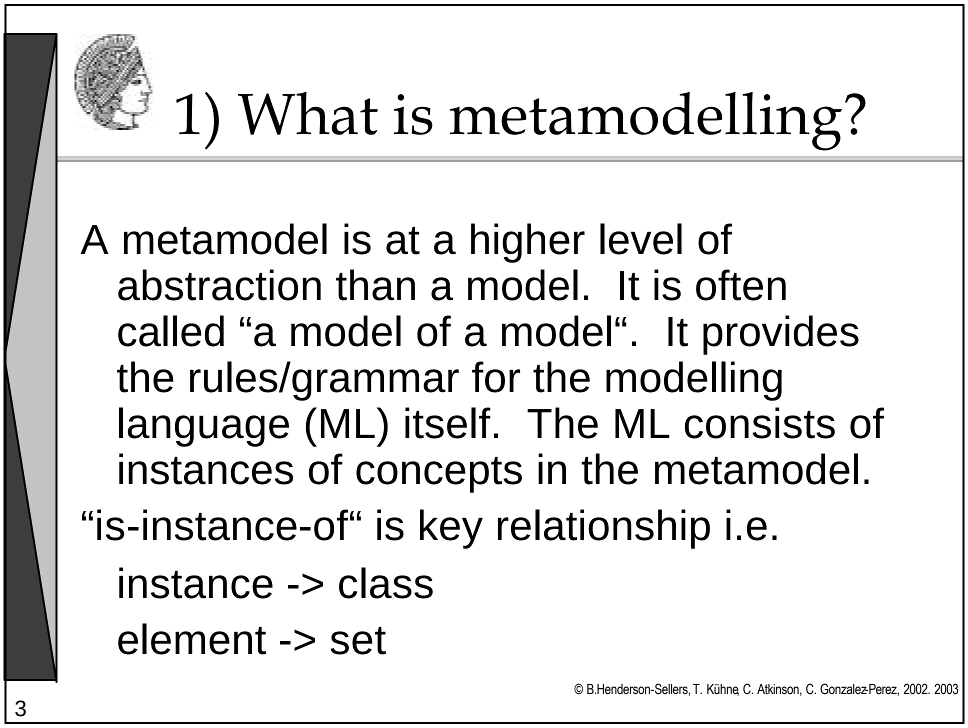
Understanding Metamodelling

ER2003 15 October 2003

Presented by Brian Henderson-Sellers
With preparation input from
C. Atkinson, T. Kühne and C. Gonzalez-Perez
brian@it.uts.edu.au

Topics

1) What is metamodelling?
2) The meaning of "is-a"
3) UML extension mechanisms
4) Process modelling in the context of UML metamodelling hierarchy
5) Summary



1) What is metamodelling?

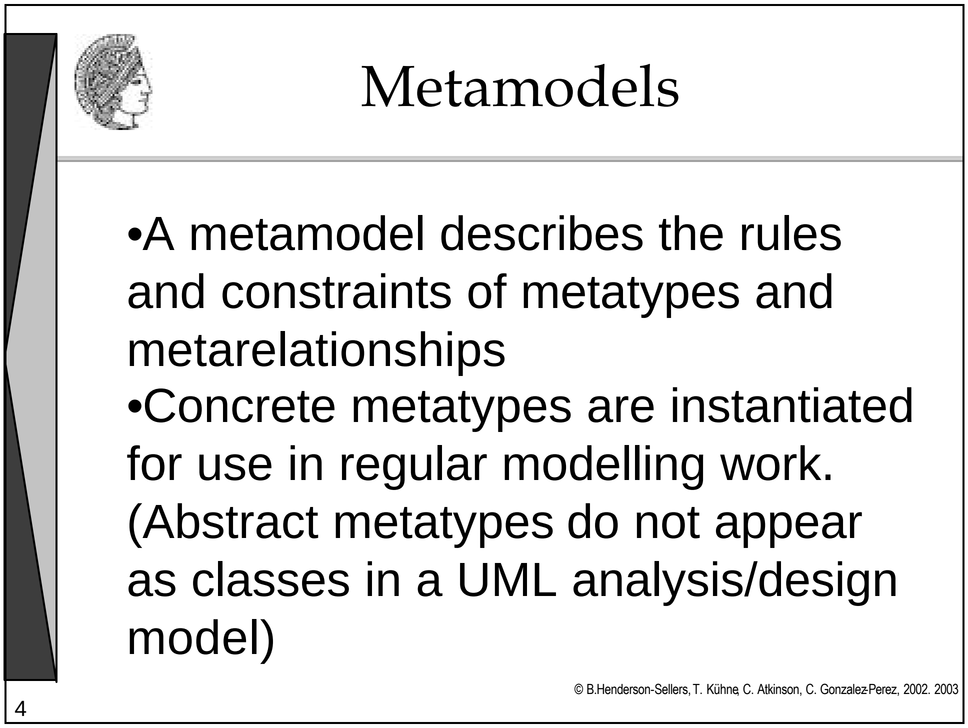
A metamodel is at a higher level of abstraction than a model. It is often called “a model of a model“. It provides the rules/grammar for the modelling language (ML) itself. The ML consists of instances of concepts in the metamodel.

“is-instance-of“ is key relationship i.e.

instance -> class

element -> set

3

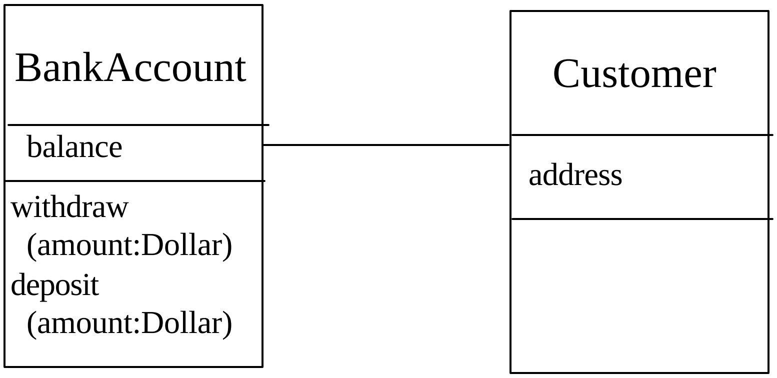


Metamodels

- A metamodel describes the rules and constraints of metatypes and metarelationships
- Concrete metatypes are instantiated for use in regular modelling work. (Abstract metatypes do not appear as classes in a UML analysis/design model)

4

Example model which uses the rules of UML



BankAccount

balance

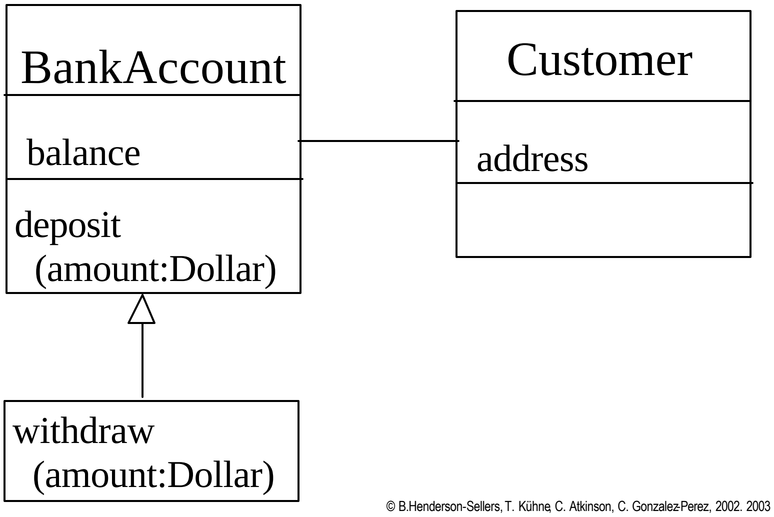
withdraw
(amount:Dollar)
deposit
(amount:Dollar)

Customer

address

5

Example model which does not adhere to UML



BankAccount

balance

deposit
(amount:Dollar)

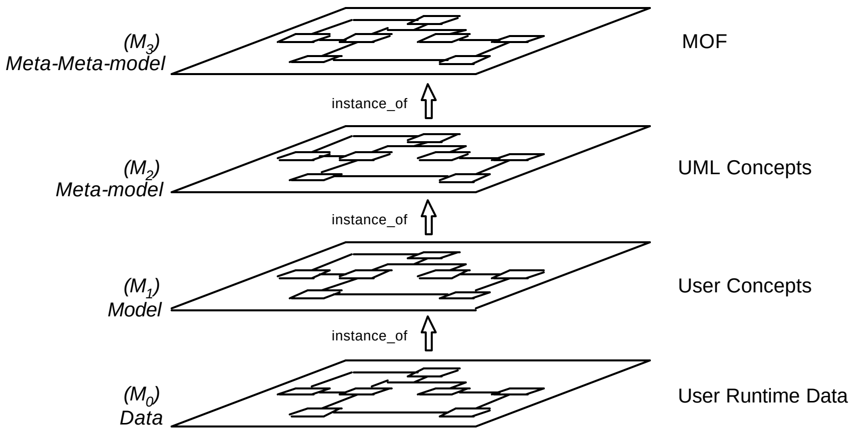
withdraw
(amount:Dollar)

Customer

address

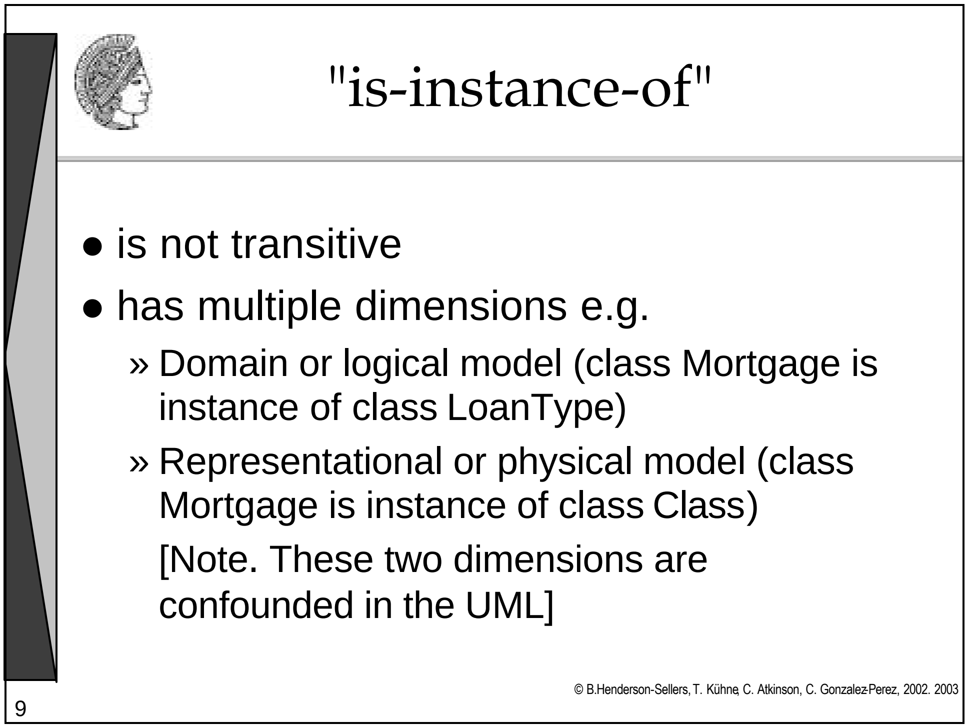
6

OMG's 4-Layer Hierarchy



The 4-Layer Hierarchy

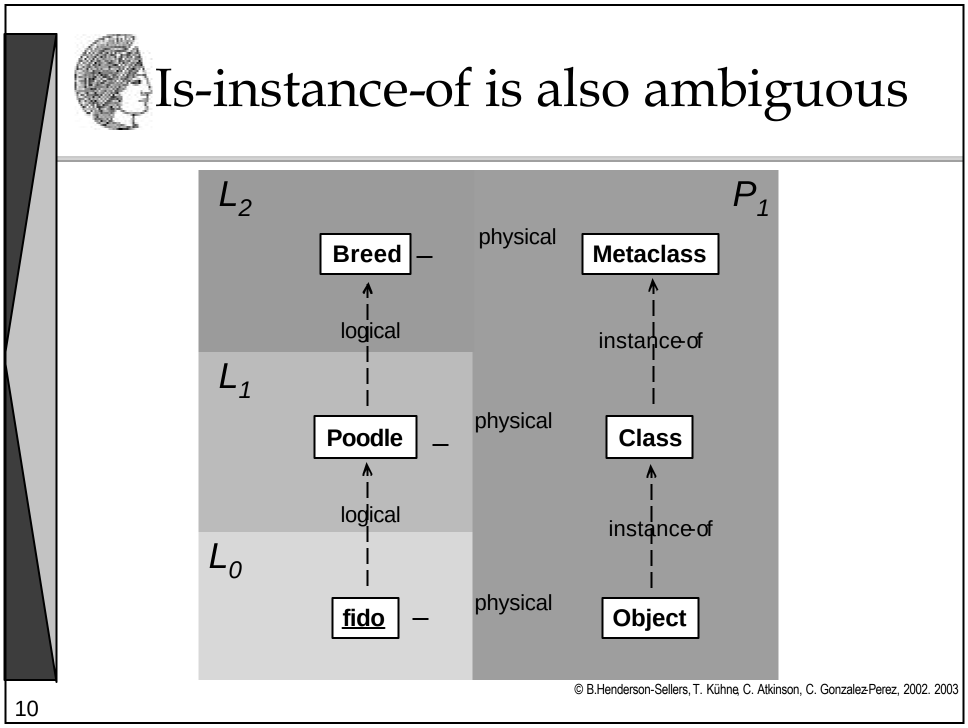
Layer	**Description**	**Example**
meta-metamodel	The infrastructure for a metamodeling architecture. Defines the language for specifying metamodels.	*MetaClass, MetaAttribute, MetaOperation*
metamodel	An instance of a meta-metamodel. Defines the language for specifying a model.	*Class, Attribute, Operation, Component*
model	An instance of a metamodel. Defines a language to describe an information domain.	*StockShare, askPrice, sellLimitOrder, StockQuoteServer*
user objects (user data)	An instance of a model. Defines a specific information domain.	*<Acme_SW_Share_98789>, 654.56, sell_limit_order, <Stock_Quote_Svr_32123>*



"is-instance-of"

- is not transitive
- has multiple dimensions e.g.
  - » Domain or logical model (class Mortgage is instance of class LoanType)
  - » Representational or physical model (class Mortgage is instance of class Class)

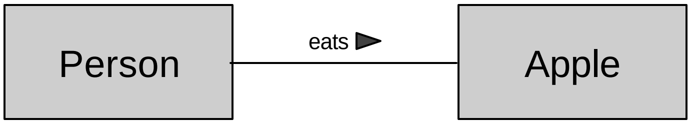
    [Note. These two dimensions are confounded in the UML]



Is-instance-of is also ambiguous

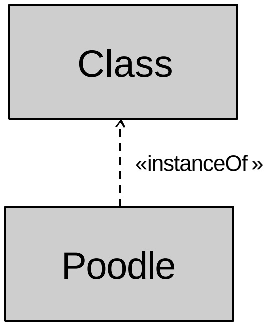
Convention in UML/OO

is that the rectangles in a diagram are surrogates for their instances e.g.



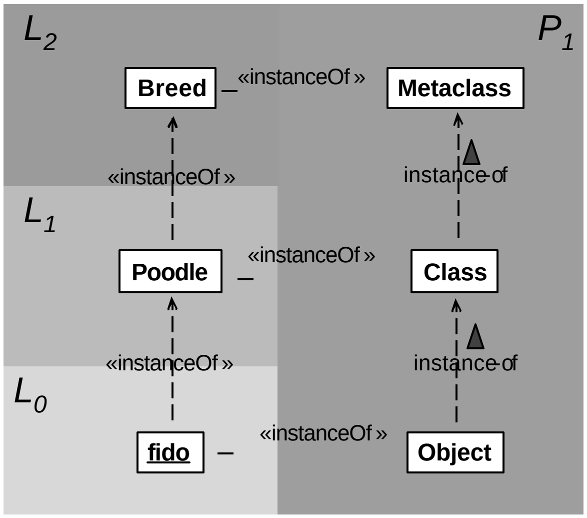
means an instance of class Person eats an instance of class Apple NOT that class Person eats class Apple

BUT instance-of relationship violates this rule



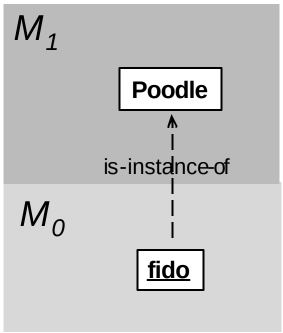
refers to actual things on diagram, *not* to instances of it (as on previous slide) viz. class Poodle is an instance of metaclass Class

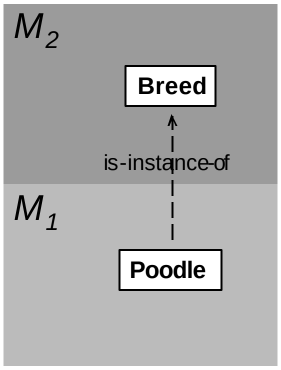
Less ambiguity if we used different notations



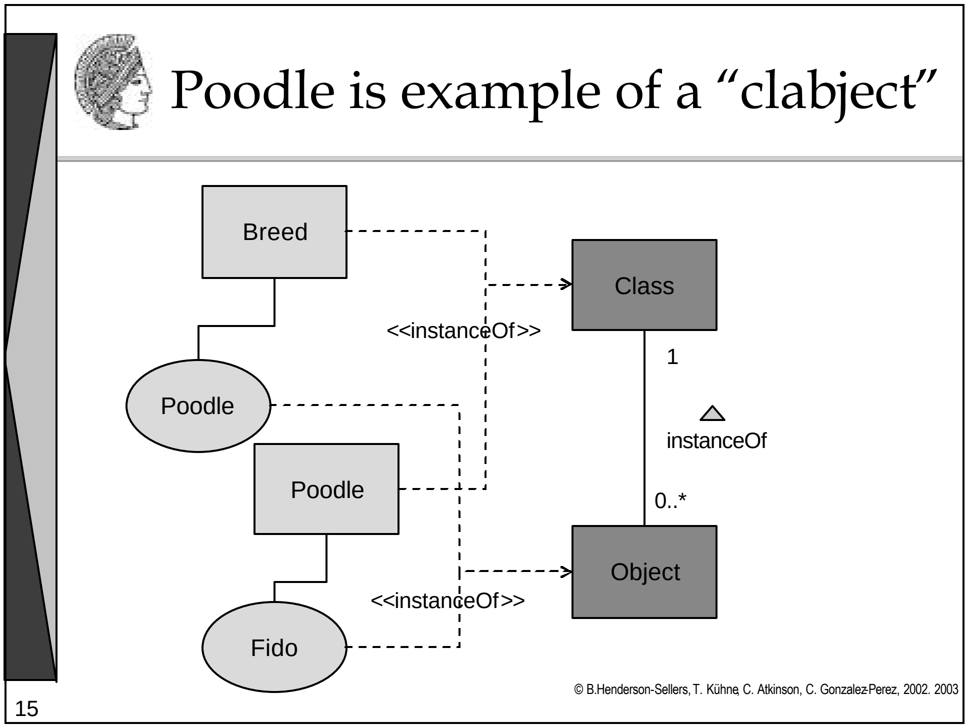
Classes as Objects

Poodle as class



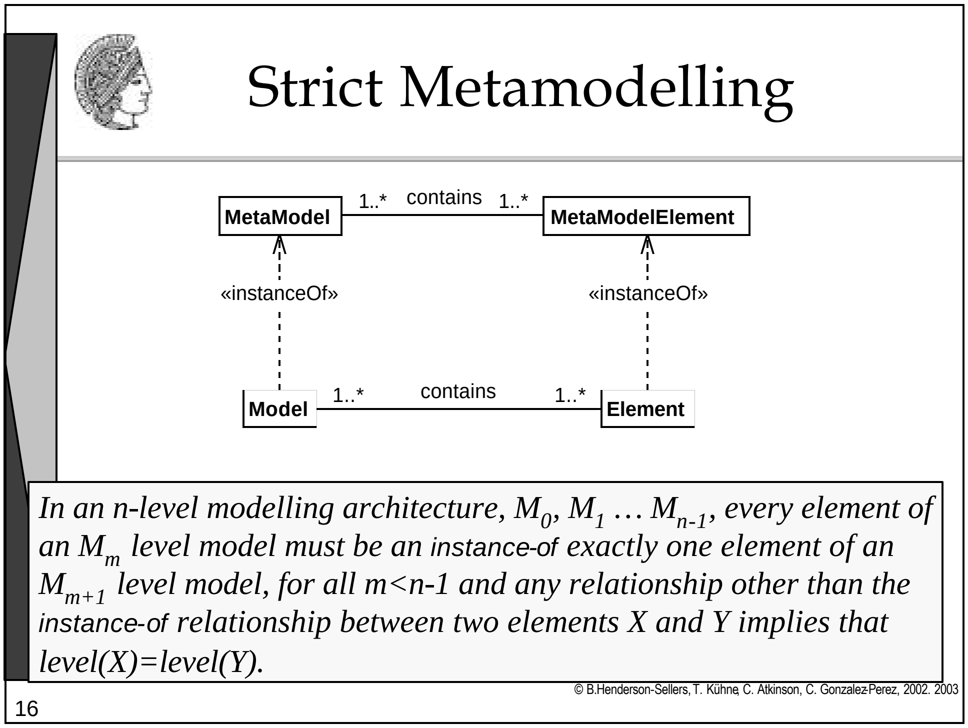


Poodle as object



Poodle is example of a “clabject”

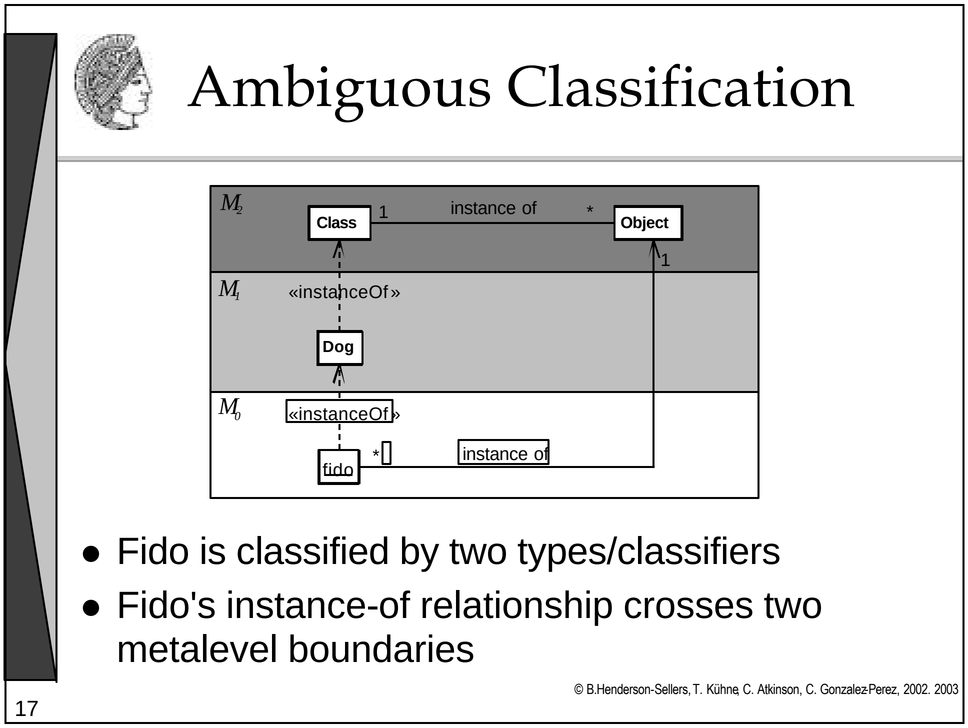
15



Strict Metamodelling

In an n-level modelling architecture, M_0, M_1 … M_{n-1}, every element of an M_m level model must be an instance-of *exactly one element of an M_{m+1} level model, for all m<n-1 and any relationship other than the* instance-of *relationship between two elements X and Y implies that level(X)=level(Y).*

16

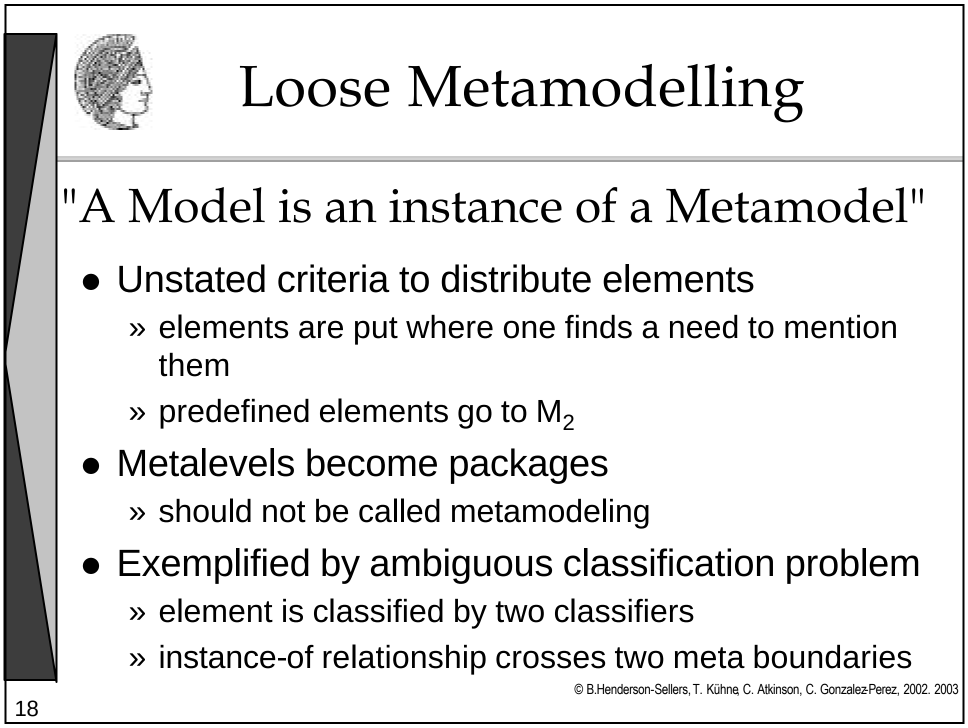


Ambiguous Classification

M2
Class
1
instance of
*
Object
1
M1
«instanceOf»
Dog
M0
«instanceOf»
fido
*
instance of

- Fido is classified by two types/classifiers
- Fido's instance-of relationship crosses two metalevel boundaries

17

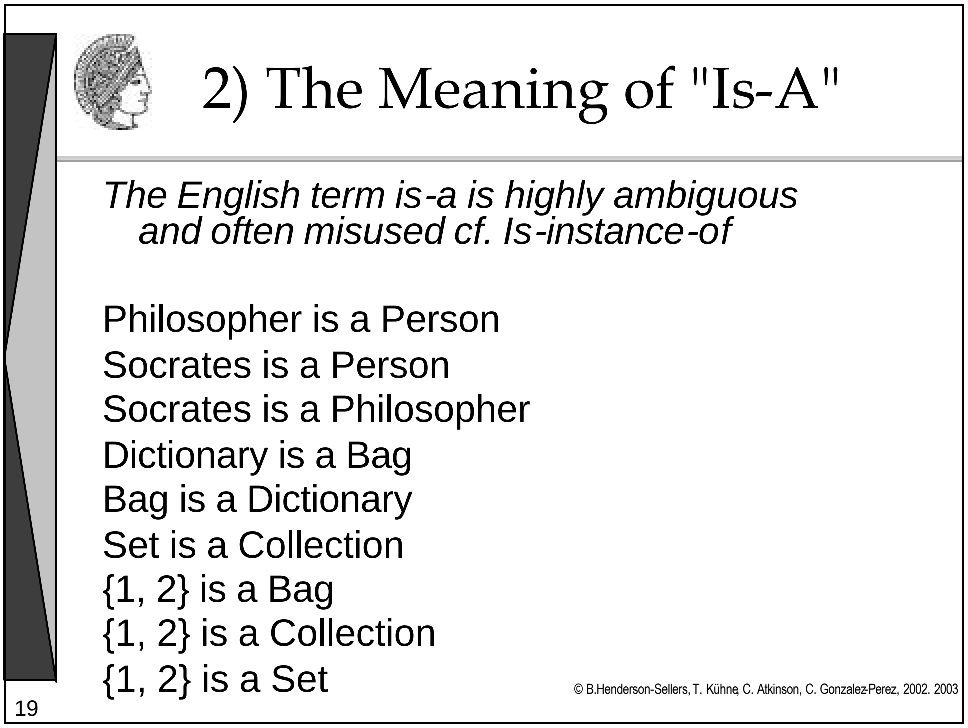


Loose Metamodelling

"A Model is an instance of a Metamodel"

- Unstated criteria to distribute elements
  - » elements are put where one finds a need to mention them
  - » predefined elements go to M_2
- Metalevels become packages
  - » should not be called metamodeling
- Exemplified by ambiguous classification problem
  - » element is classified by two classifiers
  - » instance-of relationship crosses two meta boundaries

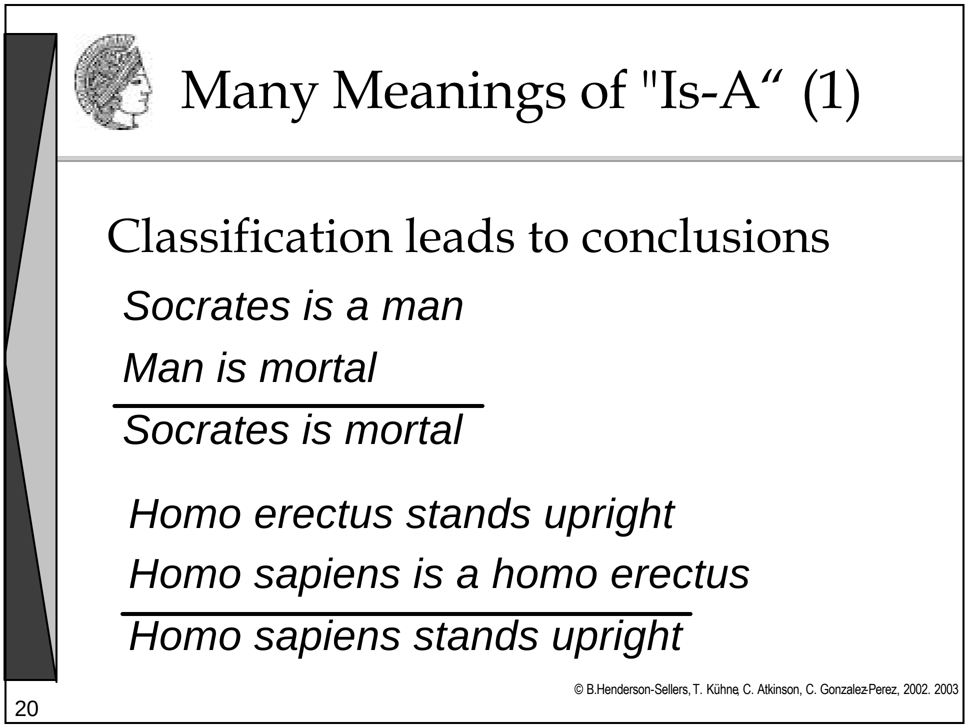
18



2) The Meaning of "Is-A"

The English term is-a is highly ambiguous and often misused cf. Is-instance-of

Philosopher is a Person
Socrates is a Person
Socrates is a Philosopher
Dictionary is a Bag
Bag is a Dictionary
Set is a Collection
{1, 2} is a Bag
{1, 2} is a Collection
{1, 2} is a Set



Many Meanings of "Is-A“ (1)

Classification leads to conclusions

Socrates is a man
Man is mortal
Socrates is mortal

Homo erectus stands upright
Homo sapiens is a homo erectus
Homo sapiens stands upright

Many Meanings of "Is-A“ (2)

Imprecision leads to errors

Socrates is a Man

Man is a Species

Socrates is a Species

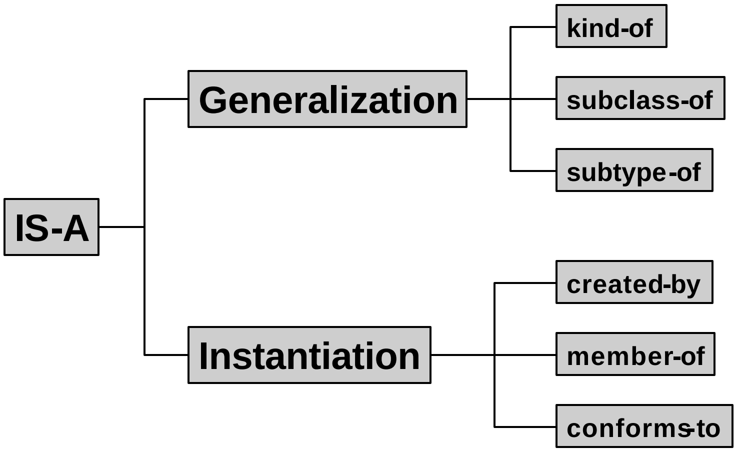
Corrected statements

Socrates is an instance-of Man

Man is an instance-of Species

Socrates is an instance-of a Species instance

Many Meanings of "Is-A“ (3)



So, rewriting …

Philosopher is a kind-of Person

Socrates is a member-of Person

Socrates is best-described-by Philosopher

created-by

Dictionary is a kind-of Bag

Bag is a subclass-of Dictionary

Set is a subtype-of Collection

{1, 2} is created-by Bag

{1, 2} is a member-of Collection

{1, 2} conforms-to Set

Precise "Is-A" relationships

Counteracting imprecision

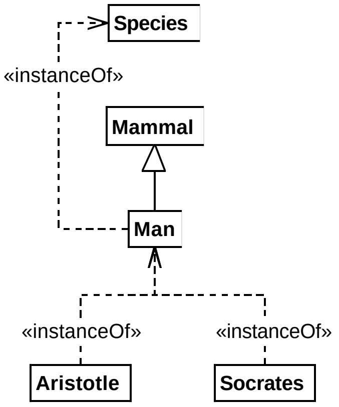
Socrates is a Man

Man is a Mammal

Man is a Species

Socrates is a Mammal

Socrates is not a Species



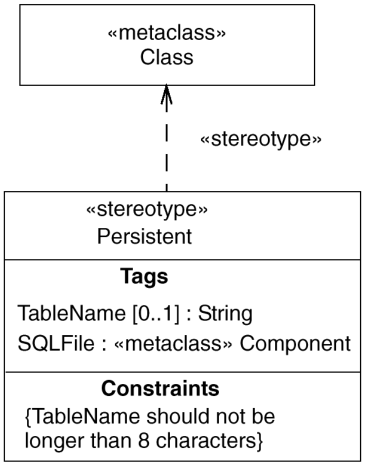
3) UML Extension Mechanisms

How to create your own version of the UML
while remaining within the OMG standard

- Constraints [not discussed further here]
- Stereotypes
  - » mainly for classification of classes,
- Tagged Values
  - » (immutable) class level attributes

Stereotypes

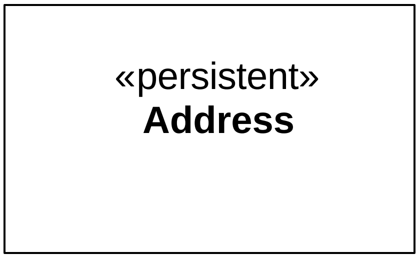
- Stereotypes are applied to a BaseClass
- They
  - » classify elements
  - » may introduce tags
  - » may define constraints



«metaclass»
Class
«stereotype»
«stereotype»
Persistent
Tags
TableName [0..1] : String
SQLFile : «metaclass» Component
Constraints
{TableName should not be longer than 8 characters}

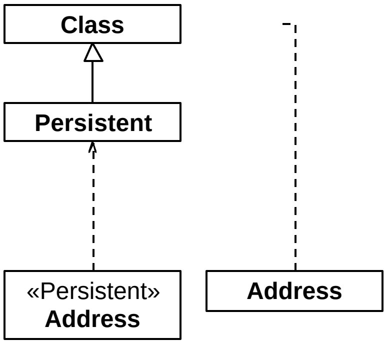
Application of stereotype

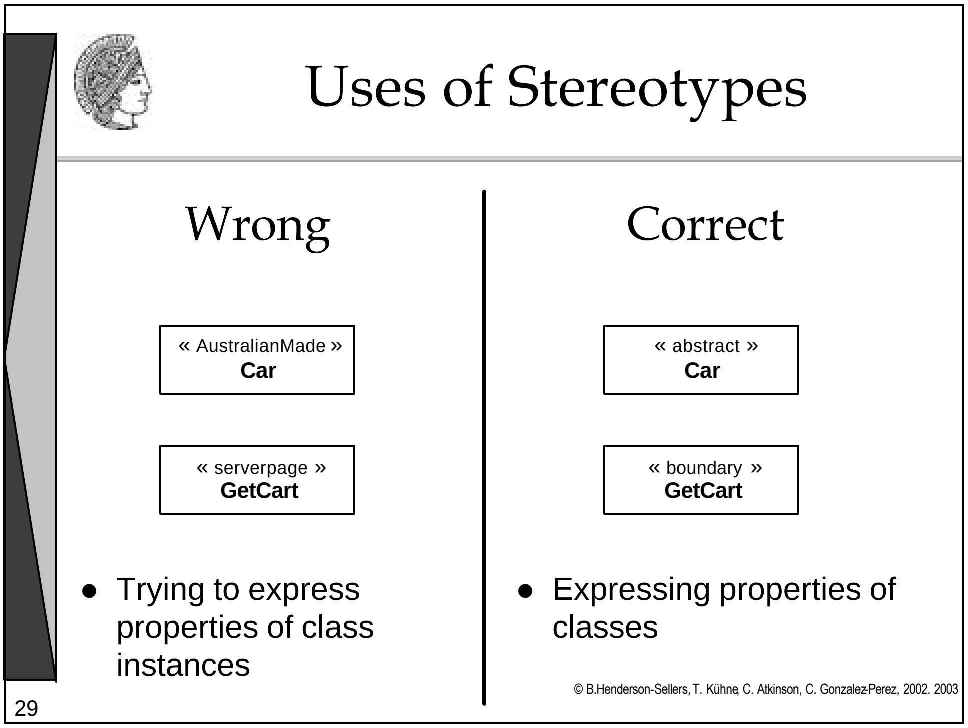
- After definition, this stereotype is then applied to model-level (M1) classes e.g.



Stereotypes

- Stereotypes effectively introduce specialized user-defined Metaclasses
  - » Meta-Subclasses define subsets of modelling elements
  - » Stereotypes can only influence modelling elements, not their instances





Uses of Stereotypes

Wrong

« AustralianMade »
Car

« serverpage »
GetCart

- Trying to express properties of class instances

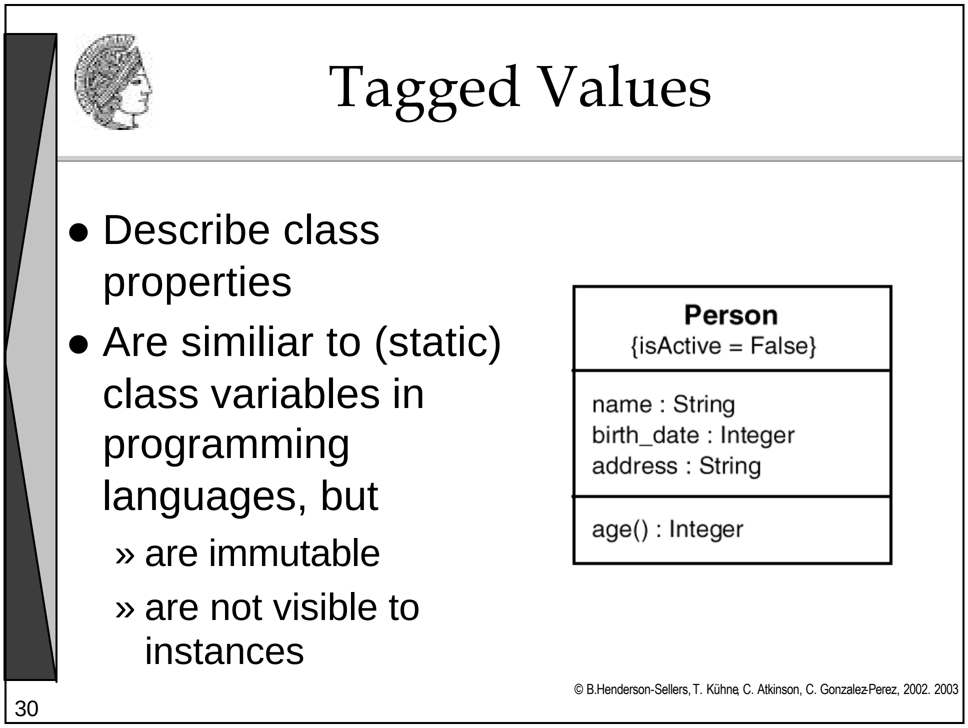
Correct

« abstract »
Car

« boundary »
GetCart

- Expressing properties of classes

29



Tagged Values

- Describe class properties
- Are similiar to (static) class variables in programming languages, but
  - » are immutable
  - » are not visible to instances

Person
{isActive = False}

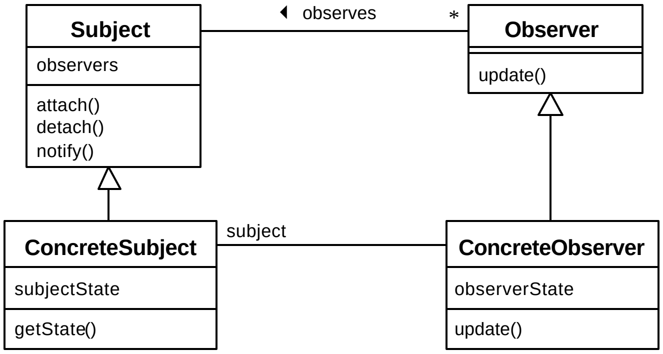
name : String
birth_date : Integer
address : String

age() : Integer

30

Where to predefine Properties? (1)

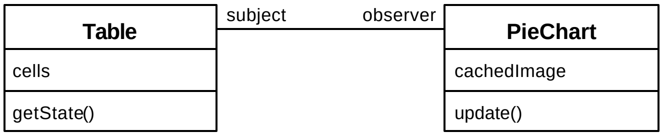
Consider first the Observer Pattern



Where to predefine Properties? (2)

An Example

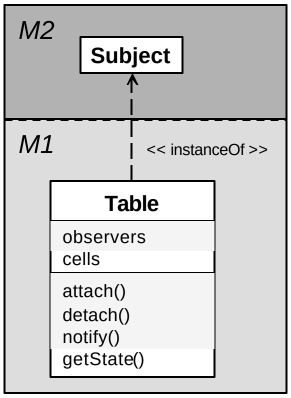
- What is the best way to let **Table** act as a subject and **PieChart** as an observer?



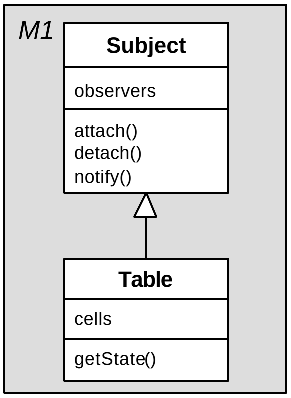
- Two ways of obtaining predefined properties:
  - » instantiation (metamodelling)
  - » inheritance (generalization)

Where to predefine Properties? (3)

Metamodelling

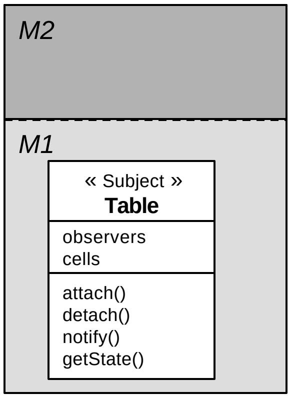


Inheritance

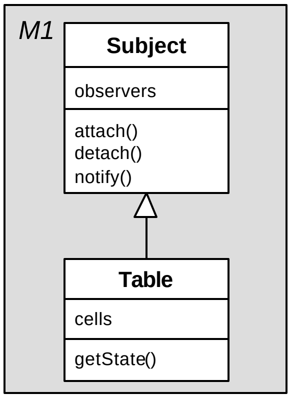


Changing the Notation

Metamodelling

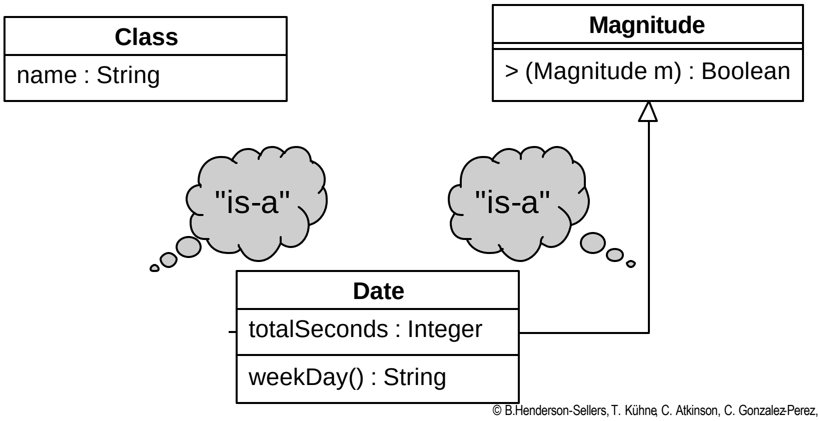


Inheritance



Classification v Generalization

There are two (different!) forms of derivation. Both may be called "is-a“.



Class

name : String

Magnitude

> (Magnitude m) : Boolean

"is-a"

"is-a"

Date

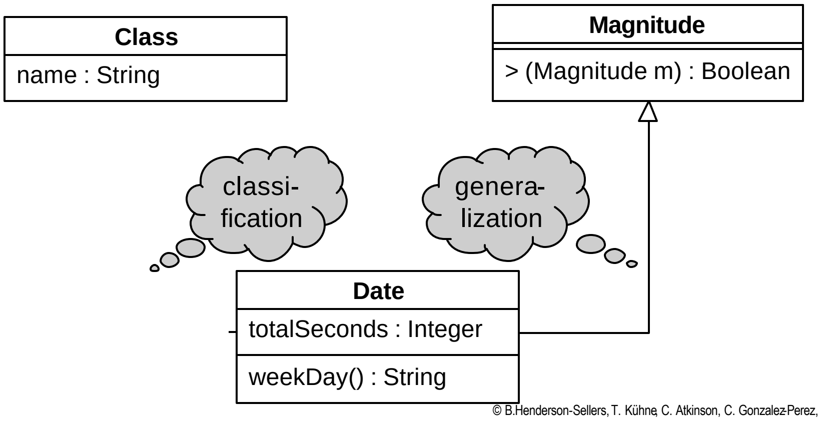
totalSeconds : Integer

weekDay() : String

35

Classification v Generalization

Renaming "is-a“ to underline the difference



Class

name : String

Magnitude

> (Magnitude m) : Boolean

classification

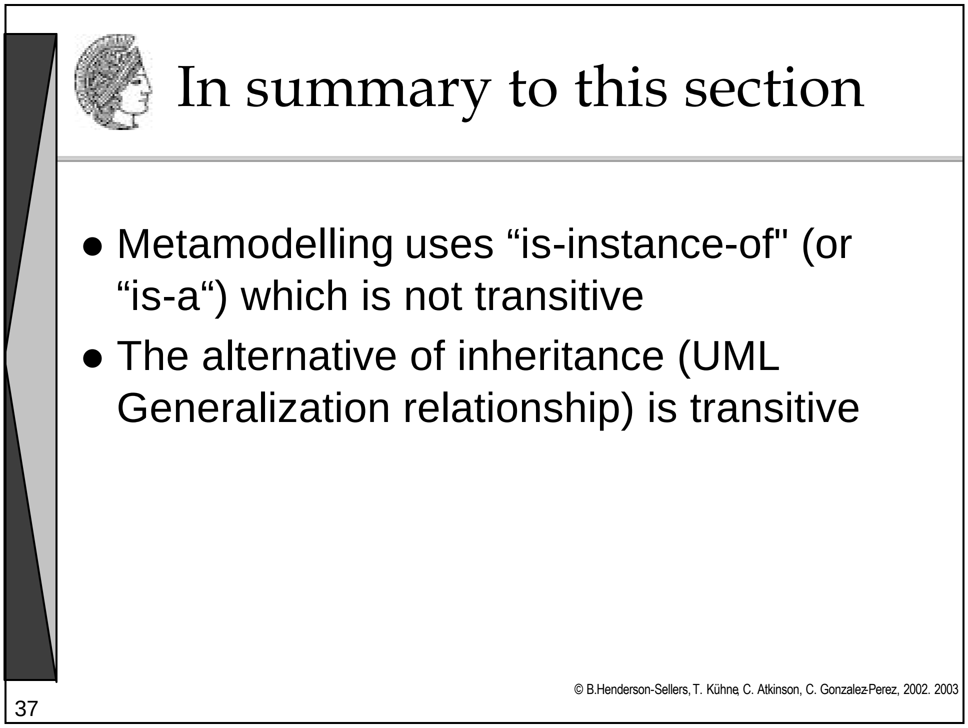
generalization

Date

totalSeconds : Integer

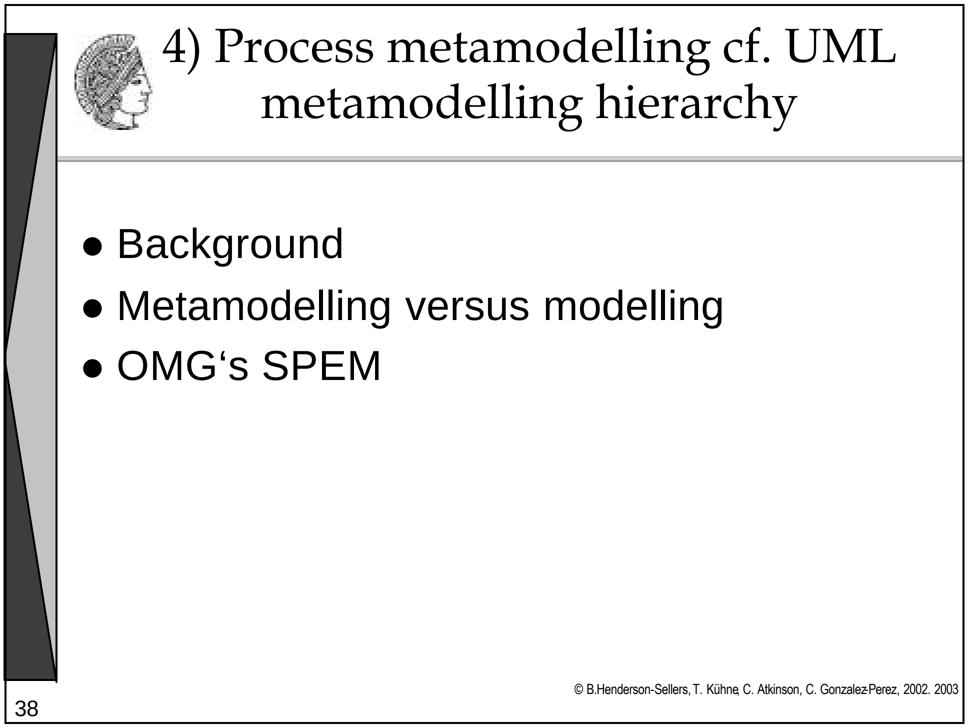
weekDay() : String

36



In summary to this section

- Metamodelling uses “is-instance-of" (or “is-a“) which is not transitive
- The alternative of inheritance (UML Generalization relationship) is transitive



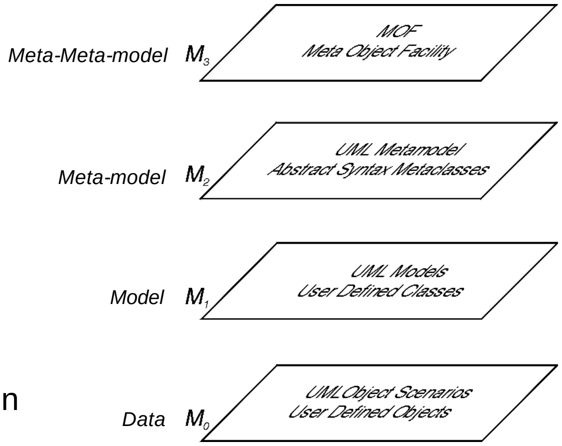
4) Process metamodelling cf. UML metamodelling hierarchy

- Background
- Metamodelling versus modelling
- OMG‘s SPEM

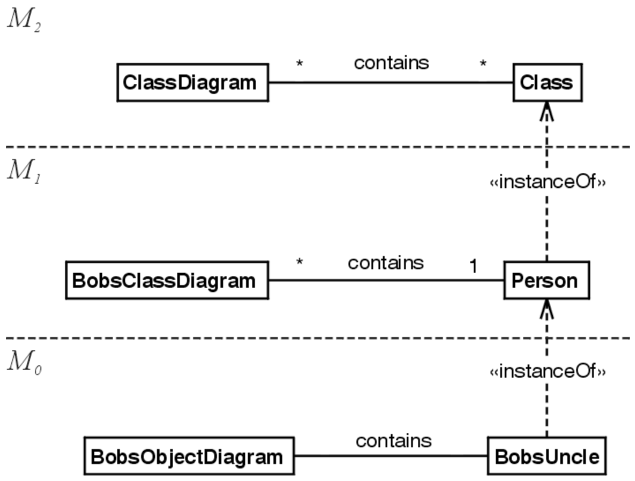
Background: The 4-Layer Hierarchy

Problems

- Where to put process elements within this hierarchy?
- What relationships should be allowed between metalevels?

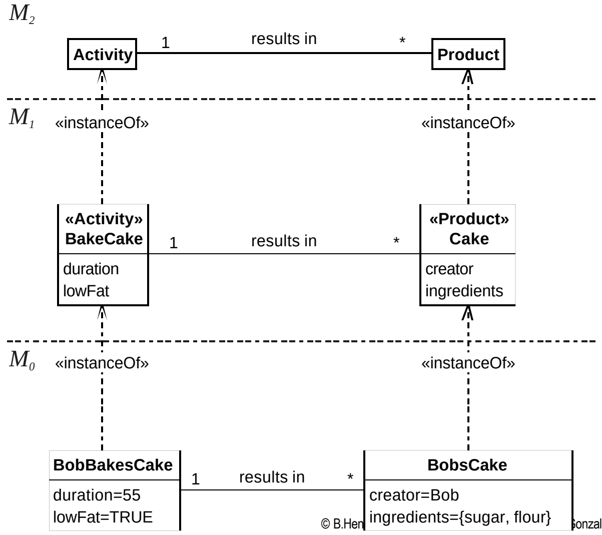


UML Artefact Description

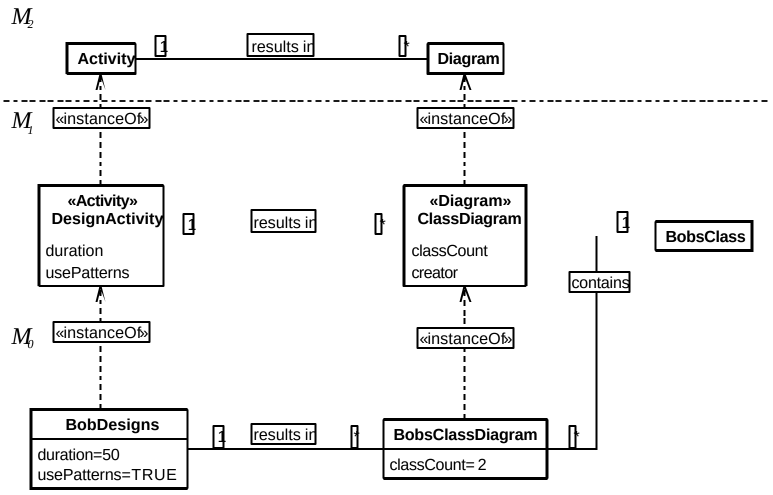


Process Description

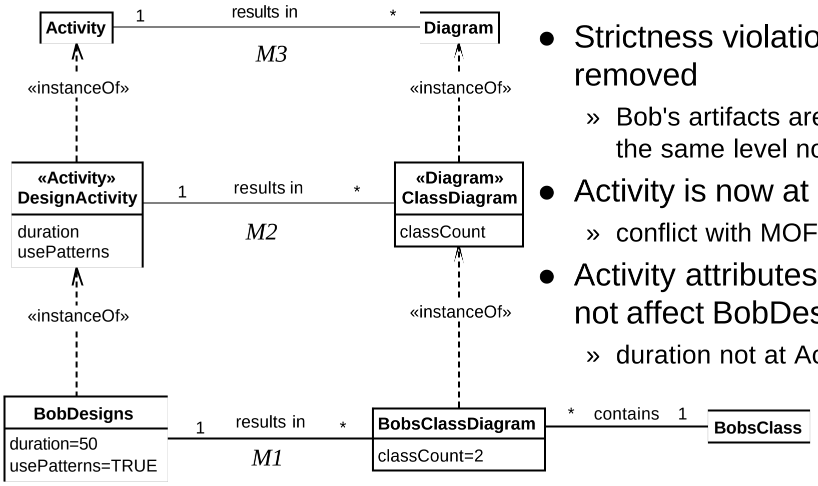
Activities have a duration



Moving to Software Domain



Shifting the Hierarchy Up



Activity 1 results in * Diagram

M3

«instanceOf» «instanceOf»

«Activity» DesignActivity | duration, usePatterns — 1 results in * — «Diagram» ClassDiagram | classCount

M2

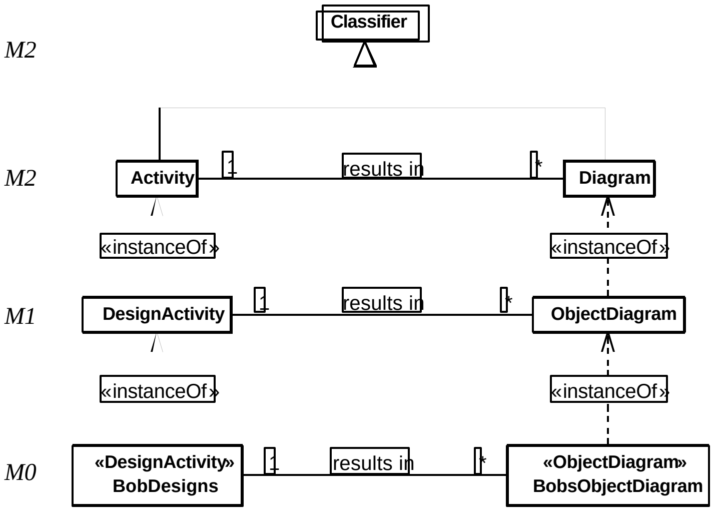
«instanceOf» «instanceOf»

BobDesigns | duration=50, usePatterns=TRUE — 1 results in * — BobsClassDiagram | classCount=2 — * contains 1 — BobsClass

M1

- Strictness violation removed
  - » Bob's artifacts are at the same level now
- Activity is now at M_3
  - » conflict with MOF
- Activity attributes do not affect BobDesigns
  - » duration not at Activity

Metamodelling vs Modelling (1)



M2 Classifier

M2 Activity 1 results in * Diagram

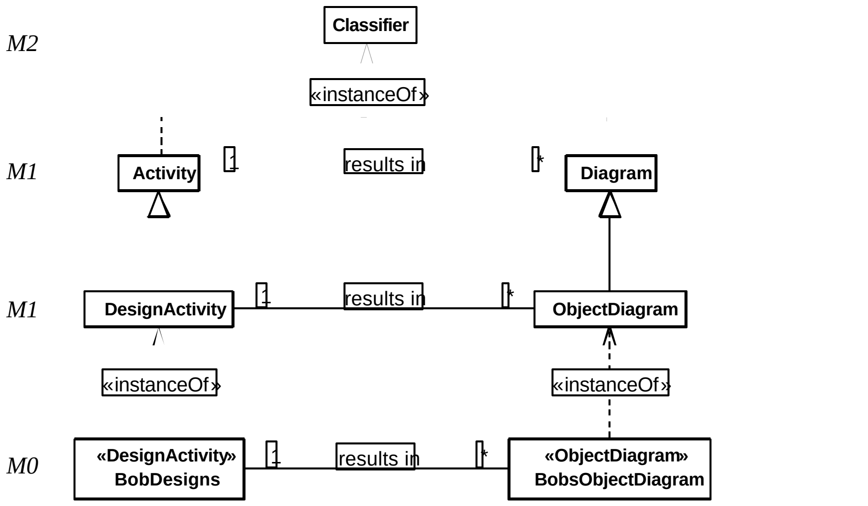
«instanceOf» «instanceOf»

M1 DesignActivity 1 results in * ObjectDiagram

«instanceOf» «instanceOf»

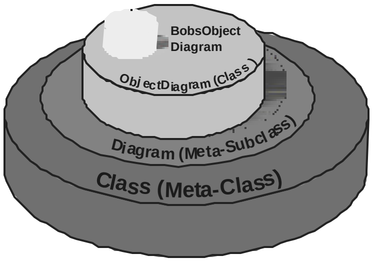
M0 «DesignActivity» BobDesigns 1 results in * «ObjectDiagram» BobsObjectDiagram

Metamodelling vs **Modelling** (2)

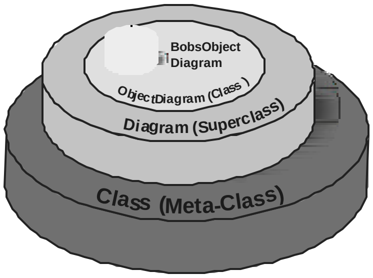


Metamodelling vs Modelling (3)

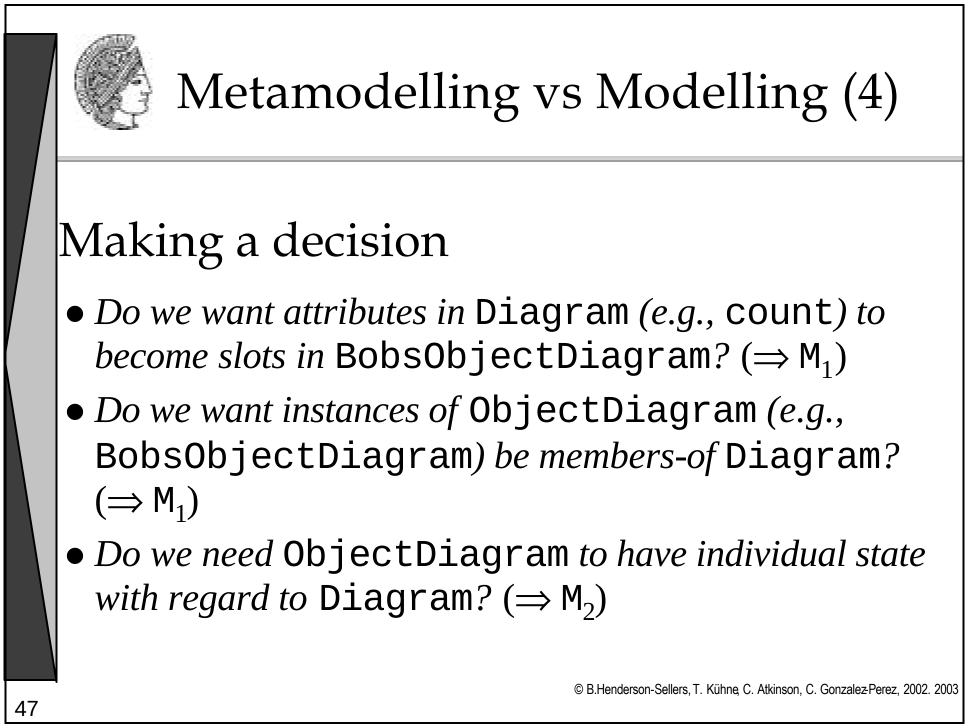
Instantiation (M_2/M_1)



Inheritance (M_1)



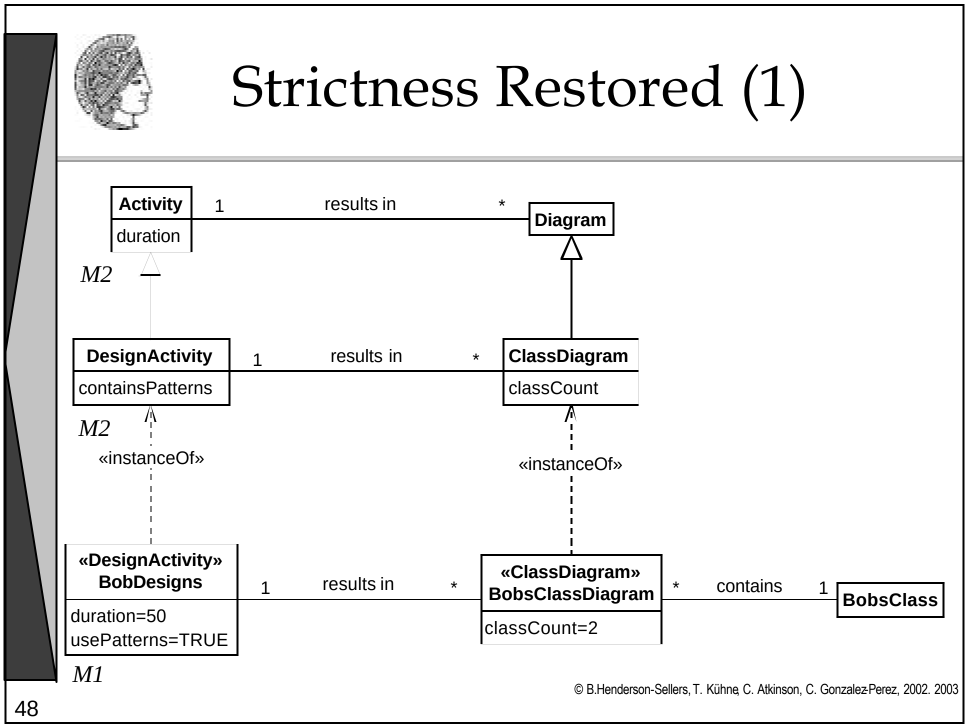
- ObjectDiagram is an instance-of Diagram
- ObjectDiagram is a subclass of Diagram



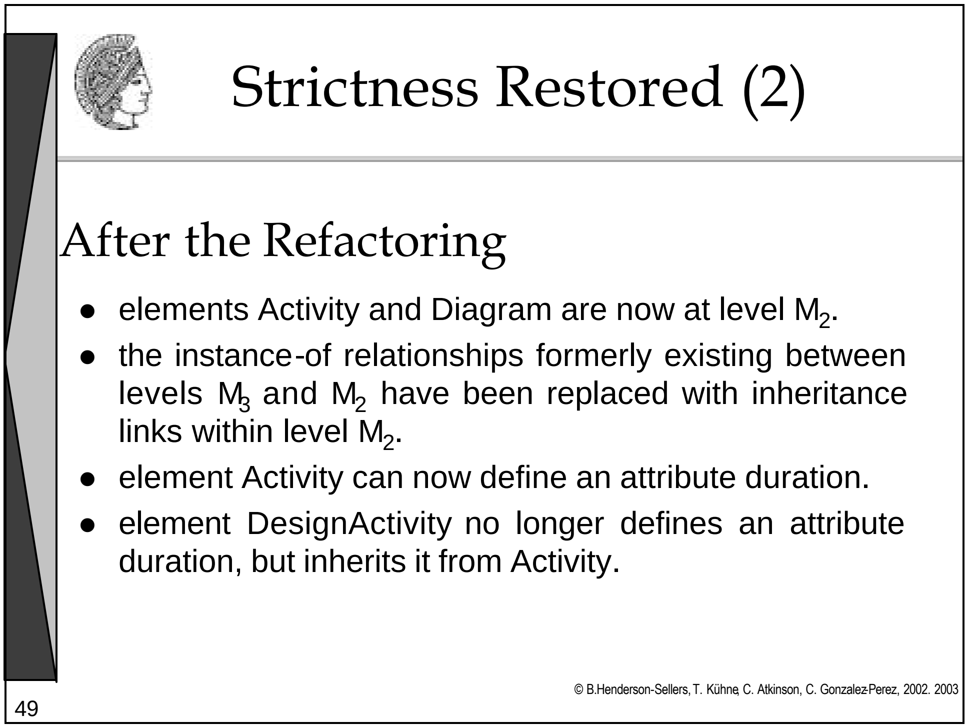
Metamodelling vs Modelling (4)

Making a decision

- *Do we want attributes in* `Diagram` *(e.g.,* `count`*) to become slots in* `BobsObjectDiagram`*?* (⇒ $\mathtt{M}_1$)
- *Do we want instances of* `ObjectDiagram` *(e.g.,* `BobsObjectDiagram`*) be members-of* `Diagram`*?* (⇒ $\mathtt{M}_1$)
- *Do we need* `ObjectDiagram` *to have individual state with regard to* `Diagram`*?* (⇒ $\mathtt{M}_2$)



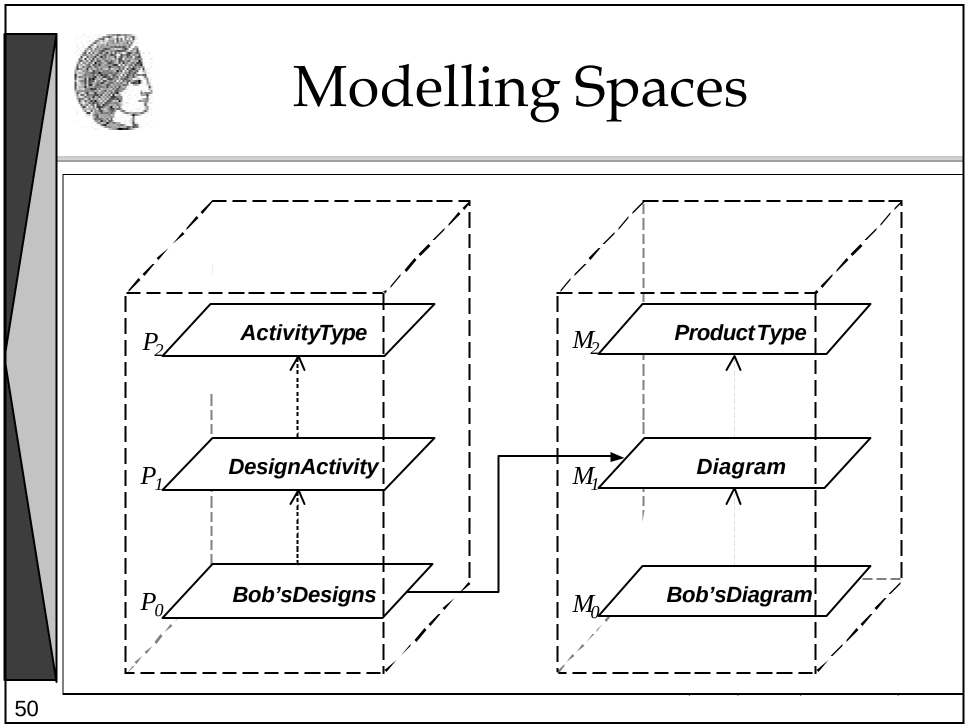
Strictness Restored (1)



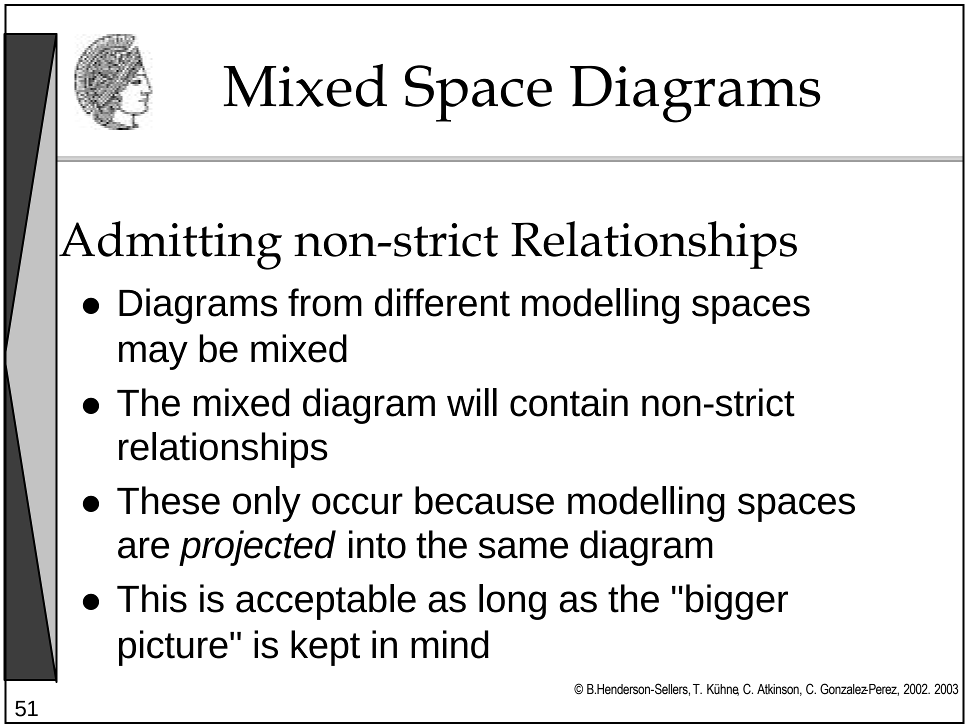
Strictness Restored (2)

After the Refactoring

- elements Activity and Diagram are now at level M_2.
- the instance-of relationships formerly existing between levels M_3 and M_2 have been replaced with inheritance links within level M_2.
- element Activity can now define an attribute duration.
- element DesignActivity no longer defines an attribute duration, but inherits it from Activity.



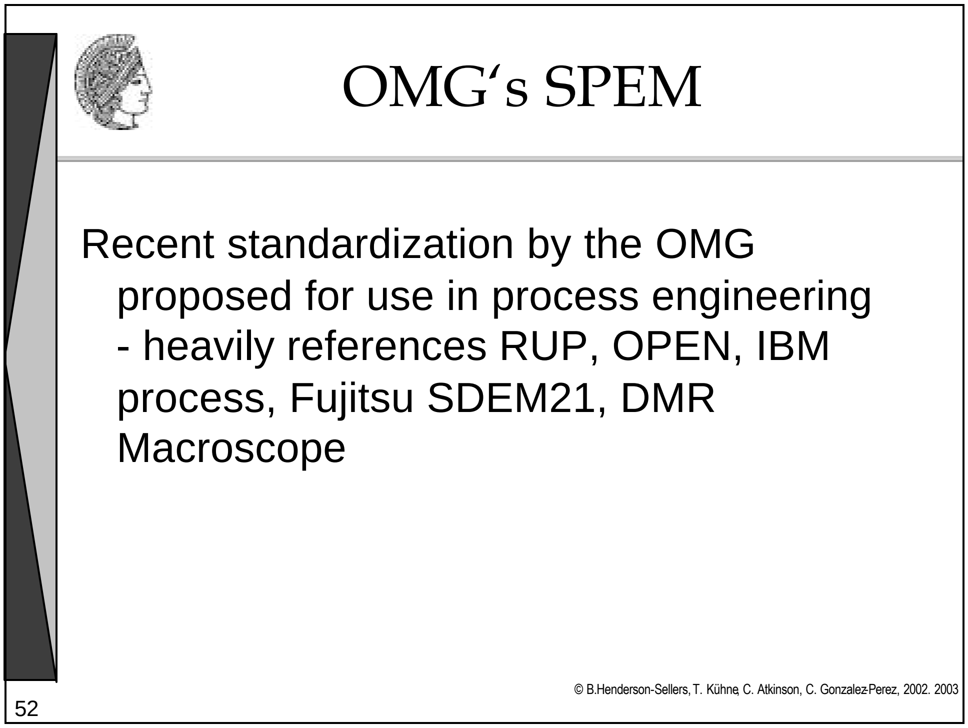
Modelling Spaces



Mixed Space Diagrams

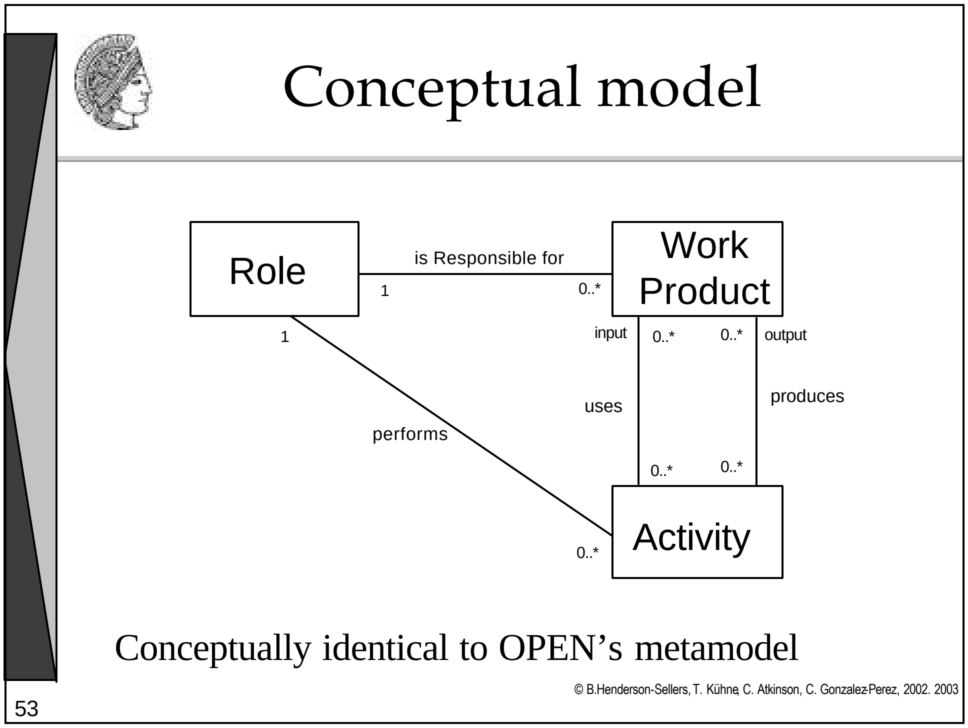
Admitting non-strict Relationships

- Diagrams from different modelling spaces may be mixed
- The mixed diagram will contain non-strict relationships
- These only occur because modelling spaces are *projected* into the same diagram
- This is acceptable as long as the "bigger picture" is kept in mind



OMG‘s SPEM

Recent standardization by the OMG proposed for use in process engineering - heavily references RUP, OPEN, IBM process, Fujitsu SDEM21, DMR Macroscope



Conceptual model

Role

is Responsible for

1

0..*

Work Product

1

input

0..*

0..*

output

uses

produces

performs

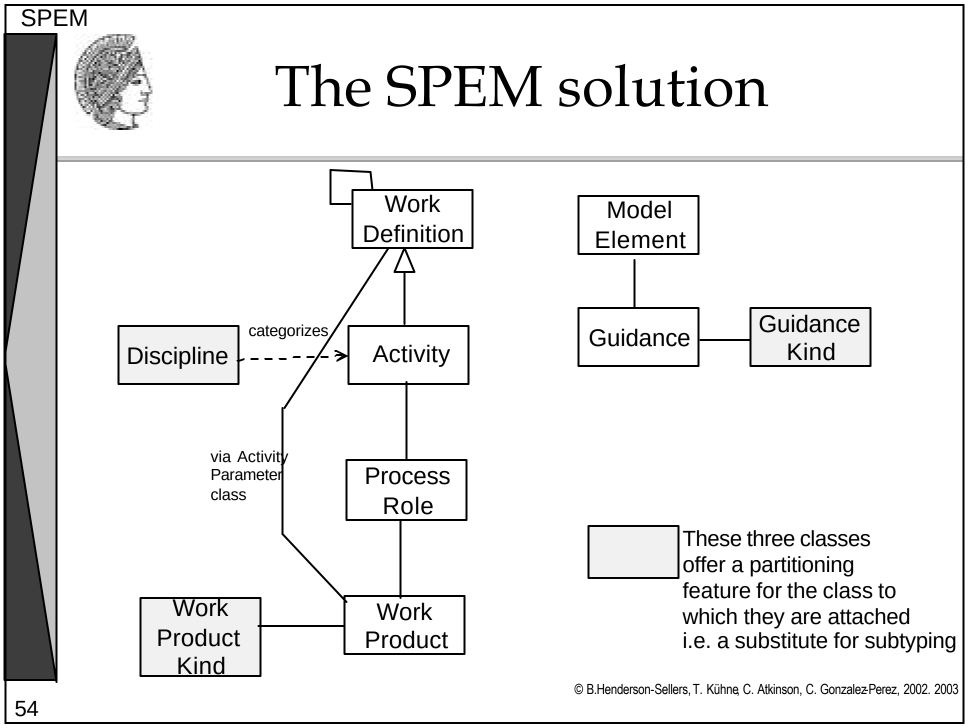
0..*

0..*

Activity

0..*

Conceptually identical to OPEN’s metamodel



SPEM

The SPEM solution

Work Definition

Model Element

Discipline

categorizes

Activity

Guidance

Guidance Kind

via Activity Parameter class

Process Role

These three classes offer a partitioning feature for the class to which they are attached i.e. a substitute for subtyping

Work Product Kind

Work Product

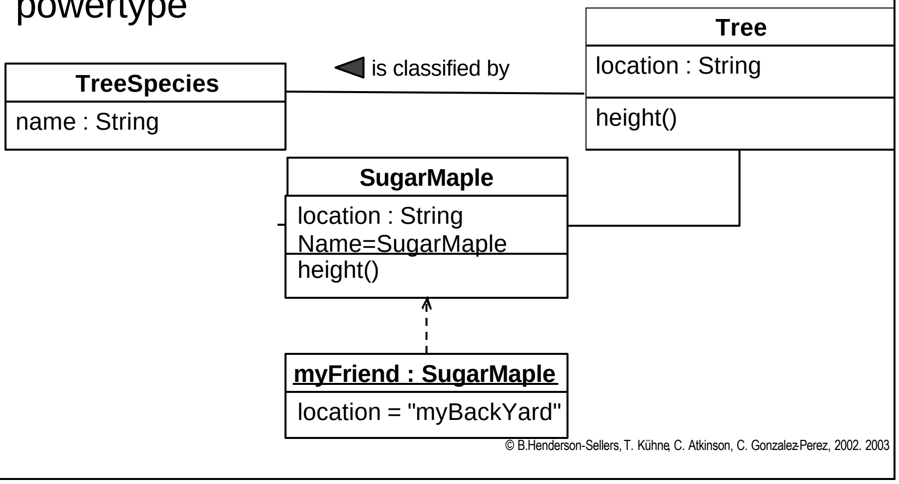
Newer solution: powertypes

- A powertype has instances which are subclasses of another type
- Seen examples already in Socrates (slide 20) and Date (slide 32)
- Powertypes do not fit into the strict metamodelling hierarchy

55

Another Example

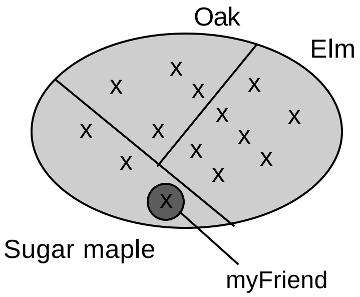
TreeSpecies is a powertype



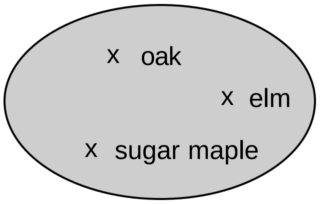
56

In terms of sets

Tree class

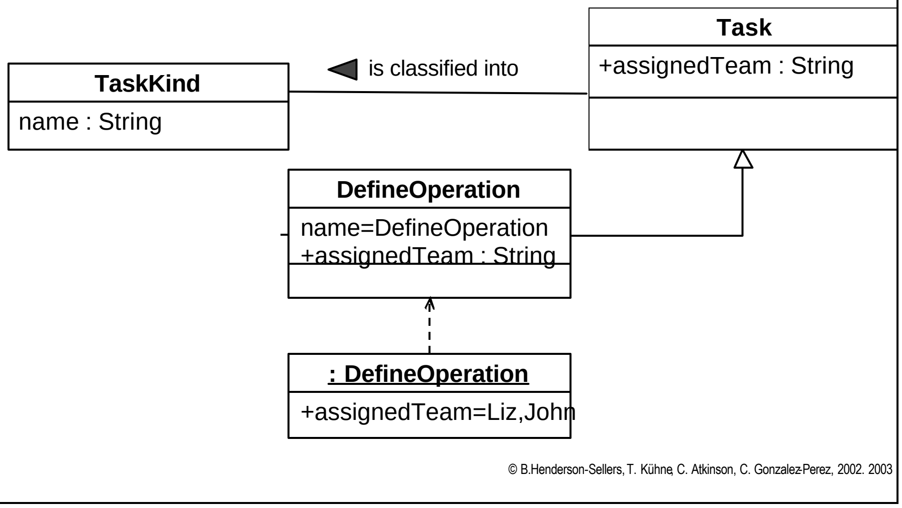


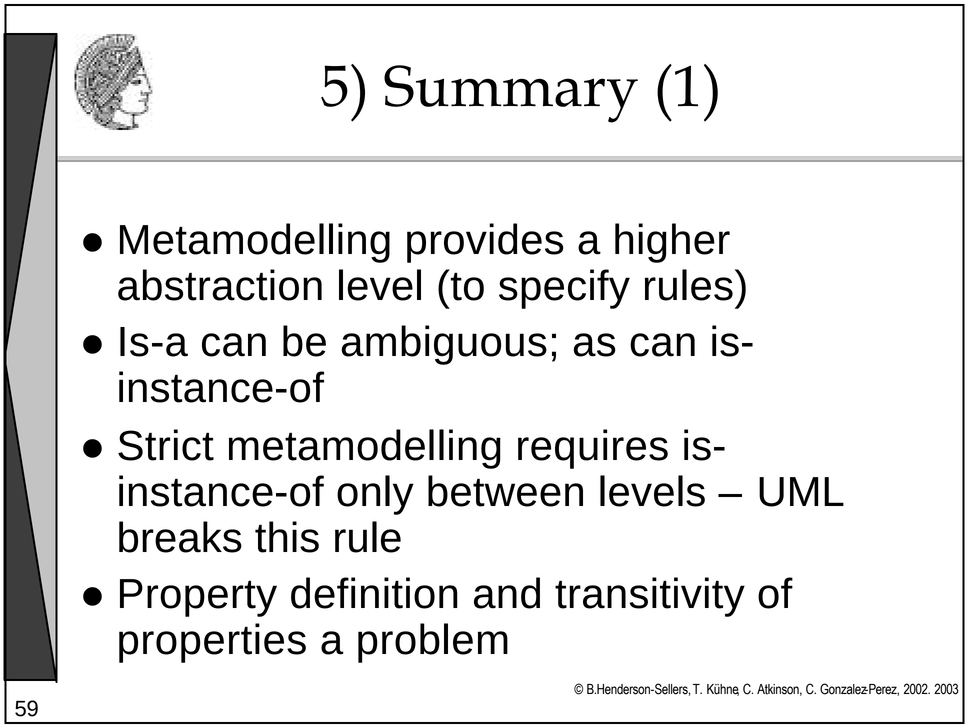
TreeSpecies class



TreeSpecies is a Powertype i.e. a set of all subsets of another set as defined by a given discriminator

An Example in Process Modelling

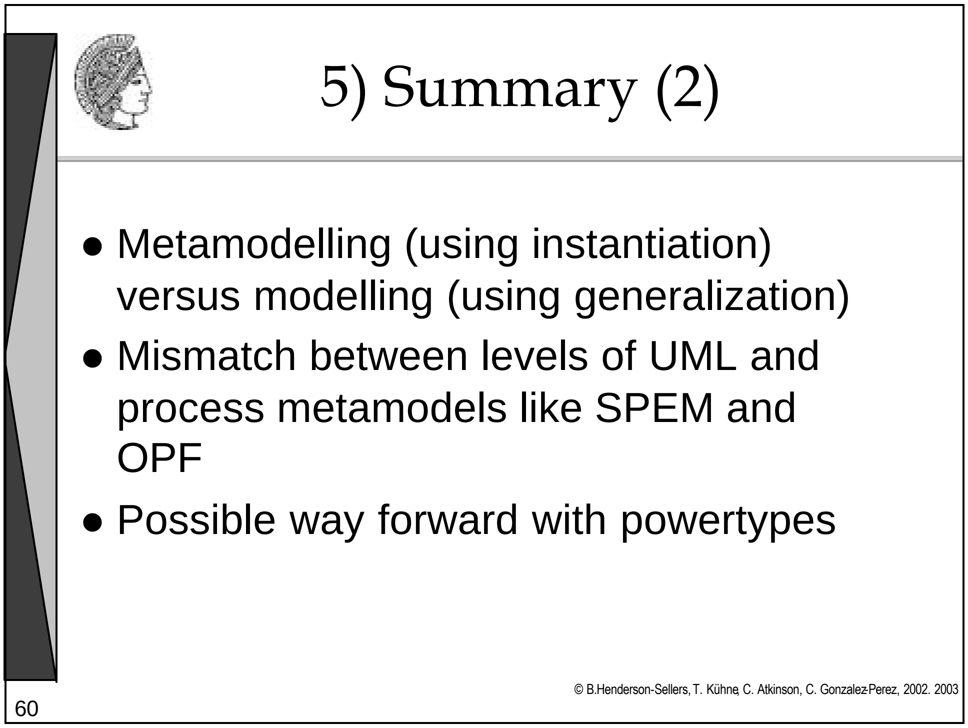




5) Summary (1)

- Metamodelling provides a higher abstraction level (to specify rules)
- Is-a can be ambiguous; as can is-instance-of
- Strict metamodelling requires is-instance-of only between levels – UML breaks this rule
- Property definition and transitivity of properties a problem

59



5) Summary (2)

- Metamodelling (using instantiation) versus modelling (using generalization)
- Mismatch between levels of UML and process metamodels like SPEM and OPF
- Possible way forward with powertypes

60