....

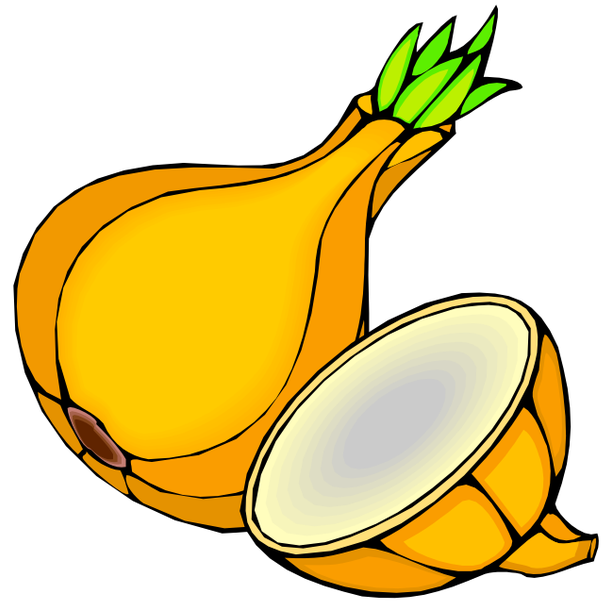
$H_2O + CH_3CH(OH)CH_3 + ...$



Atomic periodic table

So Many Cores

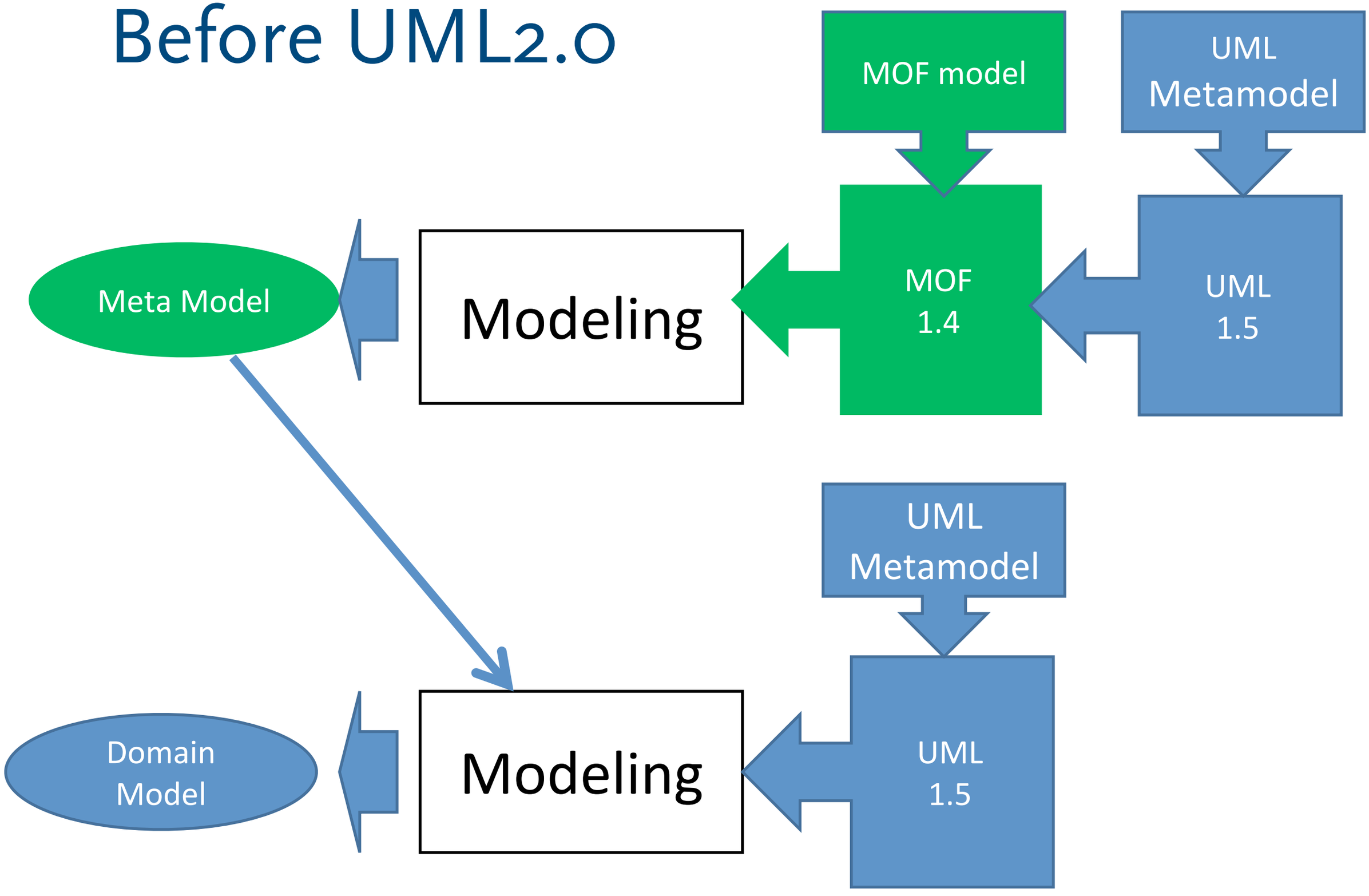




Major changes in the UML at UML 2.0

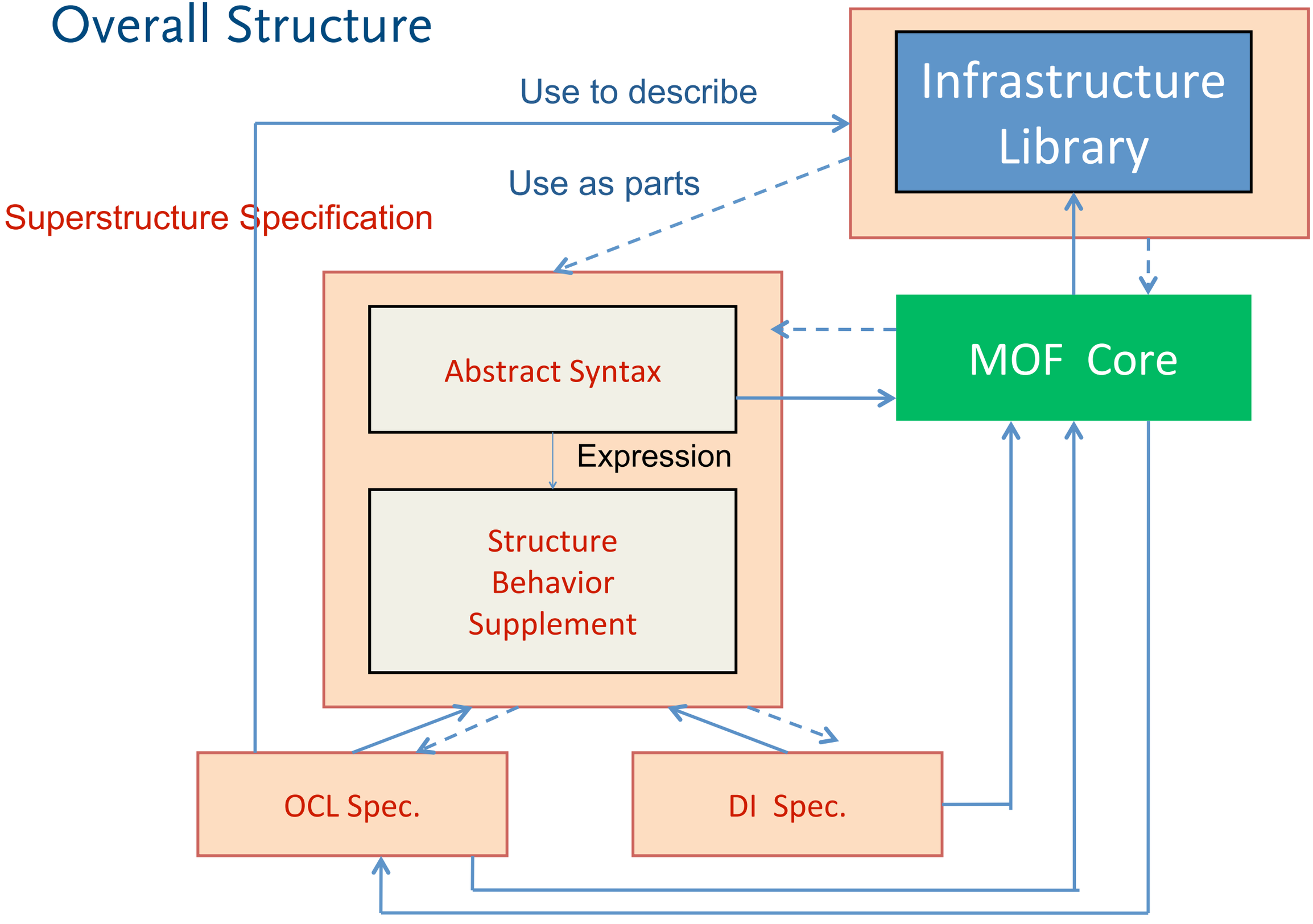
- Metamodeling Facilities in MOF and UML were Integrated as the Infrastructure Library (MOF model still exist)
- Composite Structure
- UML Profile Mechanism
- Little changes in the diagrams

Before UML2.0



UML 2.0

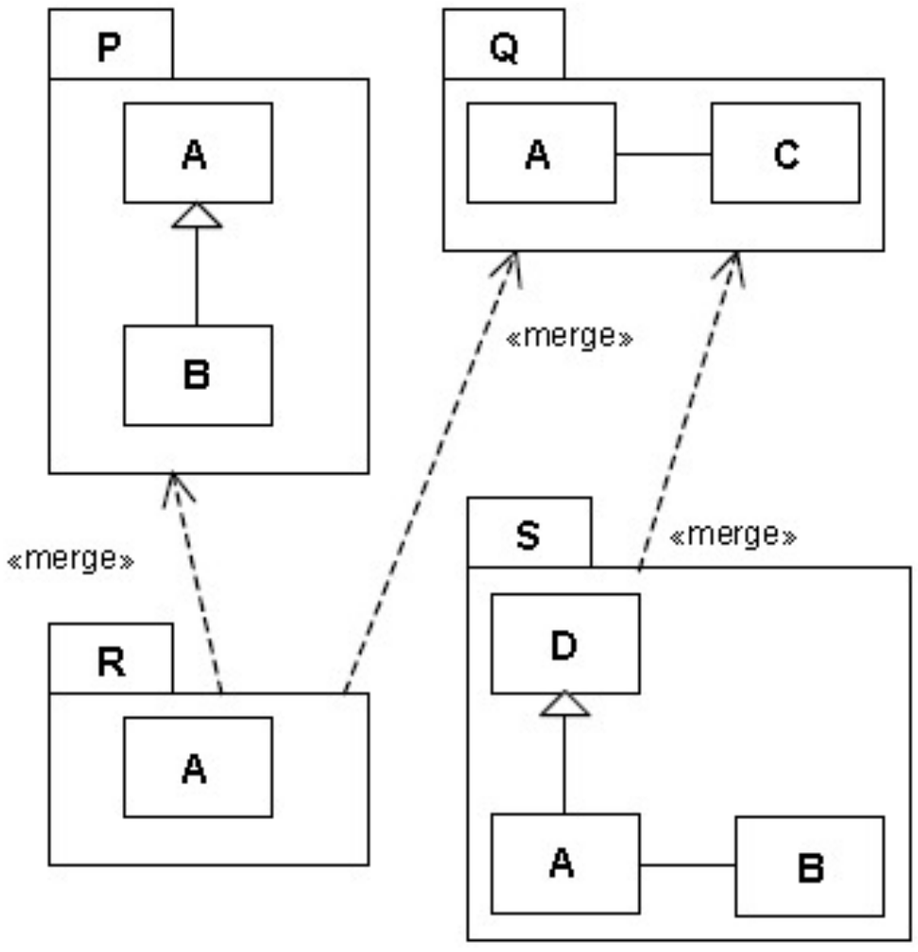
Overall Structure

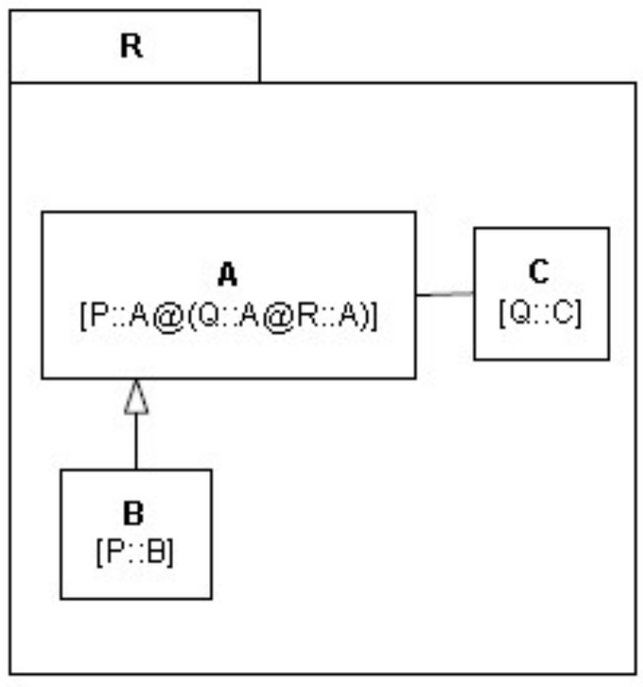


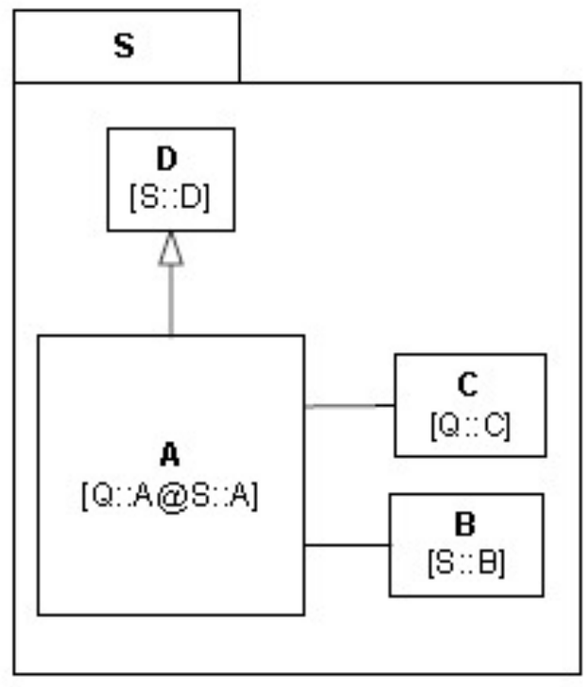
Meta level concept in UML2.0 & MOF2.0

- Multiple levels still used
- No rigid structure to the Four Layer
- The meta layer could be extended to any level more than two, relatively
- However, Type and Instance should be within two levels (Level-pair)
- The deep instantiation could be available with the Reflection capability of MOF

Composition Structure



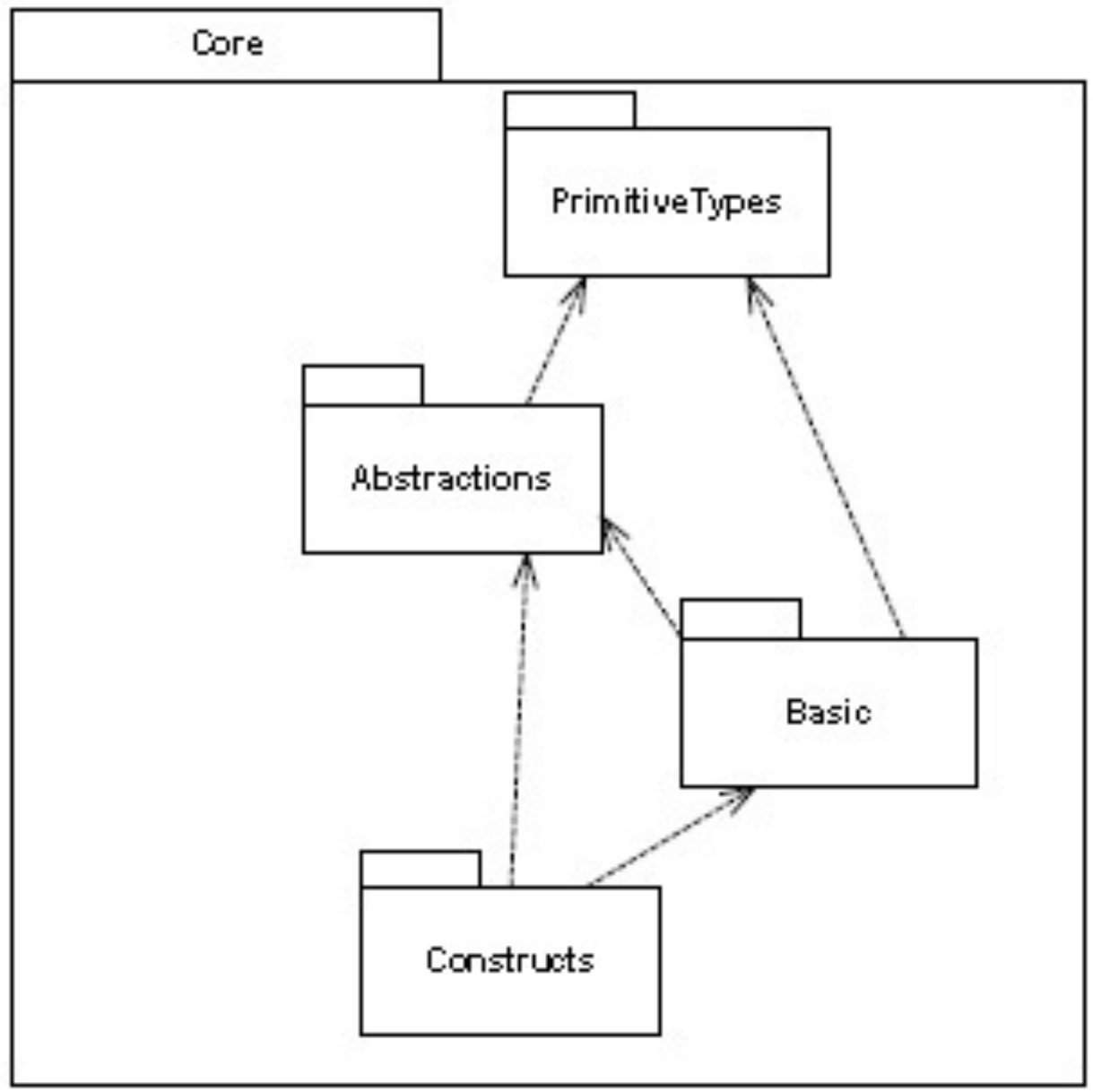




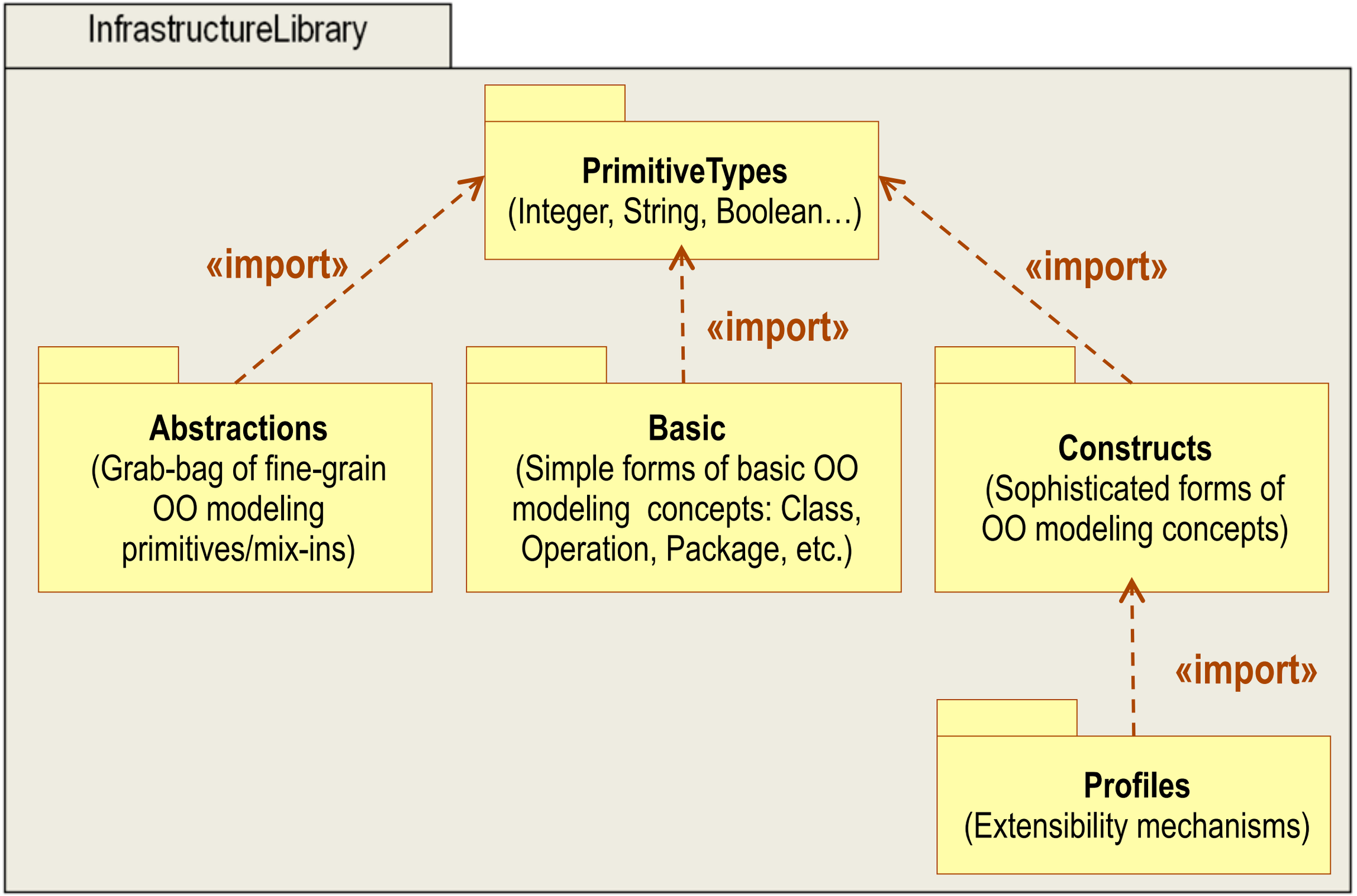
The Core of Cores:

UML 2.0 Infrastructure

Infrastructure Library



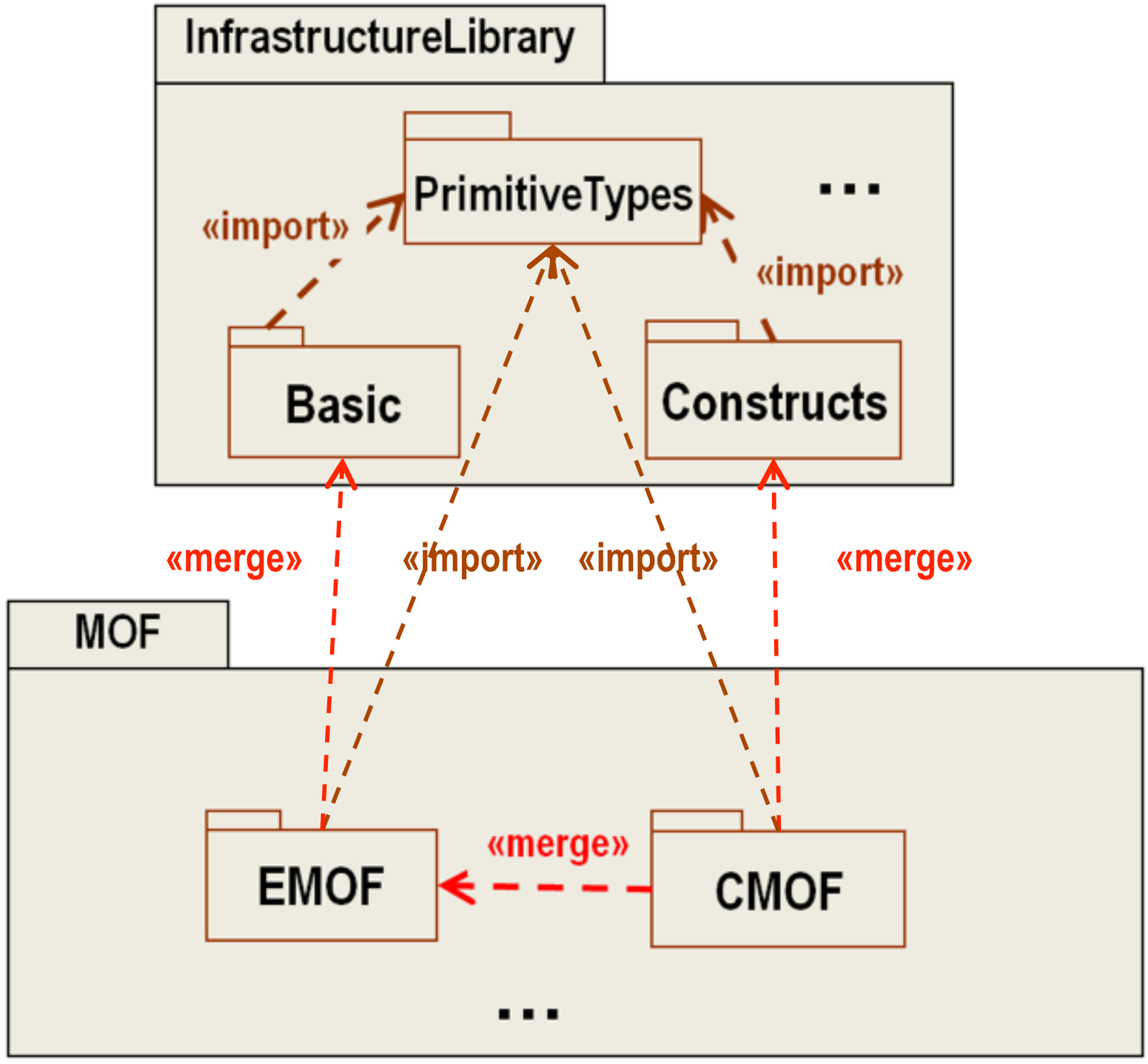
Infrastructure Library – Contents



MOF2.0 in UML2.0

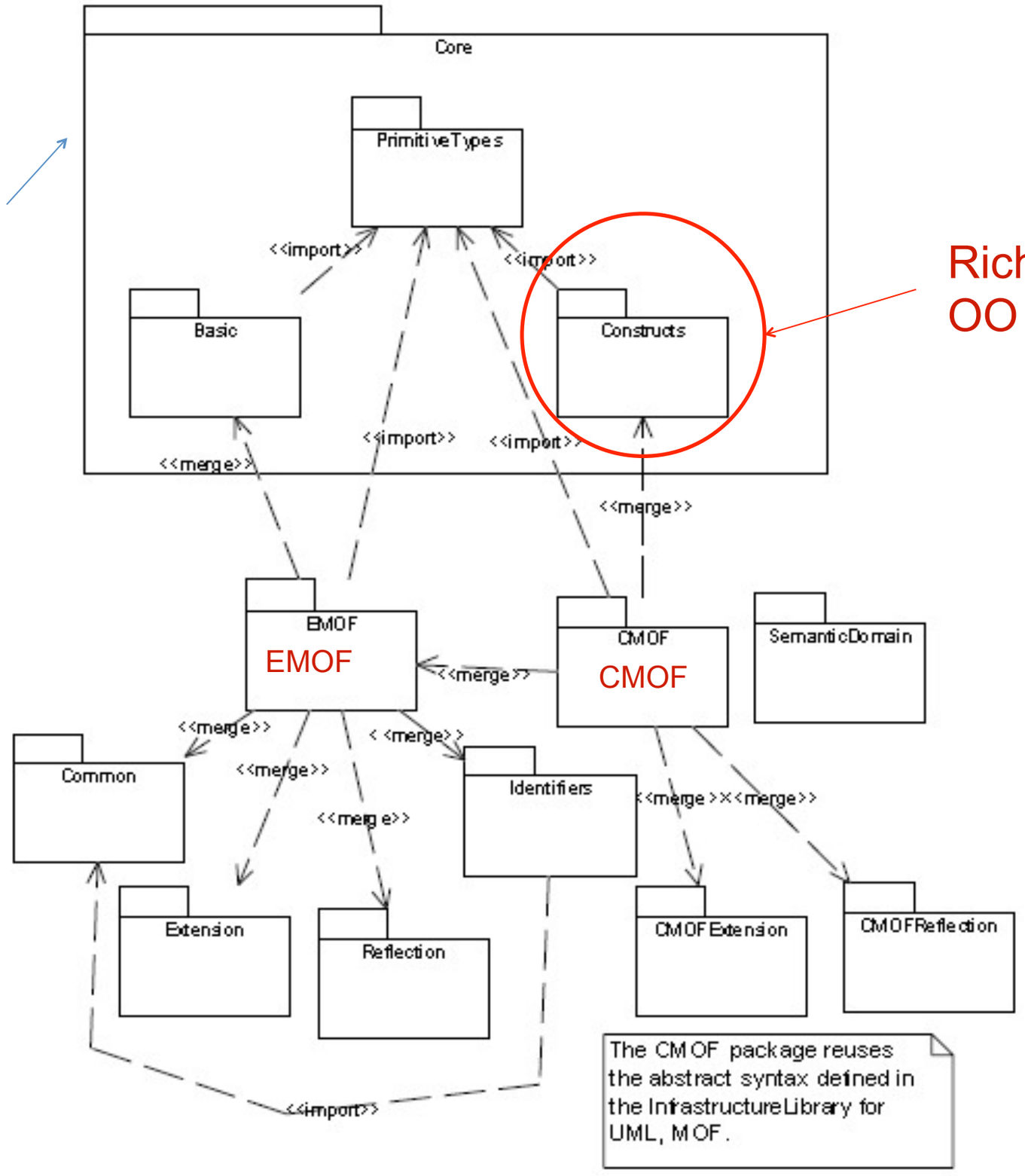
- MOF 2.0 use UML Infrastructure Library
- Define a metalanguage kernel that can define (bootstrap) UML and also be reused to define other OMG MDA metamodels (e.g., MOF, CWM, ODM)
- Provide more powerful mechanisms to customize UML
- Allow users to define language dialects for platforms (e.g., J2EE, .NET) and domains (e.g., ebusiness, finance, etc.)

Infrastructure Library – Usage



Difference of EMOF & CMOF

UML2.0
Infrastructure::
Infrastructure
Library:: core



Rich constructs for OO modeling

e.g.
association,
Package,
.
.

EMOF (Essential MOF)

- Essential MOF is the subset of MOF that closely corresponds to the facilities found in OOPLs and XML.
- The value of Essential MOF is that it provides a straightforward framework for mapping MOF models to implementations such as JMI and XMI for simple metamodels.
- A primary goal of EMOF is to allow simple metamodels to be defined using simple concepts
- reuse the UML2 Infrastructure.
- The EMOF Model imports two packages from MOF.
  - Identity

CMOF: Complete MOF Model

- The CMOF Model is the metamodel used to specify other metamodels such as UML2.
- It is built from the Core of UML 2.
- The Model package does not define any classes of its own. Rather, it imports packages that together define basic modeling capabilities.