The Role of Foundational Ontologies in Deep Modeling

Colin Atkinson

MODELSWARD 2014
Keynote Lecture
January 8th, 2014

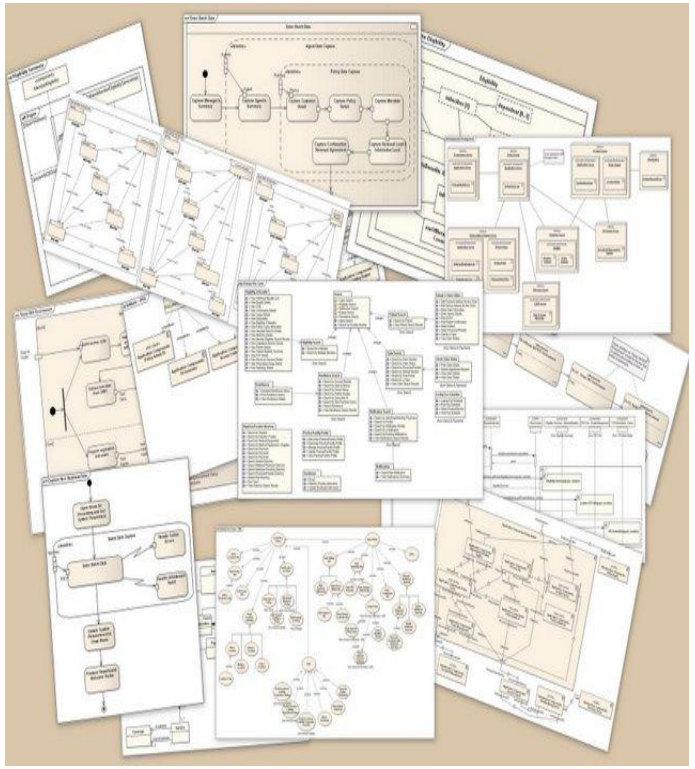
Software Engineering Group

UNIVERSITÄT MANNHEIM

Modeling Everywhere ...

- Modeling is a now key activity in virtually all IT projects
  - today is performed in may different languages

- General-purpose structural languages
  - UML, OWL, ERA, RDF ...
- General-purpose behavioural languages
  - UML activity diagrams, Petri nets, ...
- Associated textual languages
  - OCL, ATL, QVT, Xtext
- Domain specific language
  - ……



Wikipedia 2012

Basic Tension in Language Design

- Since models are used to communicate properties of the real world between humans they should be able to -
  - accurately and unambiguously represent the real world (representational adequacy [GGO5])
  - at any desired level of detail
  - in the simplest and most concise way possible (lucidity, ontological clarity, construct redundancy [GGO5])
- There is a basic tension between expressive power and simplicity
  - rich, expressive languages often complex and verbose
  - simple, concise languages often lack expressive power
- Related to the question
  - General-purpose versus domain specific?

UML 1.x Four Layer Model Architecture

Layer	Description	Example	
meta-metamodel	The infrastructure for a metamodeling architecture. Defines the language for specifying metamodels.	*MetaClass, MetaAttribute, MetaOperation*	M_3
metamodel	An instance of a meta-metamodel. Defines the language for specifying a model.	*Class, Attribute, Operation, Component*	M_2
model	An instance of a metamodel. Defines a language to describe an information domain.	*StockShare, askPrice, sellLimitOrder, StockQuoteServer*	M_1
user objects (user data)	An instance of a model. Defines a specific information domain.	*<Acme_Software_Share_98789>, 654.56, sell_limit_order, <Stock_Quote_Svr_32123>*	M_0

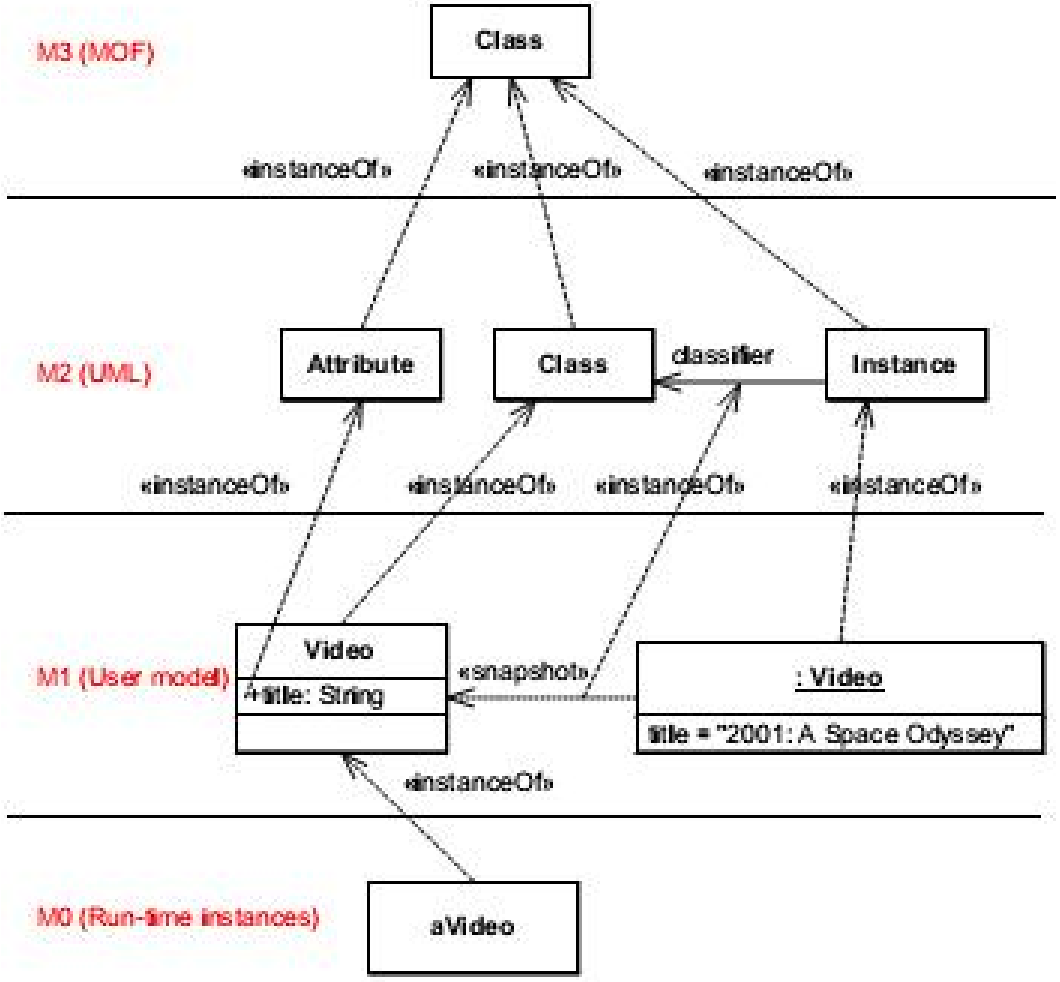
(from UML Semantics Version 1.1)

Strict Metamodeling (Classic Definition)

In an n-level modeling 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).

- every element has exactly one type, except those elements in the top level
- instance-of relationship used to define levels
- model elements allocated to levels according to their location in the type hierarchy

UML 2.x Infrastructure

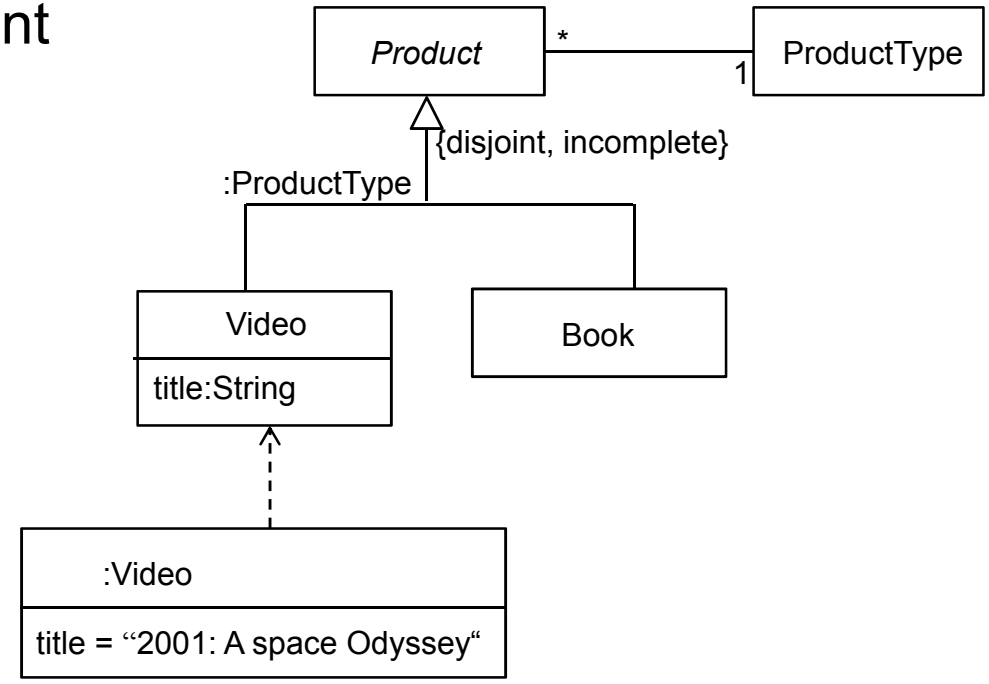


(from UML Infrastructure Specification v2.4.1)

- M_0 level contains the real-world elements (the subject of the model)
- M_1 level contains model representation of types and instances
- M_2 level contains linguistic types of the M_1 elements
- M_3 level contains the linguistic types of the M_2 level concepts

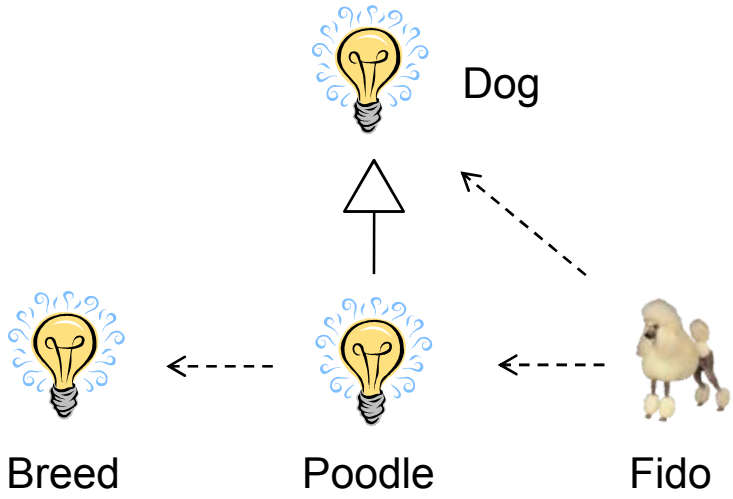
The Powertype Problem

- X is a power type of Y, if the instances of X are subclasses of Y
  - power types are classes whose instances are also subclasses
- therefore, according to the strict modeling tenet, powertypes must be at the M2 level
  - but they are represent domain concepts
  - the *isPowerTypeOf* relationship crosses level boundaries
- occur very frequently in practice



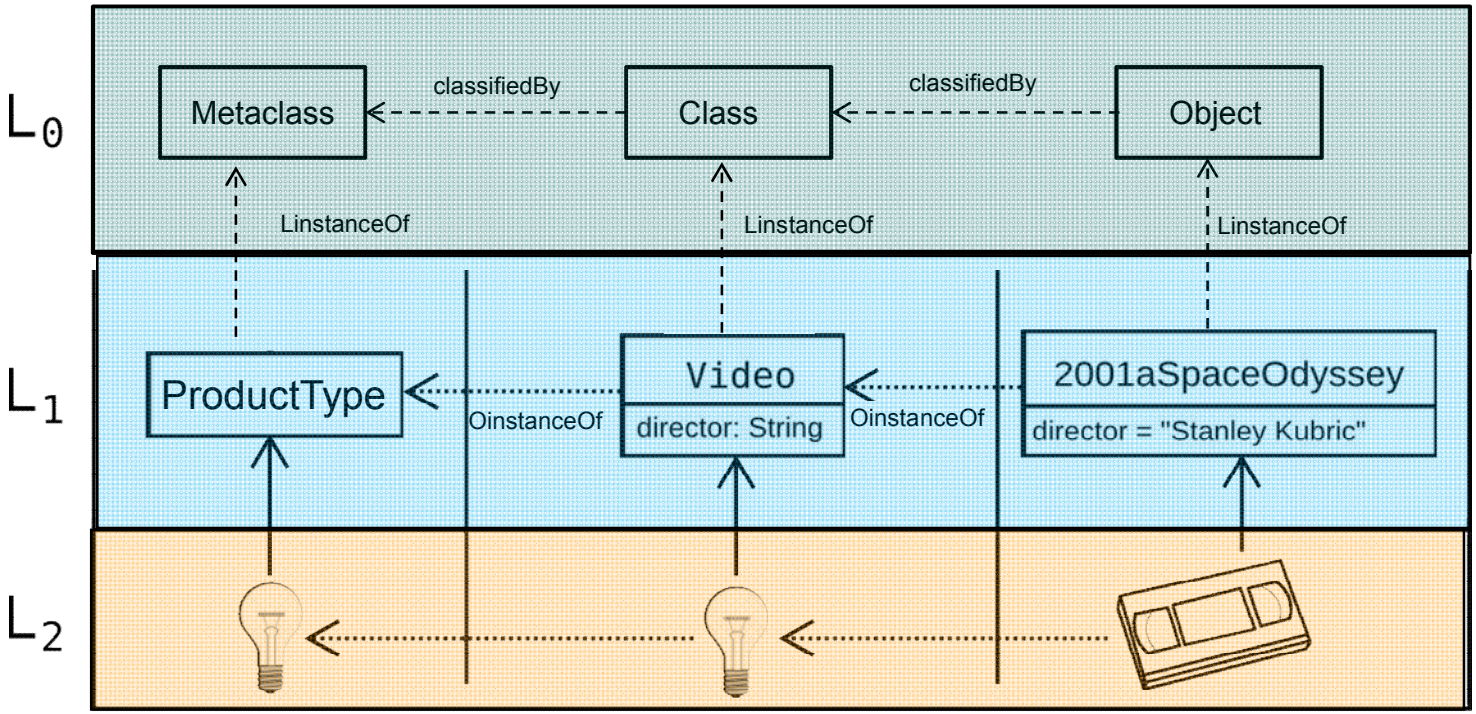
Deep Characterization

- usually a type only describes its instances
  - e.g., every dog has an age
- sometimes, however, a type needs to define constraints on the properties of the instances of its instances
  - e.g. every instance, of a breed instance, has an age
- such transitive influences cannot be directly expressed using traditional instanceOf relationships
- with UML must resort to -
  - constraints
  - powertypes
  - stereotypes
  - …..



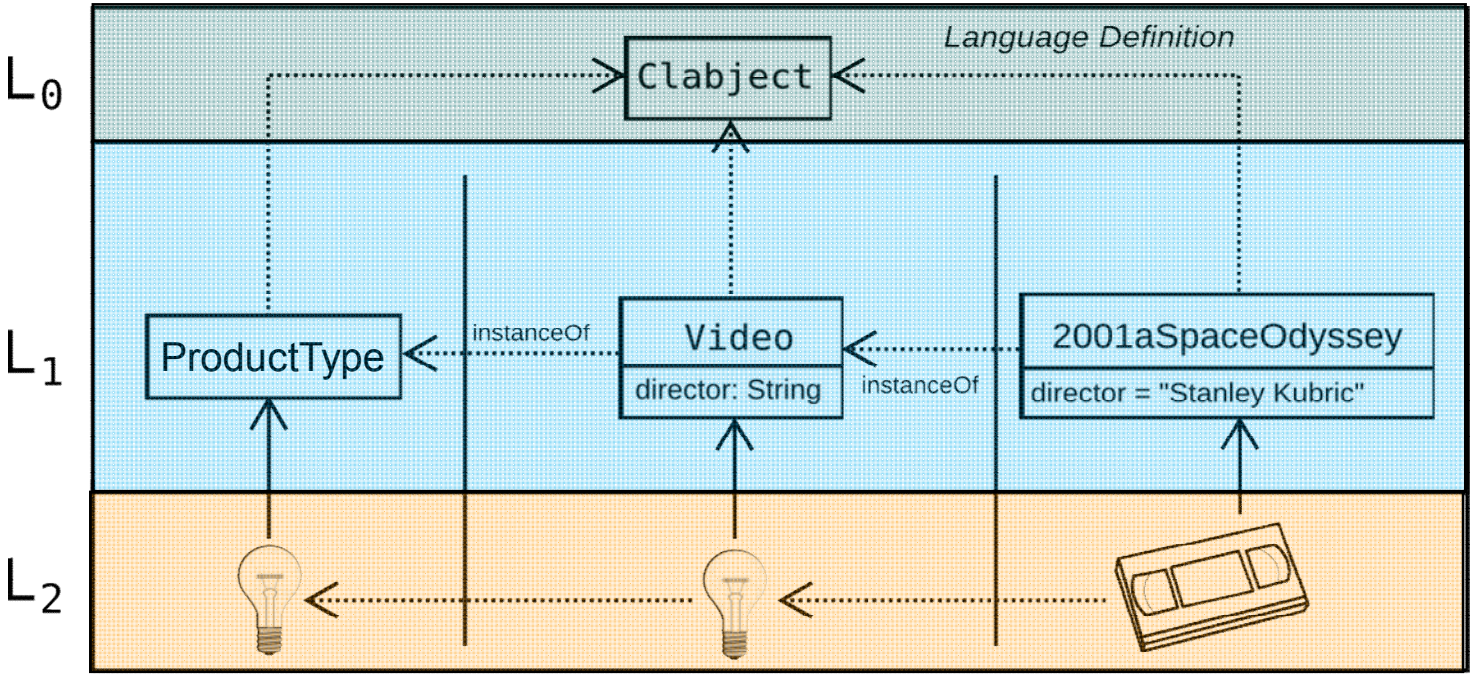
Orthogonal Classification Architecture

- two distinct forms of classification organized in different, dimensions
  - linguistic and ontological
- Strict (meta)-modeling in each dimension
- **So called "linguistic/ontological metamodeling paradox" [EHA13]**

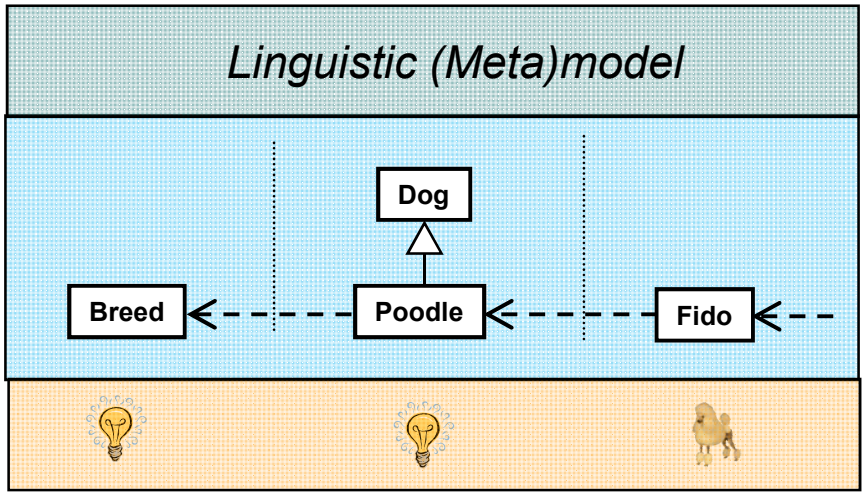


Deep (Multi-Level)Modeling

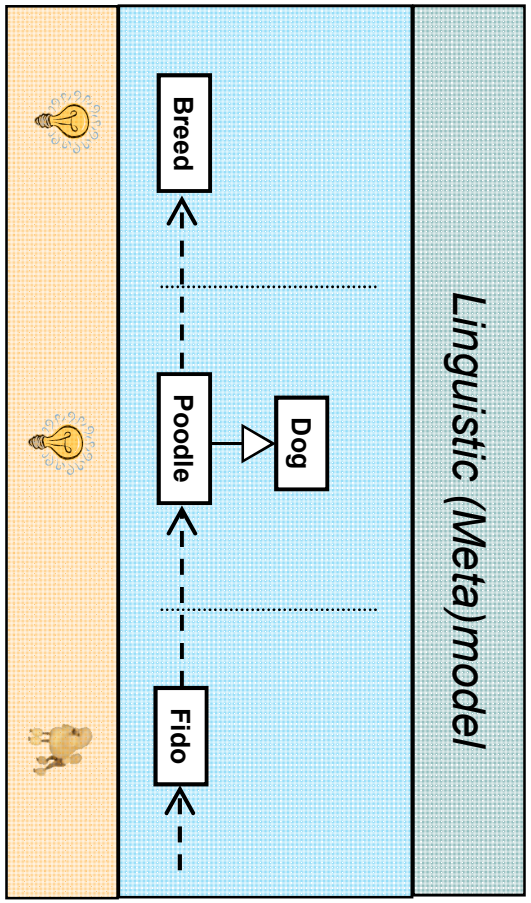
- Orthogonal classification architecture
- Unified class/object model element
  - Clabject
- Level-agnostic mechanisms for representing “typeness” of clabjects
  - Potency (deep instantiation)



Different Views of the Infrastructure



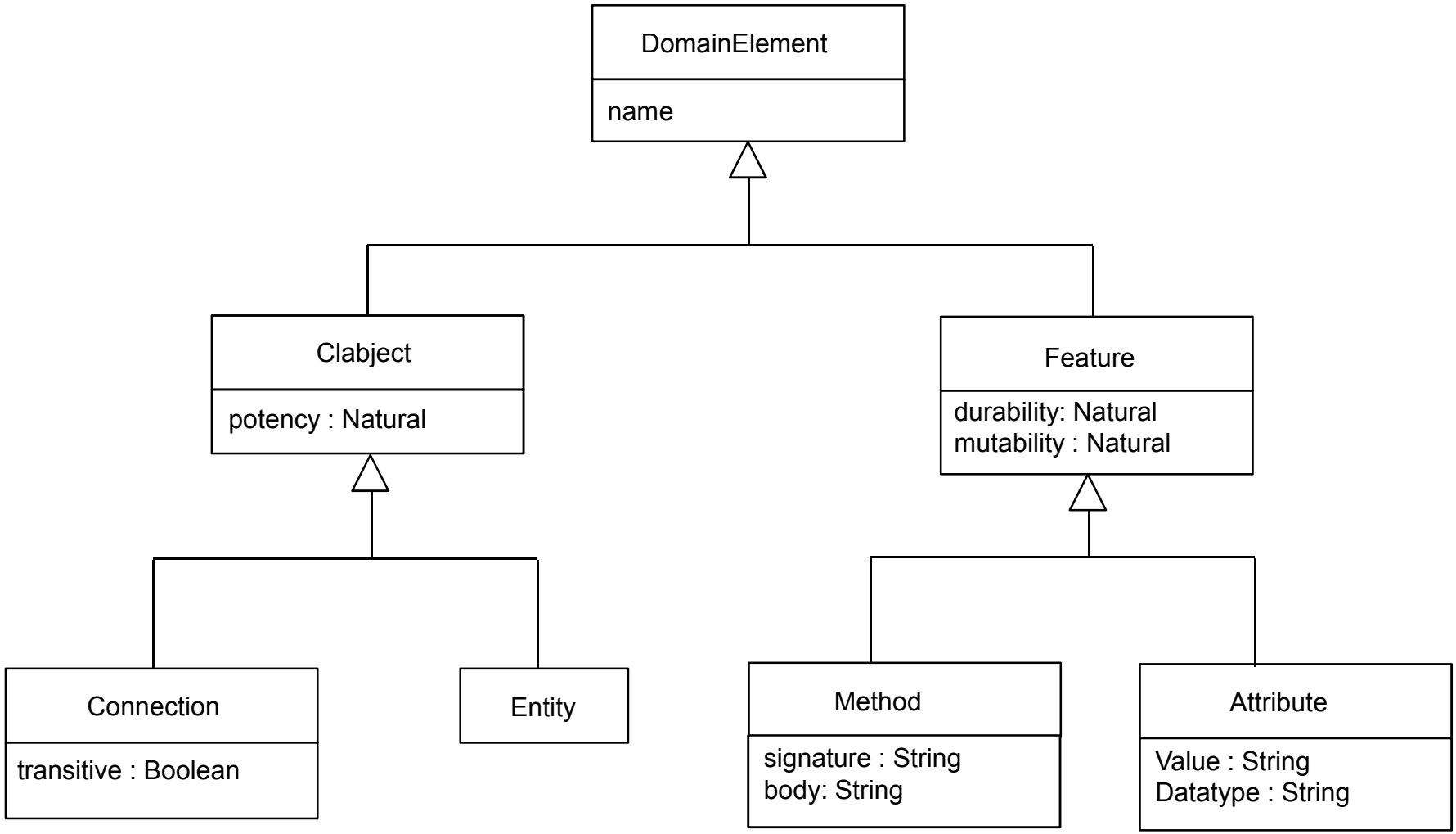
90⁰



Tool Developer's View

Modeler's View

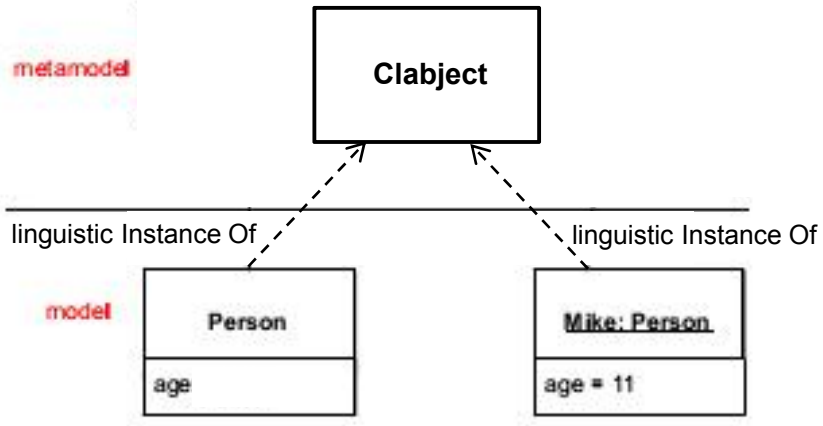
Essence of a Deep Linguistic Metamodel



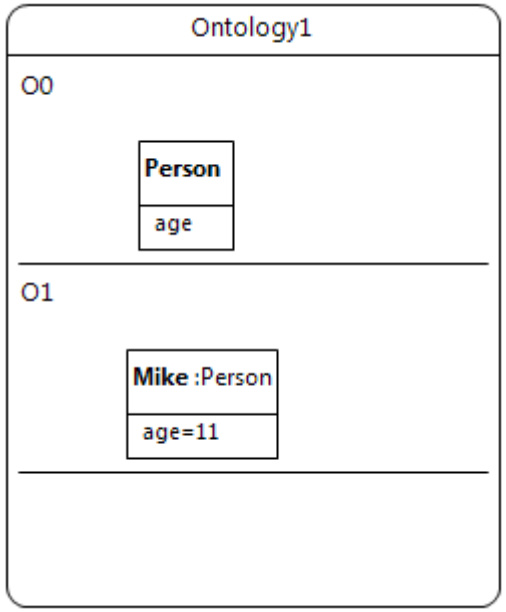
- abstract syntax for a Level-Agnostic Modeling language (LML)

Classes and Objects

- In LML classes and objects are unfied into the notion of “clabjects“
  - clabjects can have ontological attributes that play the role of UML attributes / slots
- ontological attributes must always have a name
  - can also, optionally, have a type and /or a value



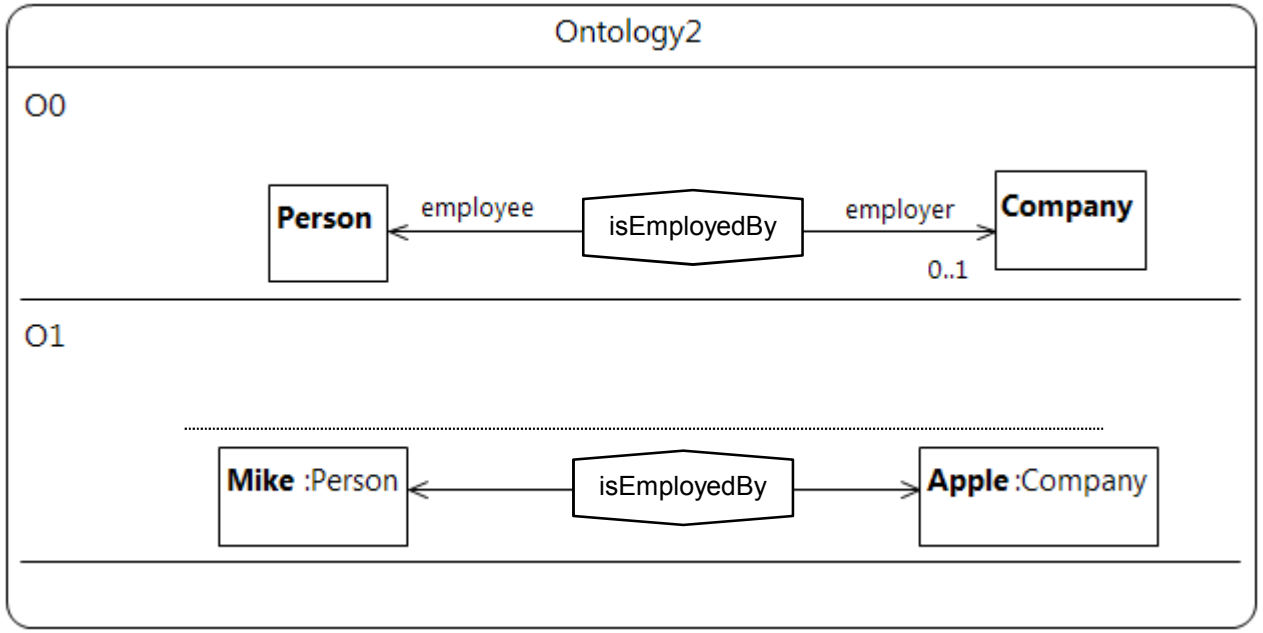
UML



LML

Associations and Links

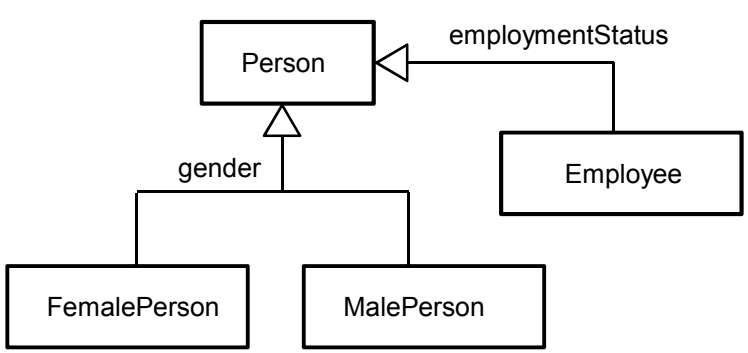
- Associations / links are also represented by a unified concept in LML – Connections
- conection are also clabjects
  - can have attributes and particpate in generalizations
  - can be visualized in an exploded or “dotted“ form



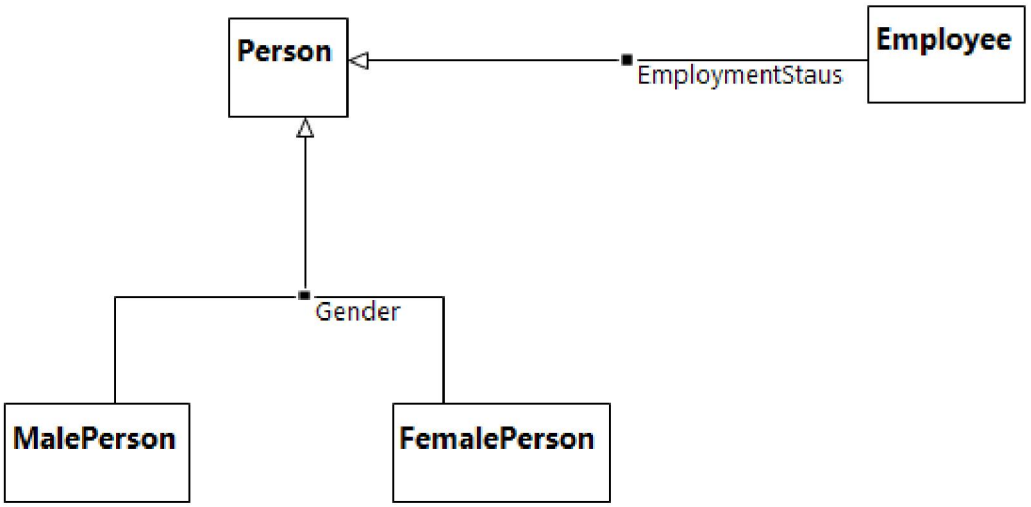
ULML

Generalization

- Generalizations are another first class citizen of LML
  - can also be view in an exploded or visually “insignificant“ form
- The name of the generalization usually identifies the discriminant
- Usual constraints such as disjoint and complete also supported



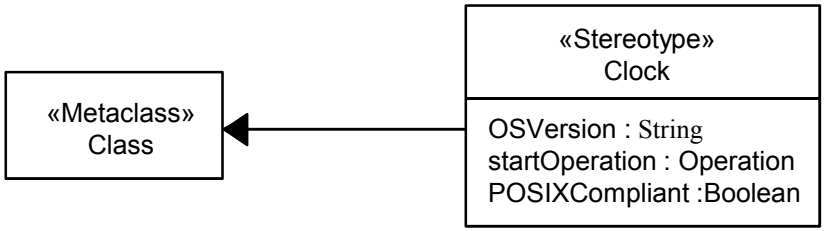
UML



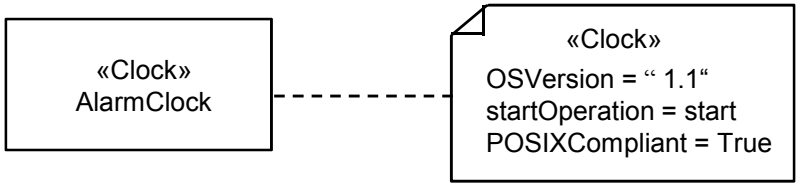
LML

Stereotypes

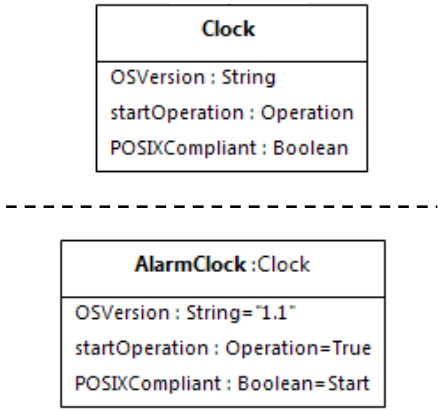
- Stereotypes and stereotyping supported by metamodeling



Stereotype



Stereotyped Element

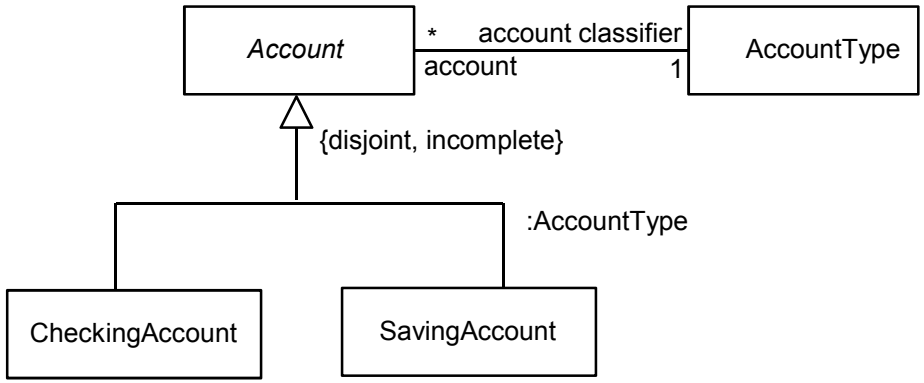


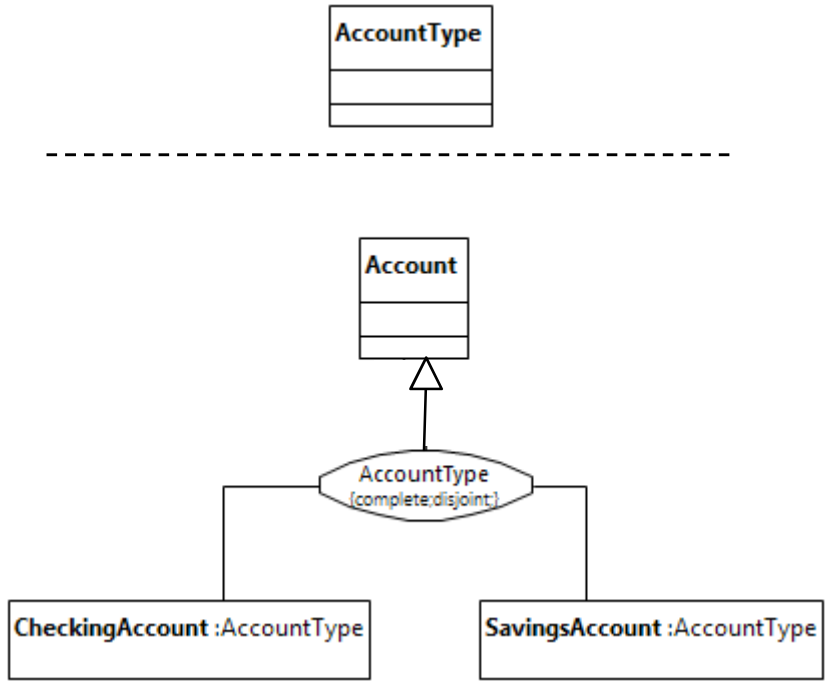
UML

LML

Powertypes

- LML also provides natural support for powertypes
- power types are metaclasses whose instances are also subclasses.
  - X is a power type of Y, if the instances of X are subclasses of Y



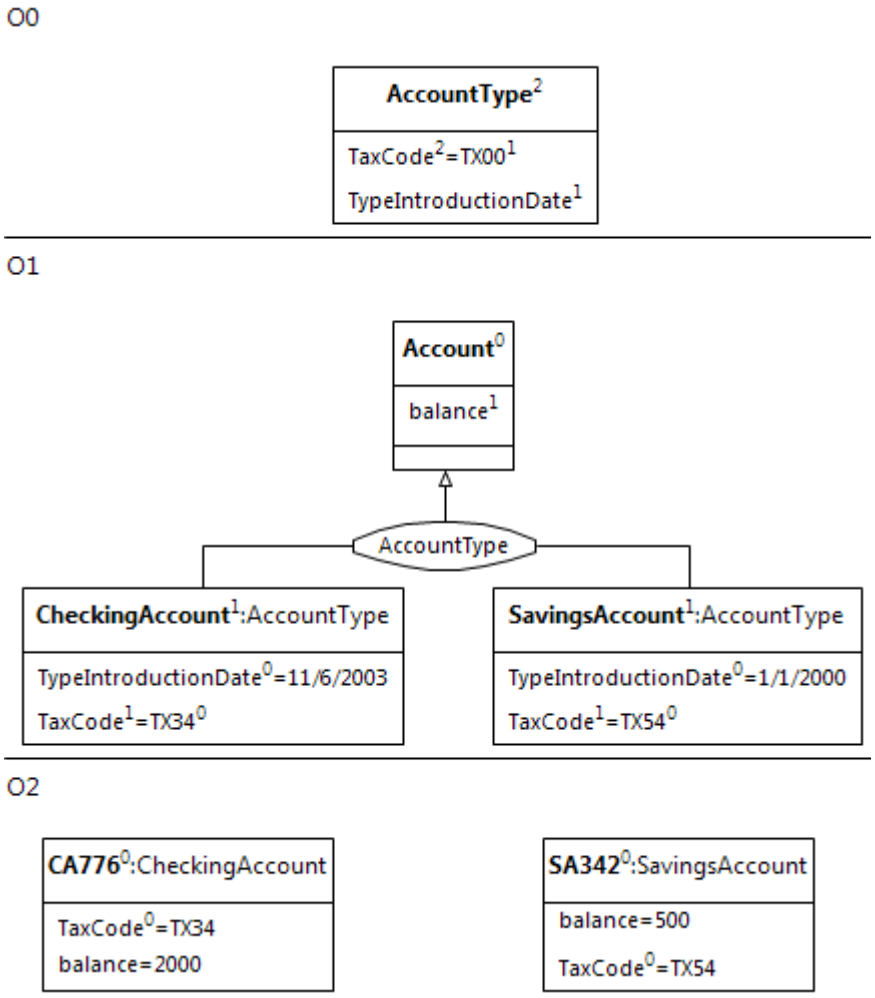


UML LML

Deep Instantiation

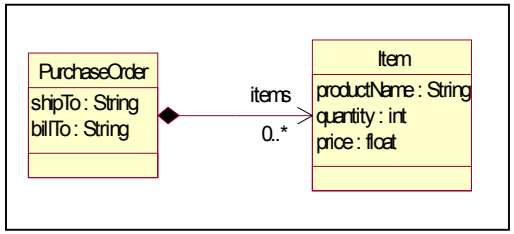
- a simple mechanism supporting deep characterization
- allows feature properties to be exactly specified using the notion of
  - ⇒ potency
- provides a simple but precise definition of instanceOf relationships (principle of application) over multiple classification levels
  - potency takes non-negative integer values
  - an instance of a type always has a potency that is one lower than the type
- although instantiation also lowers the level of a clabject by one,
  - the potency and level of a clabject do not have to be the same

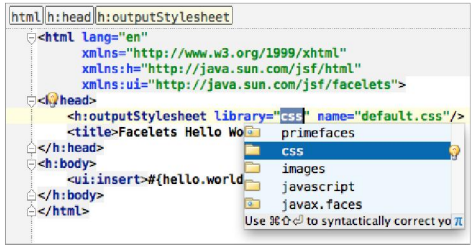
The Power Type Example Revisited

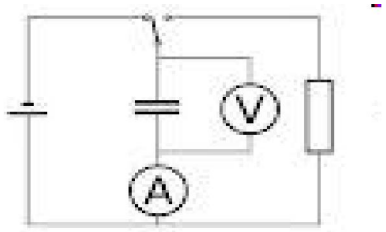


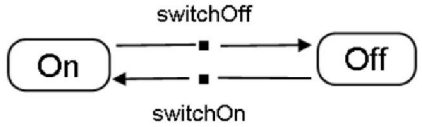
LML

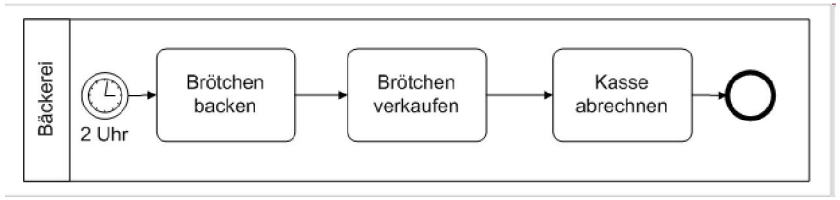
Domain Specific v General Purpose?











GPLs

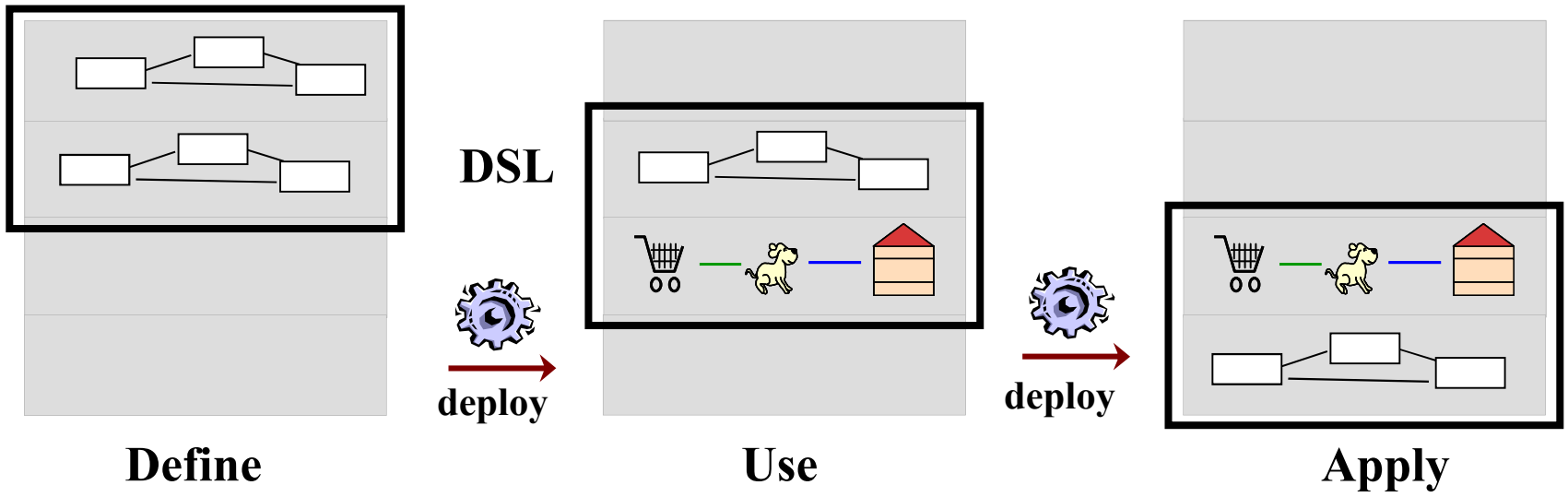
- E.g UML, Java, XML etc.
- Lingua Franca
- less concise and efficient
- facilitate inter-domain communication

DSLs

- E.g. BPMN, Circuit Diagrams etc
- optimized for special tasks
- more concise and efficient
- “Tower of Babel“ problem

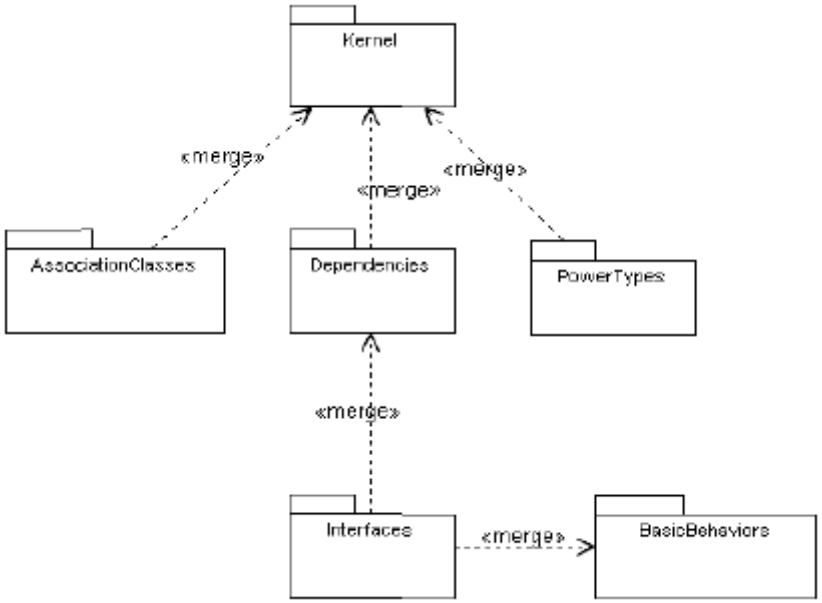
DSL State-of-the-Art

- DSLs defined in rigid, disjoint phases and environments
- result of one phase has to be “deployed” in a major compilation step to enable the second phase
- in each phase/environment only one classification level is “soft”
- only one concrete syntax is available in a given phase.

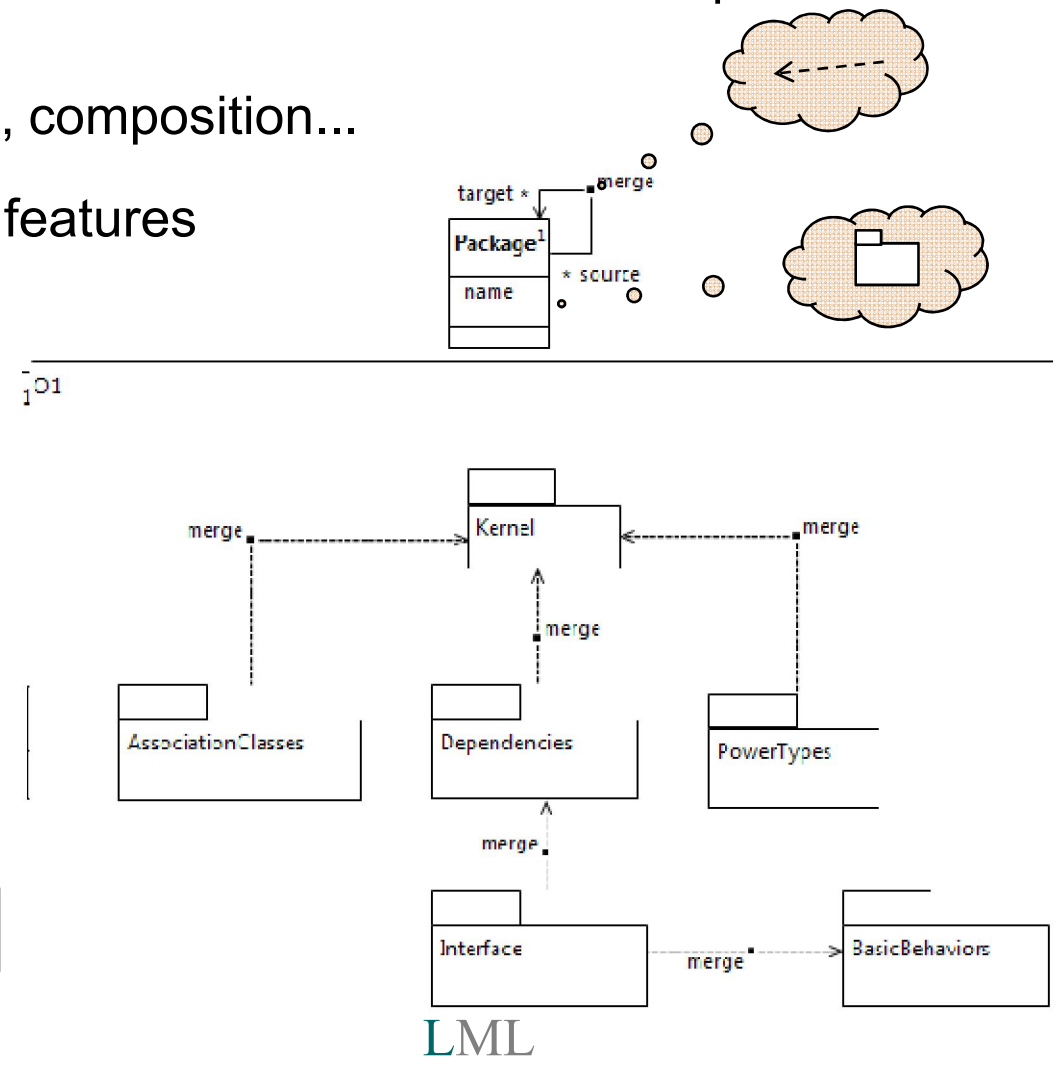


Package Diagrams as a DSL

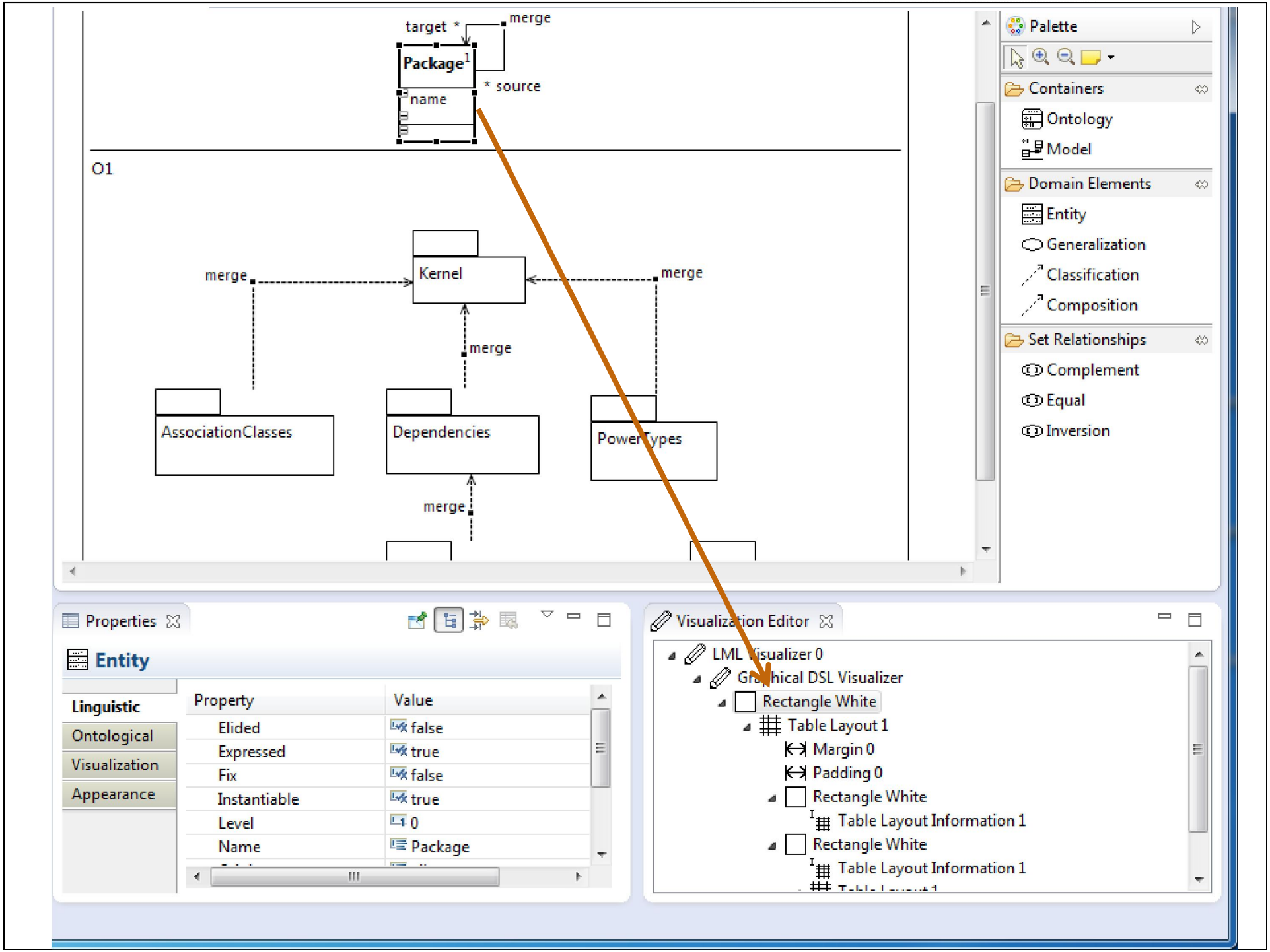
- Many features of UML were not included in the earlier comparison of UML and LML
  - packages, components, composition...
- these can be added as DSL features



UML



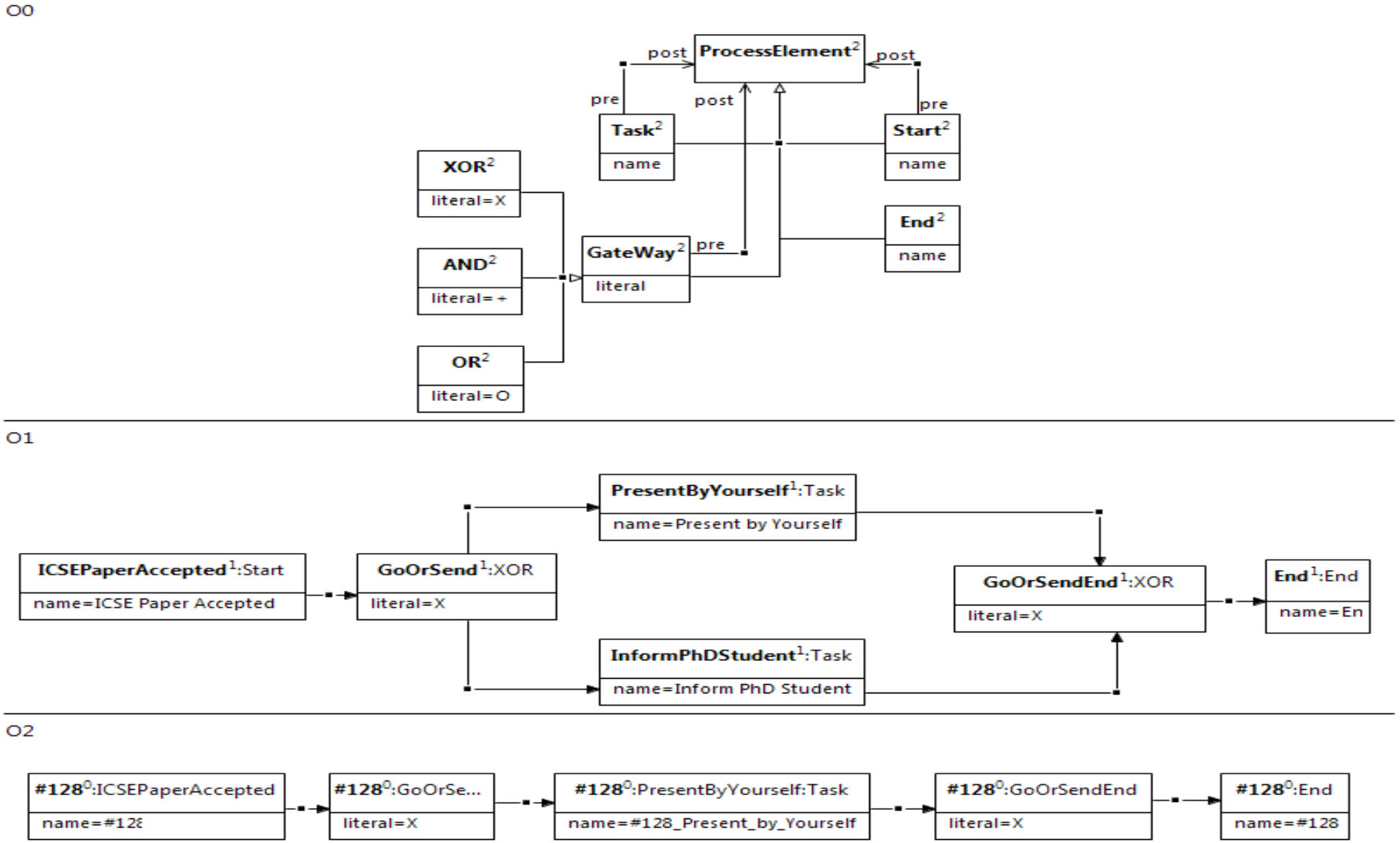
LML


target *
merge
Package
1
* source
name
O1
merge
Kernel
merge
merge
AssociationClasses
Dependencies
PowerTypes
merge
Palette
Containers
Ontology
Model
Domain Elements
Entity
Generalization
Classification
Composition
Set Relationships
Complement
Equal
Inversion
Properties
Entity
Linguistic
Ontological
Visualization
Appearance
Property
Value
Elided
false
Expressed
true
Fix
false
Instantiable
true
Level
0
Name
Package
Visualization Editor
LML Visualizer 0
Graphical DSL Visualizer
Rectangle White
Table Layout 1
Margin 0
Padding 0
Rectangle White
Table Layout Information 1
Rectangle White
Table Layout Information 1

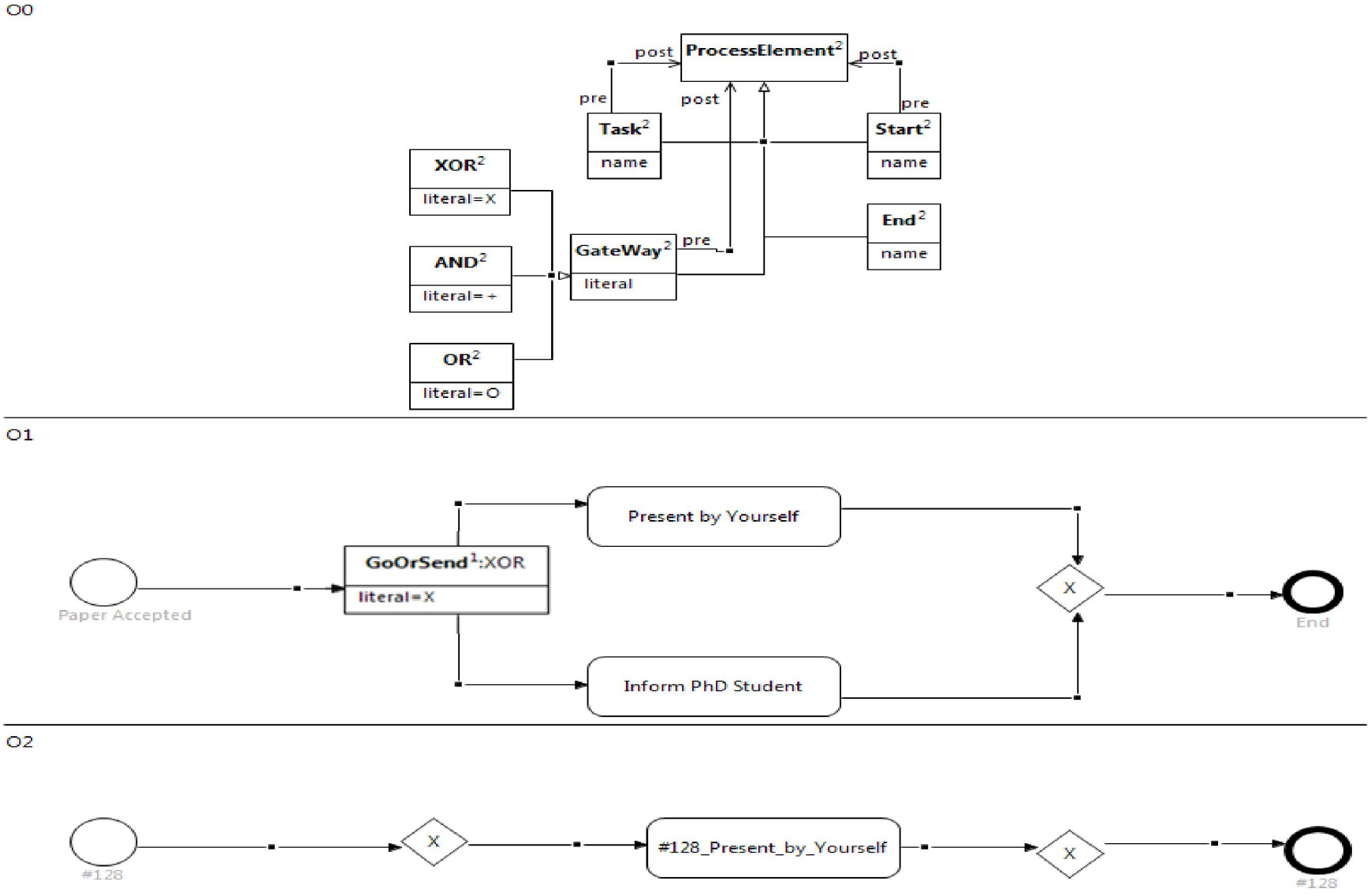
Symbiotic Languages

- Symbiosis – a relationship in which two things compenstate for each other‘s weaknesses and reinforce each other‘s strengths
- Symbiotic languages
  - coexists and can be mixed in arbitrary ways
  - Melanie supports a symbiosis between general purpose and domain specific visualizations
- Key features –
  - choice between purely general purpose or domain specific representations of the same model
  - symbols can be arbitrarily mixed
    - any symbol can be dynamically toggled from domain-specific to general purpose and back
  - can be applied at multiple classification level (i.e. models) simultaneously

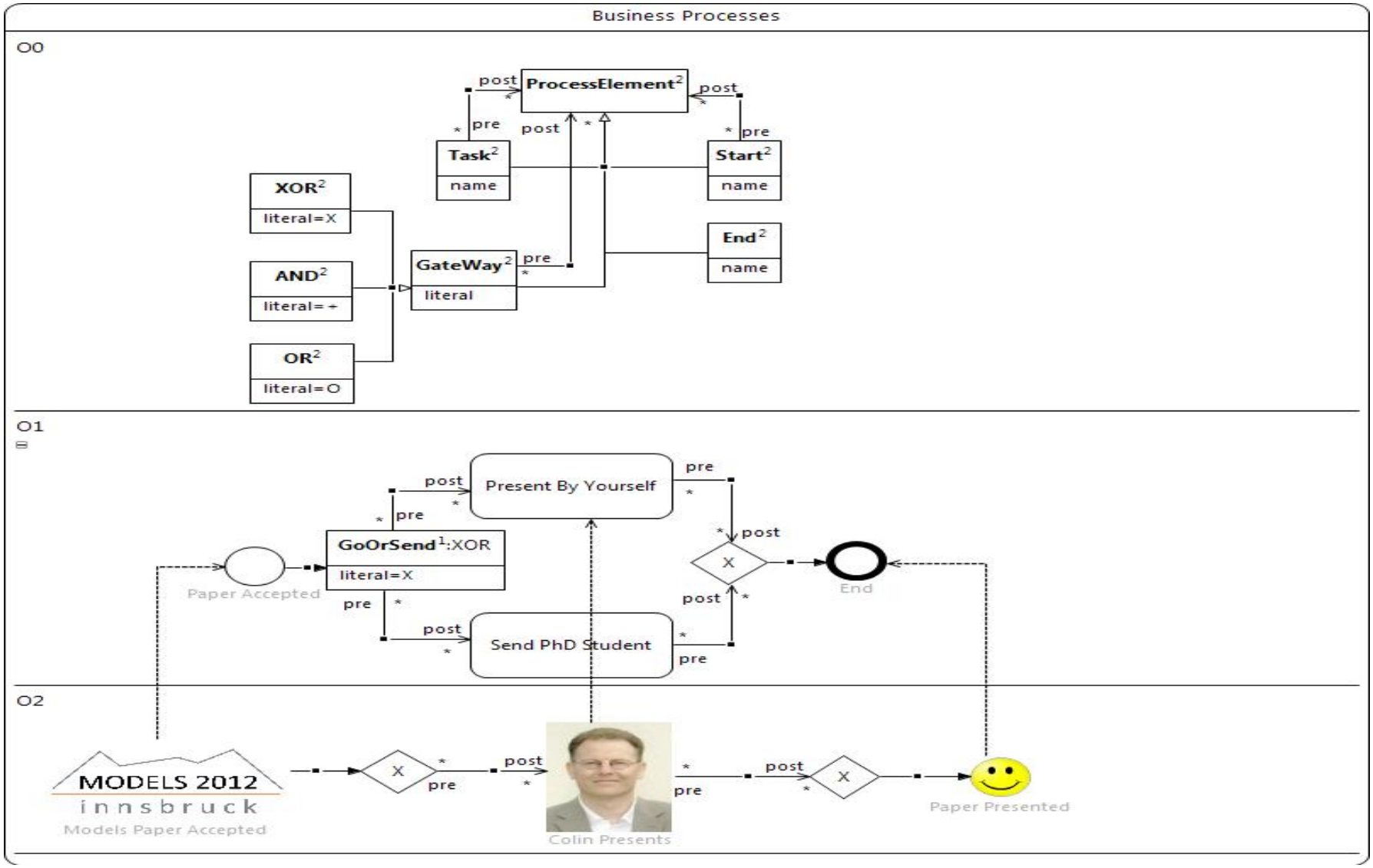
Symbiotic Language Example (1/3)



Symbiotic Language Example (2/3)



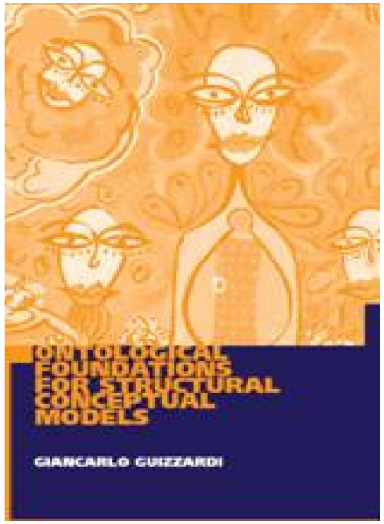
Symbiotic Language Example (3/3)



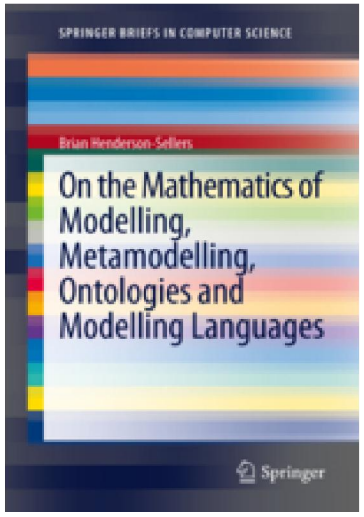
Foundation Ontologies

- a foundation ontology (also known as a top-level ontology or upper ontology) describes very general concepts that are common across all knowledge domains
- meet the following basic needs [EHA13]
  - promote reuse in a higher level of abstraction aimed at maximising the reuse of domain models
  - produce domain specifications that are truthful to reality
  - theoretical foundations for conceptual modelling languages
- Examples
  - Bunge-Ward-Weber (BWW), Basic Formal Ontology (BFO), General Formal Ontology (GFO), Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), Suggested Upper Merged Ontology (SUMO),
  - Unified Foundation Ontology (UFO) [GG05]

References



- [GG05] Guizzardi, G. Ontological Foundations for Structural Conceptual Models, PhD Thesis, University of Twente, The Netherlands.
  - published as the book “Ontological Foundations for Structural Conceptual Models”, Telematica Instituut Fundamental Research Series No. 15, ISBN 90-75176-81-3 ISSN 1388-1795

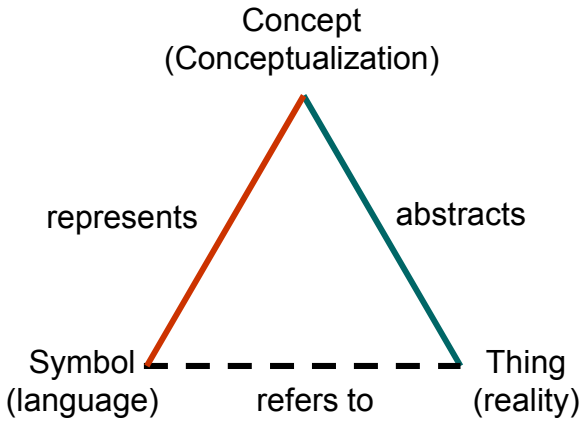


- [EHG13] Owen Eriksson, Brian Henderson-Sellers, and Pär J. Agerfalk. 2013. Ontological and linguistic metamodelling revisited: A language use approach. Inf. Softw. Technol. 55, 12, December 2013.
- B. Henderson-Sellers, On the Mathematics of Modelling, Metamodelling,Ontologies and Modelling Languages, SpringerBriefs in Computer Science, Springer-Verlag, Heidelberg, 2012.

Semantics of Models

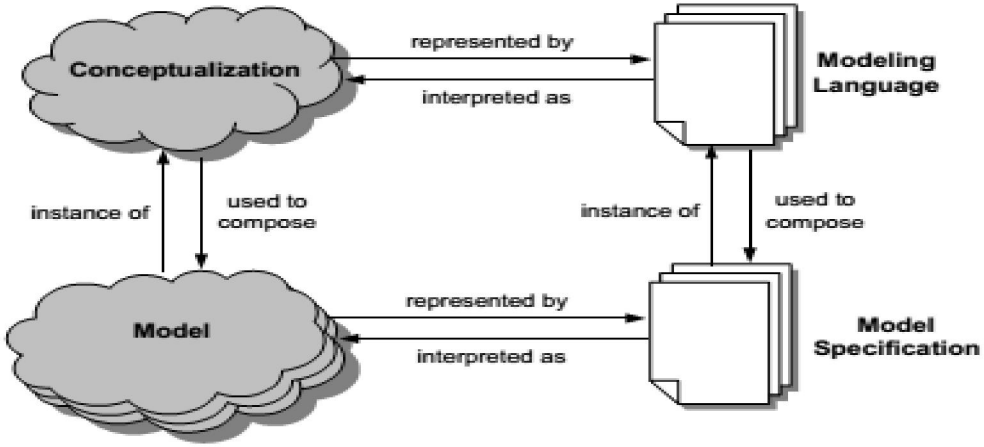
Ullmann's Triangle

- basis for understanding how symbols represent concepts in a language
- "the relation between language and reality is always intermediated by a certain conceptualization" (Baldinger)



Guizzardi's Square

- characterizes relationship between conceptualizations models, languages and specializations [GG05]

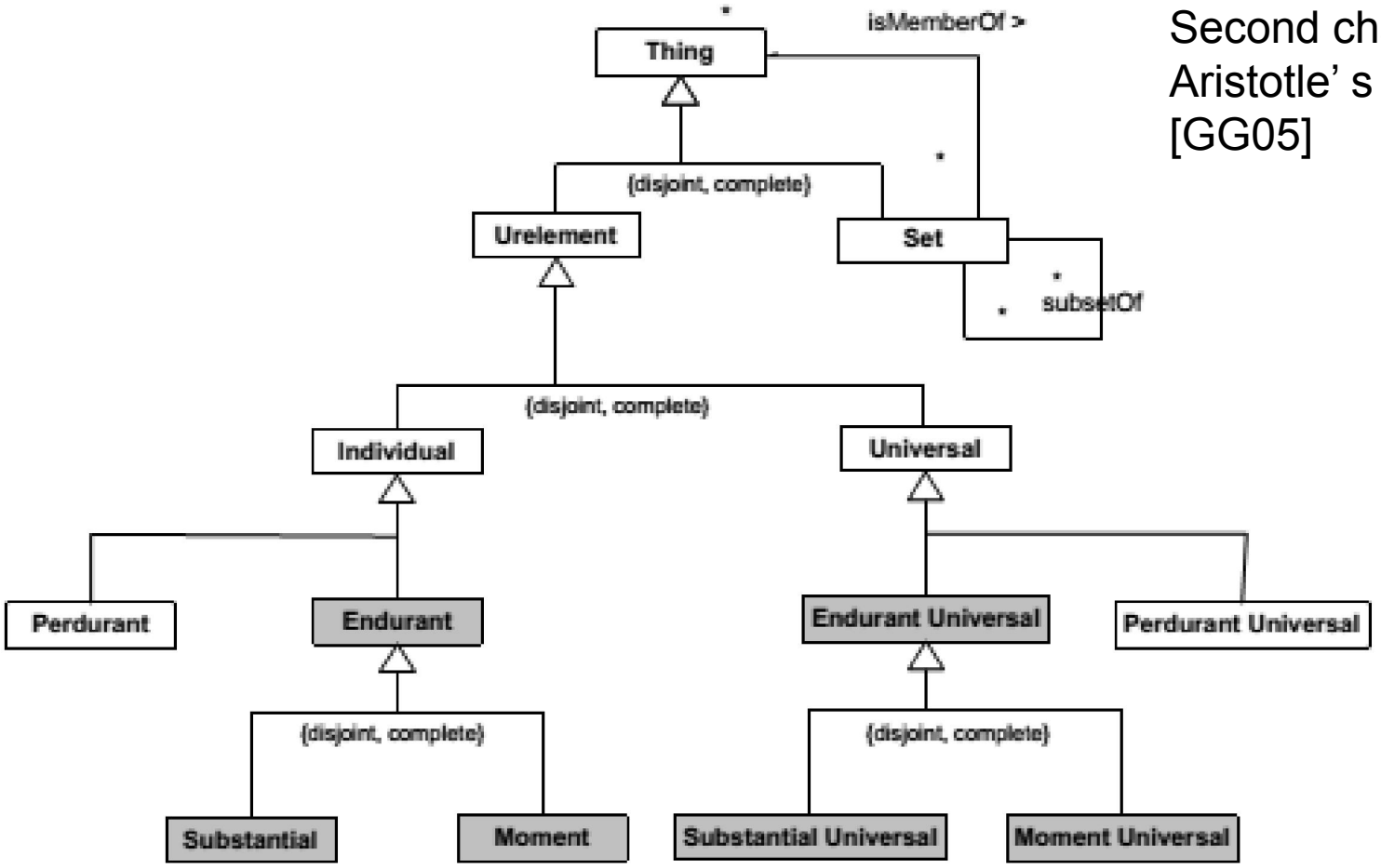


Concrete Incarnation

- In practice, boils down to a detailed taxonomy of modelling constructs capturing and philosophically well-founded concepts
- Based on a traditional (two-level) class / object dichotomy
  - Aristotle's four category ontology
- Most widely used and practical incarnation is OntoUML
  - a UML profile capturing the UFO taxonomy
  - supported by a tool developed by NEMO (Ontology & Conceptual Modeling Research Group) in Vitoria, Brazil
- Arguments for validity and utility are philosophical
  - the necessity or correctness of modelling features can not be proven per se
  - often the information they convey can be modelling in another way (albeit perhaps less elegant)
    - phases can be modelled by attributes of enumeration types

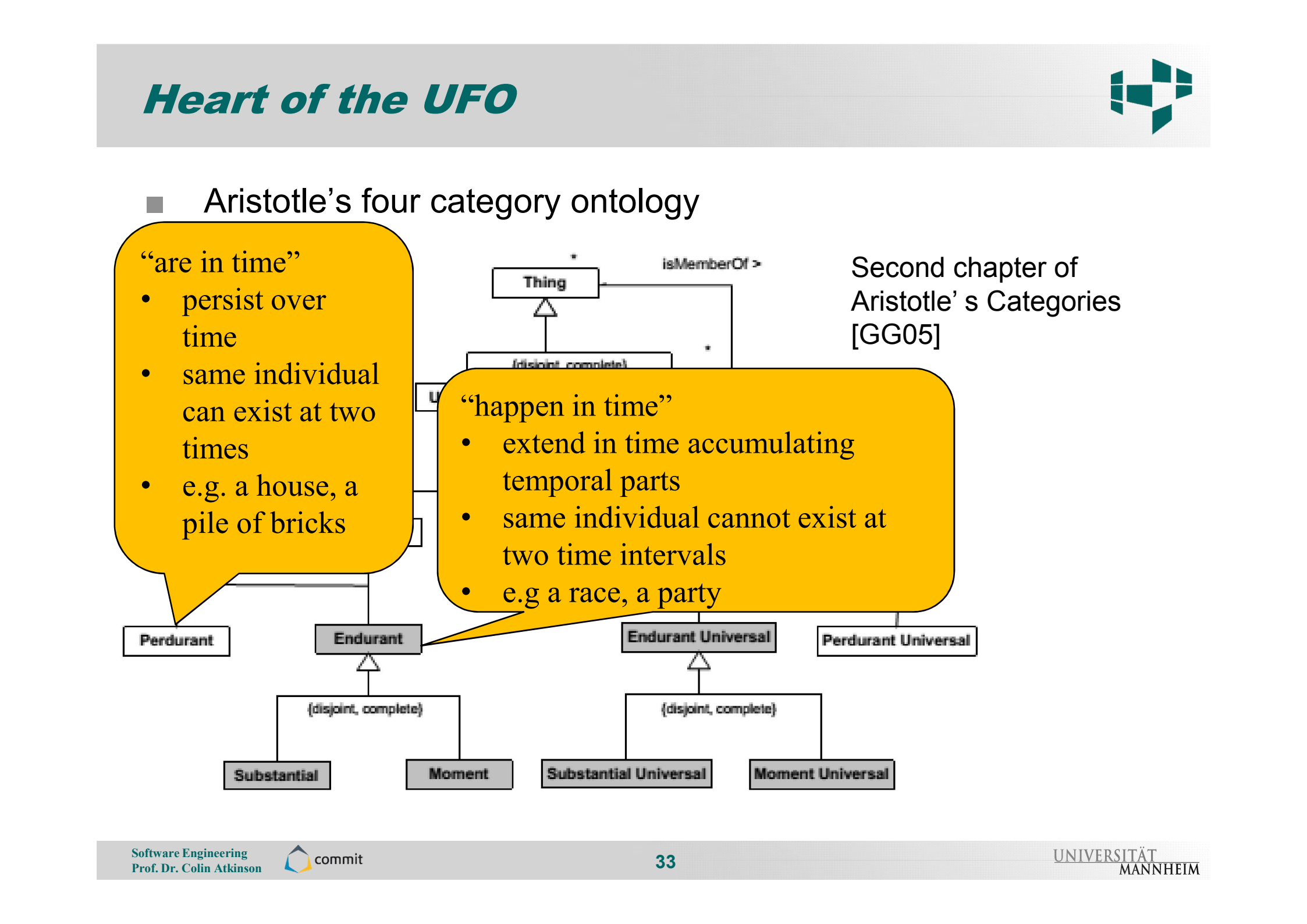
Heart of the UFO

- Aristotle's four category ontology



Second chapter of Aristotle' s Categories [GG05]

Heart of the UFO



- Aristotle's four category ontology

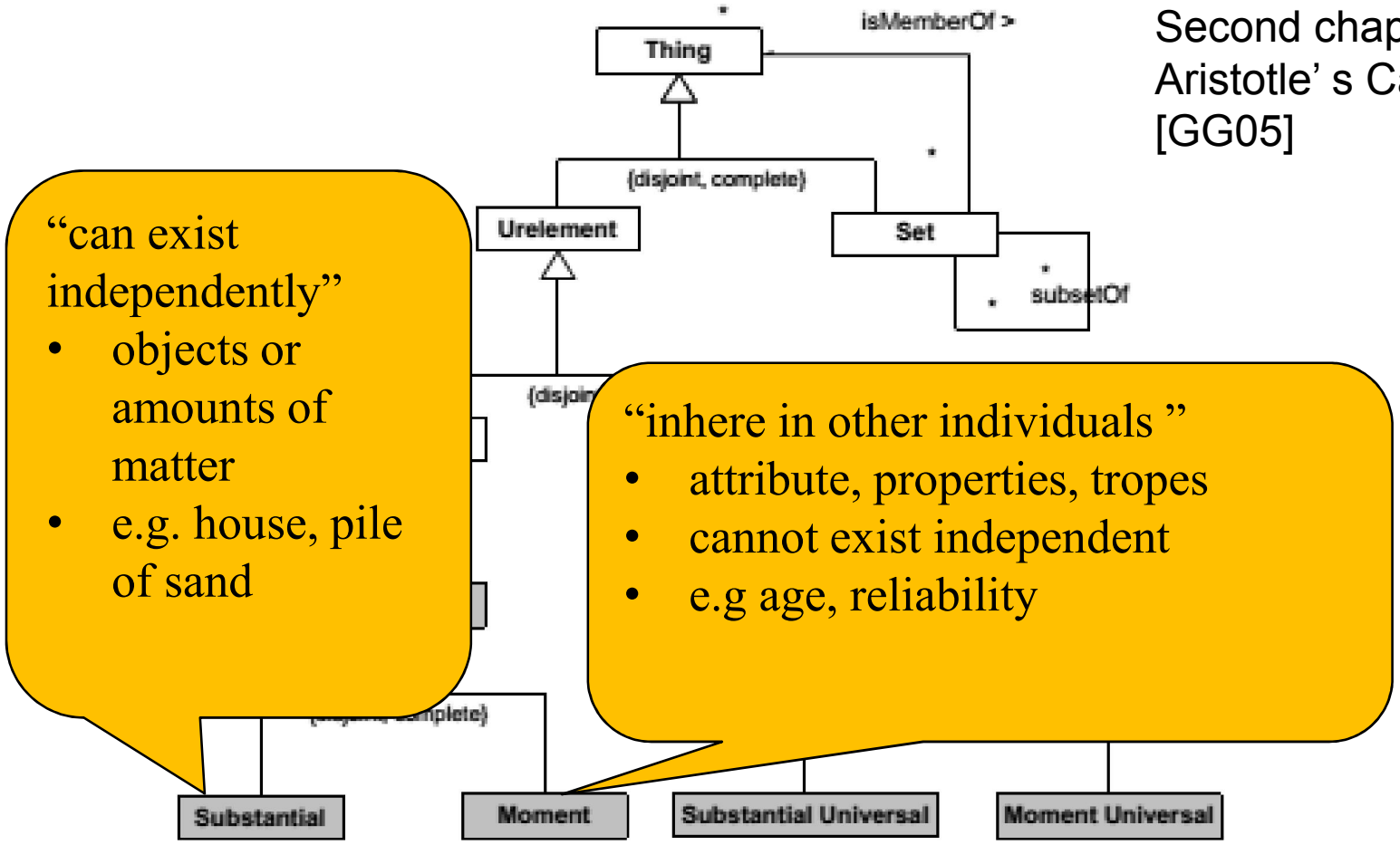
"are in time"
- persist over time
- same individual can exist at two times
- e.g. a house, a pile of bricks

"happen in time"
- extend in time accumulating temporal parts
- same individual cannot exist at two time intervals
- e.g a race, a party

Second chapter of Aristotle' s Categories [GG05]

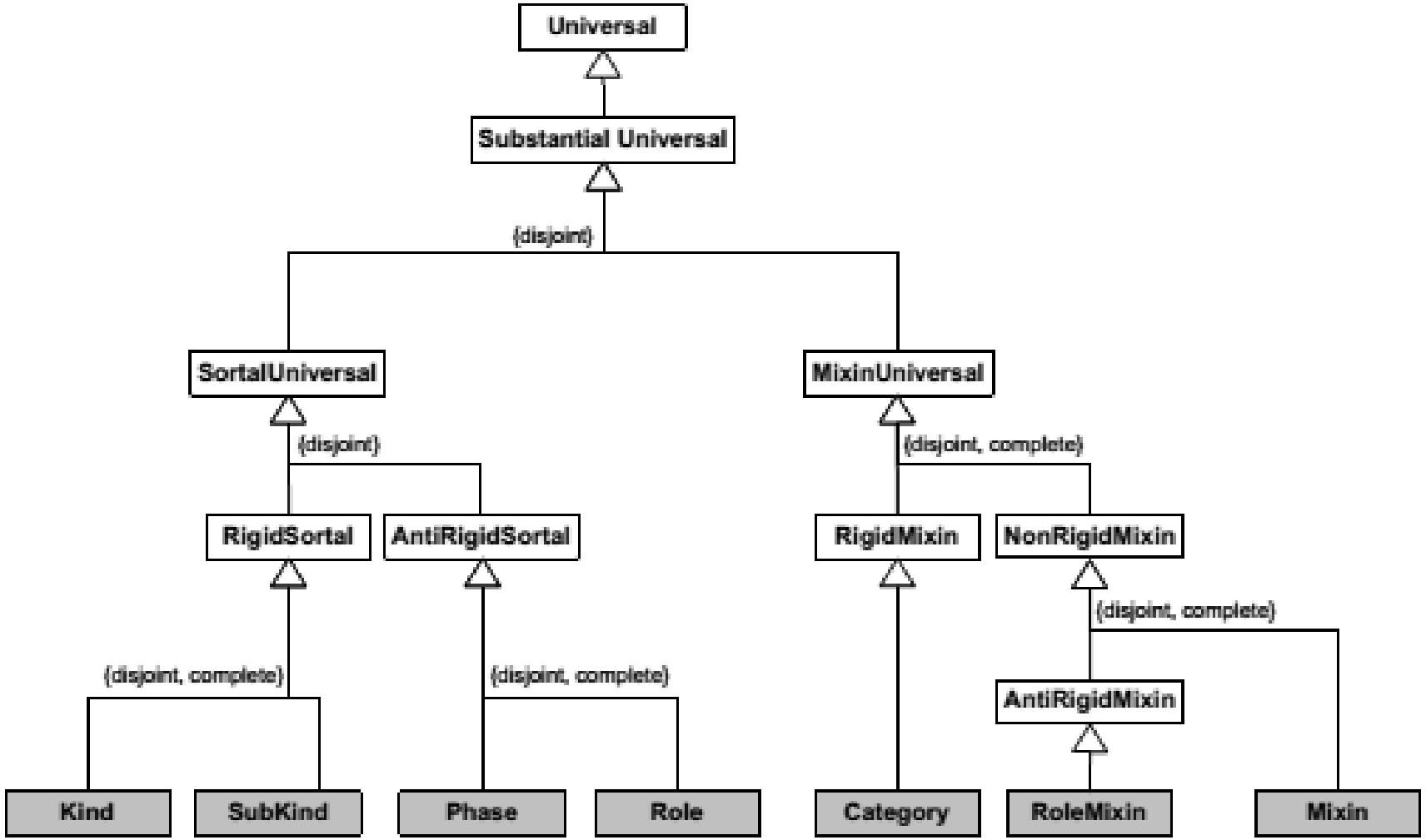
Heart of the UFO

- Aristotle's four category ontology

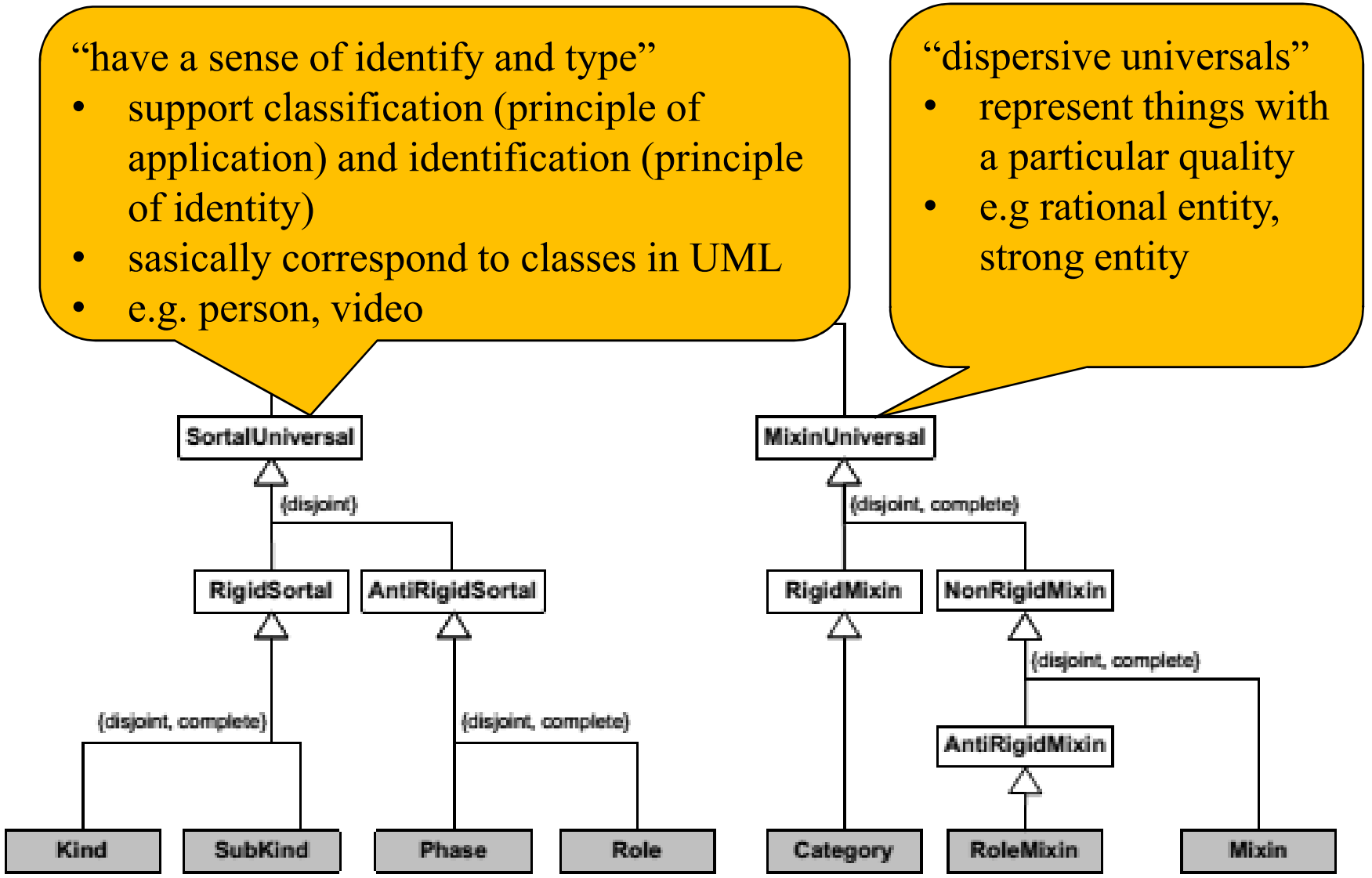


Second chapter of Aristotle' s Categories [GG05]

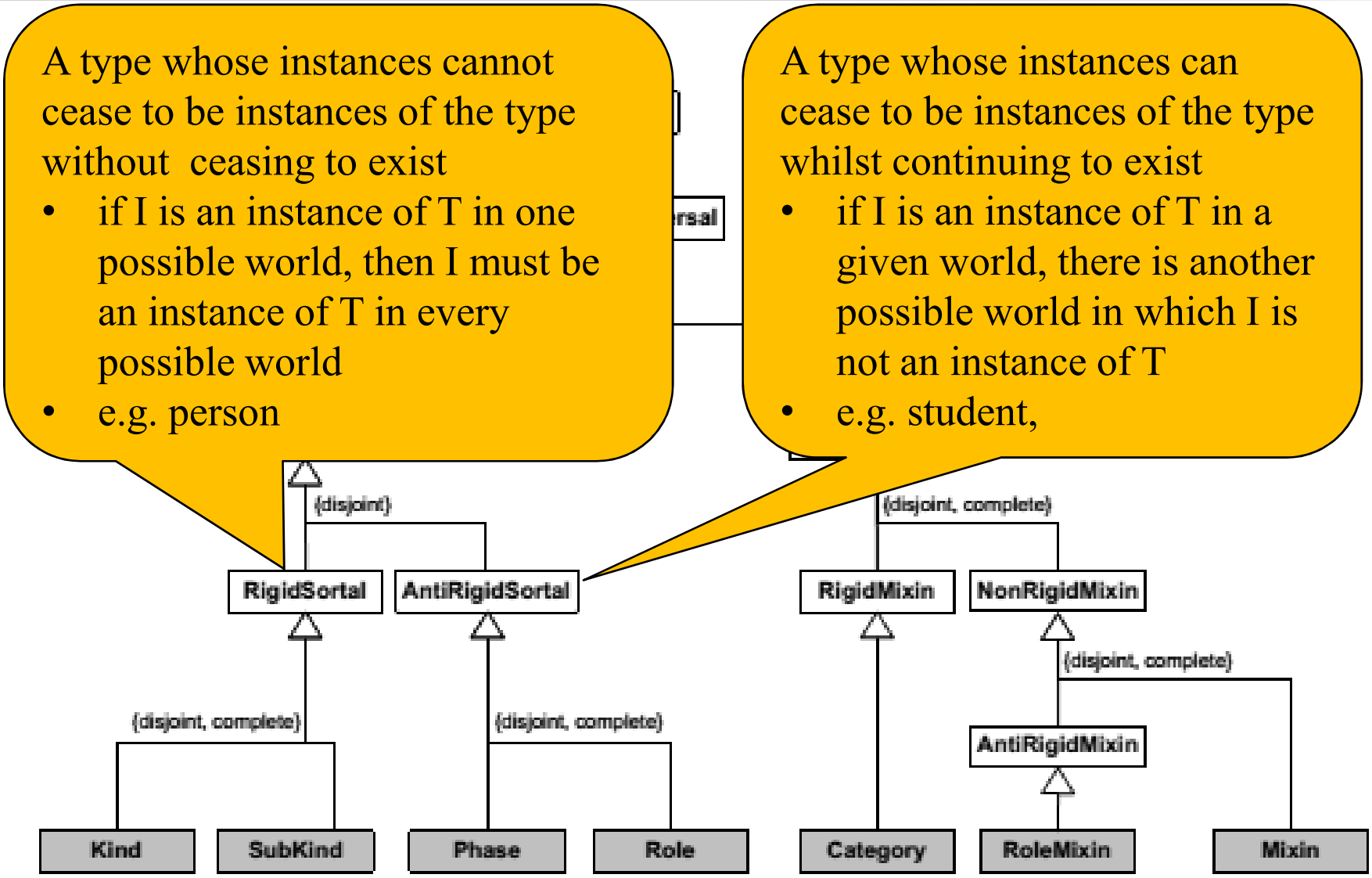
Taxonomy of Substantial Universals



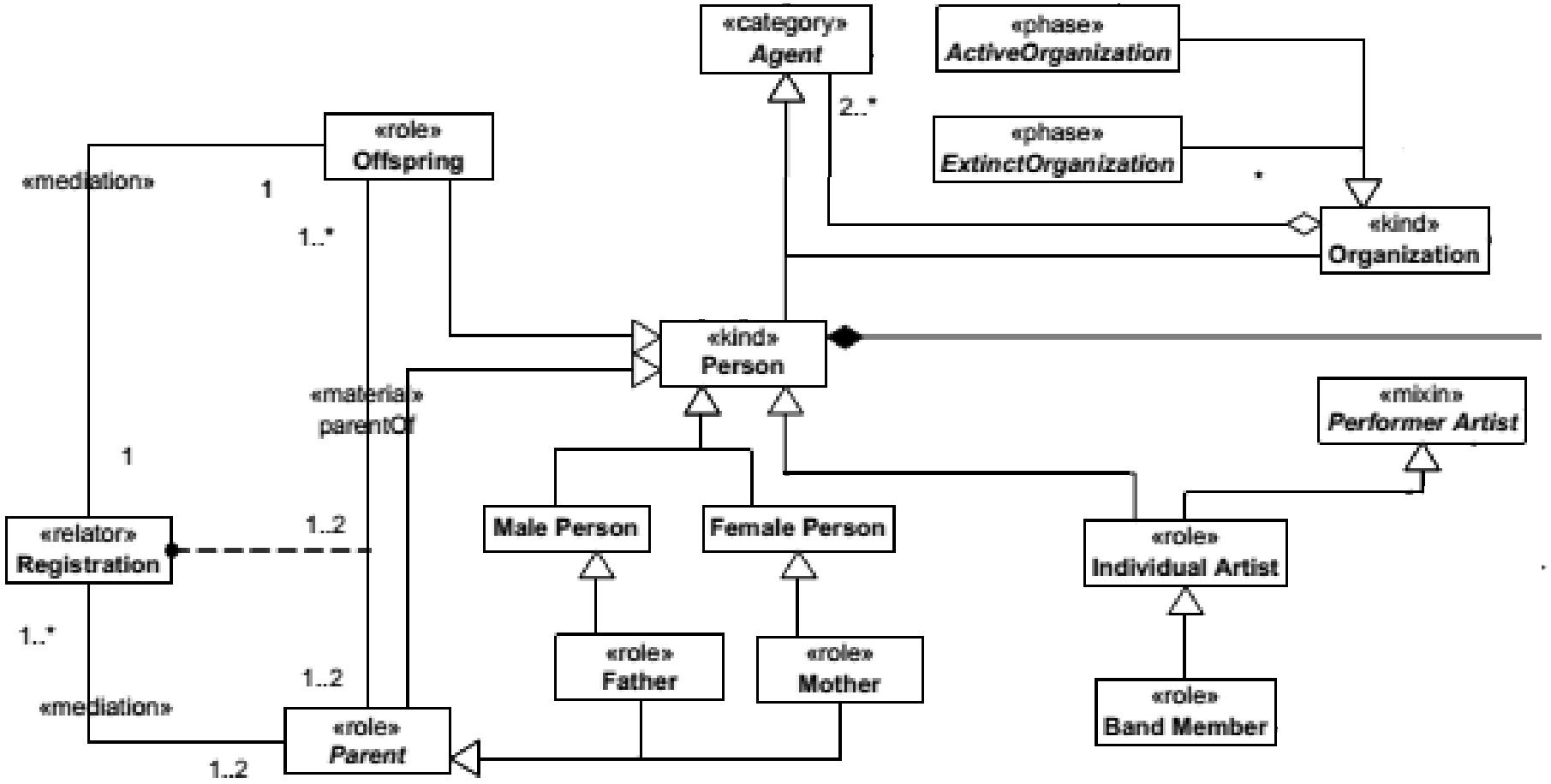
Taxonomy of Substantial Universals



Taxonomy of Substantial Universals



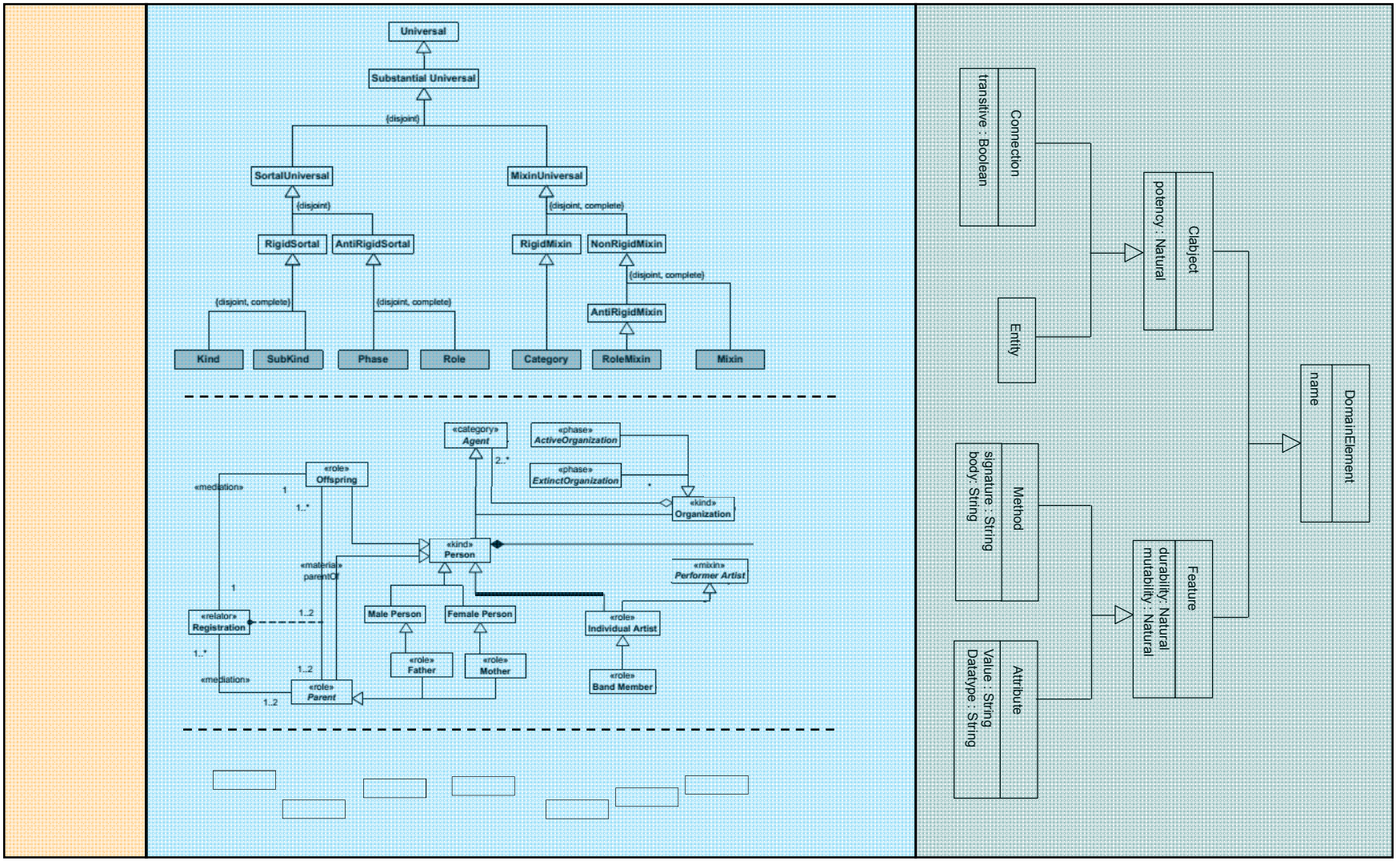
Example (OntoUML) [GG05]



Compatability of FOs and Deep Modeling

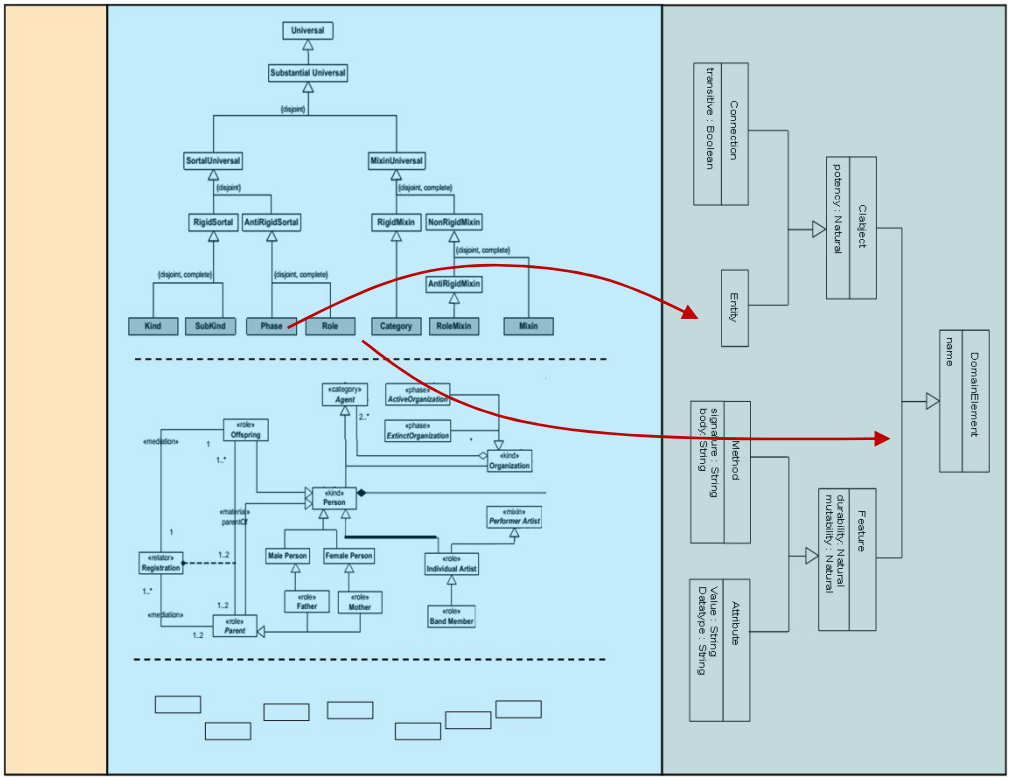
- At first sight FOs and deep modelling are fundamentally incompatible
  - UFO based on the fundamental universal / instance dichotomy
  - deep modelling exists to precisely to overcome this limitation
- UFO does not explain some concepts that are important in IT
  - abstract classes
  - interfaces
  - power types
  - stereotypes?
- but UFO can be regarded as a DSL in deep modelling
  - provides exactly the same expressive power as OntoUML

UFO as DSL

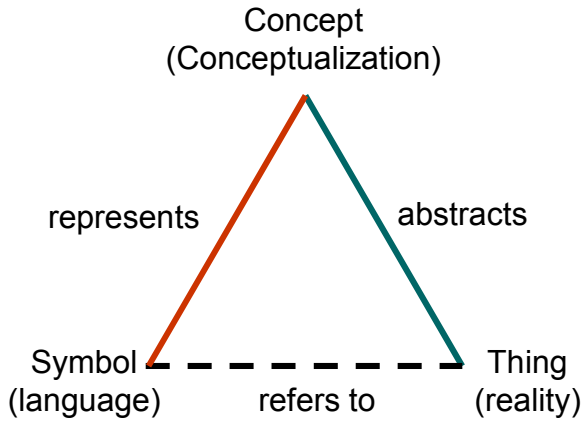

Universal
Substantial Universal
{disjoint}
SortalUniversal
MixinUniversal
{disjoint}
{disjoint, complete}
RigidSortal
AntiRigidSortal
RigidMixin
NonRigidMixin
{disjoint, complete}
{disjoint, complete}
{disjoint, complete}
{disjoint, complete}
AntiRigidMixin
Kind
SubKind
Phase
Role
Category
RoleMixin
Mixin
«category» Agent
«phase» ActiveOrganization
«phase» ExtinctOrganization
«role» Offspring
«mediation»
1
1..*
2..*
*
«kind» Organization
«kind» Person
«material» parentOf
«mixin» Performer Artist
1
«relator» Registration
1..2
Male Person
Female Person
«role» Individual Artist
1..*
«mediation»
1..2
«role» Father
«role» Mother
«role» Band Member
«role» Parent
1..2
Connection
transitive : Boolean
Entity
Clabject
potency : Natural
DomainElement
name
Method
signature : String
body: String
Feature
durability: Natural
mutability : Natural
Attribute
Value : String
Datatype : String

Big Question

- In an optimal deep modelling environment, what features in (e.g.) UFO be moved to the linguistic (meta)model ?
  - would the notion of rigid or antirigid clabjects be useful?
  - would it be useful to introduce the notion of role clabjects?
- For example EHA13 argue that Dog is a substantial class while Breed is a moment type
  - but what if Dog were not in the model?
- The top ontological model and the linguistic model need to be aligned and optimized to support deep, FO based modeling

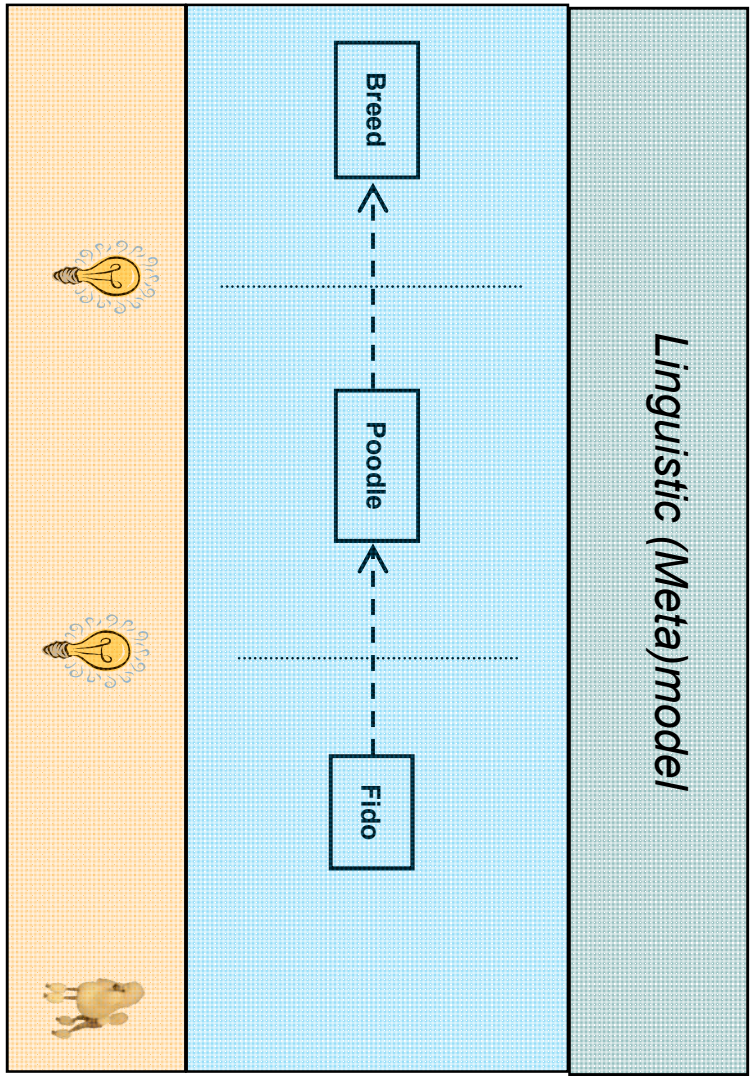


Deep Interpretation of Ullman's Triangle

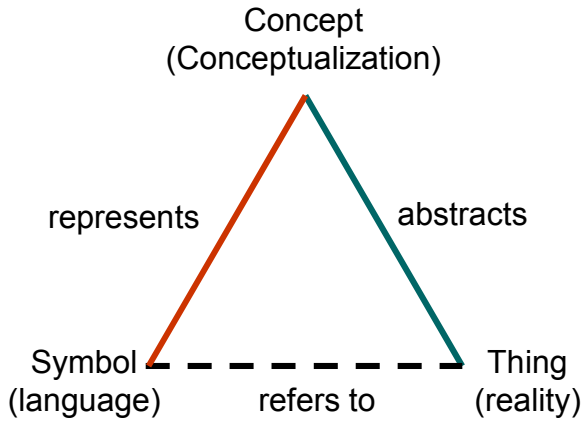


- ontological model elements are symbols
- real world entities, including concepts, are things
- higher level model elements are concepts

a concept can be an instance of another concept (counter to [EHA13])

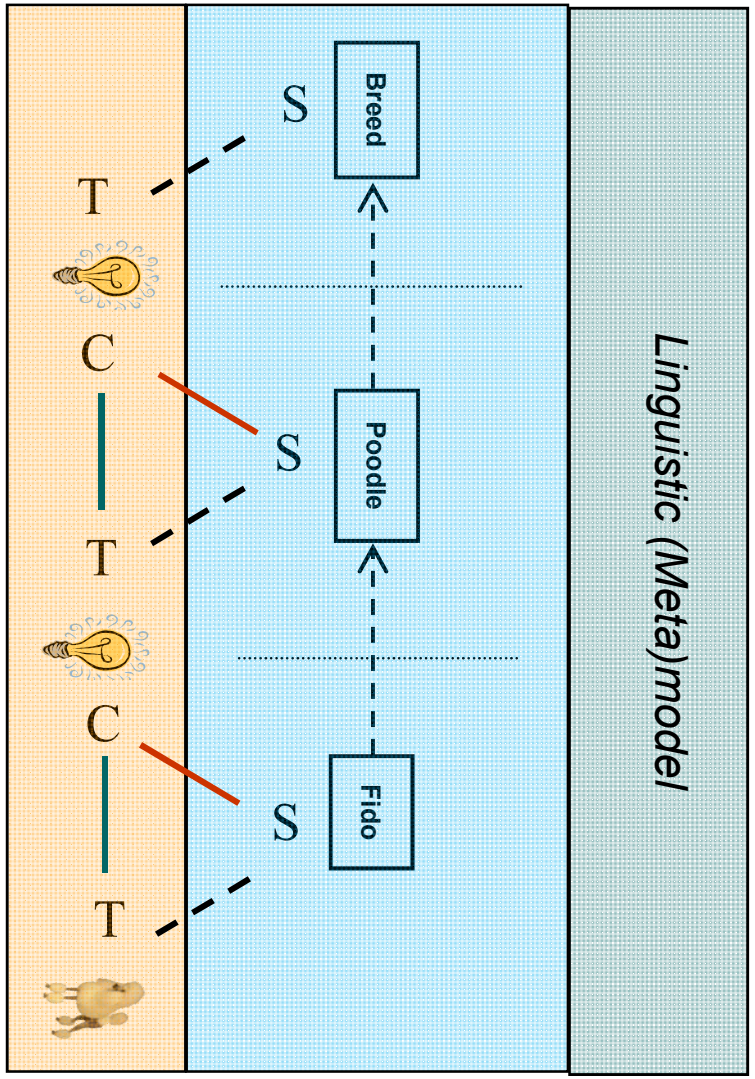


Deep Interpretation of Ullman's Triangle

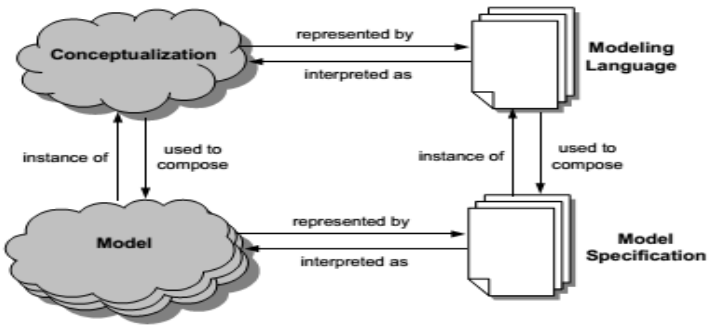


- ontological model elements are symbols
- real world entities, including concepts, are things
- higher level model elements are concepts

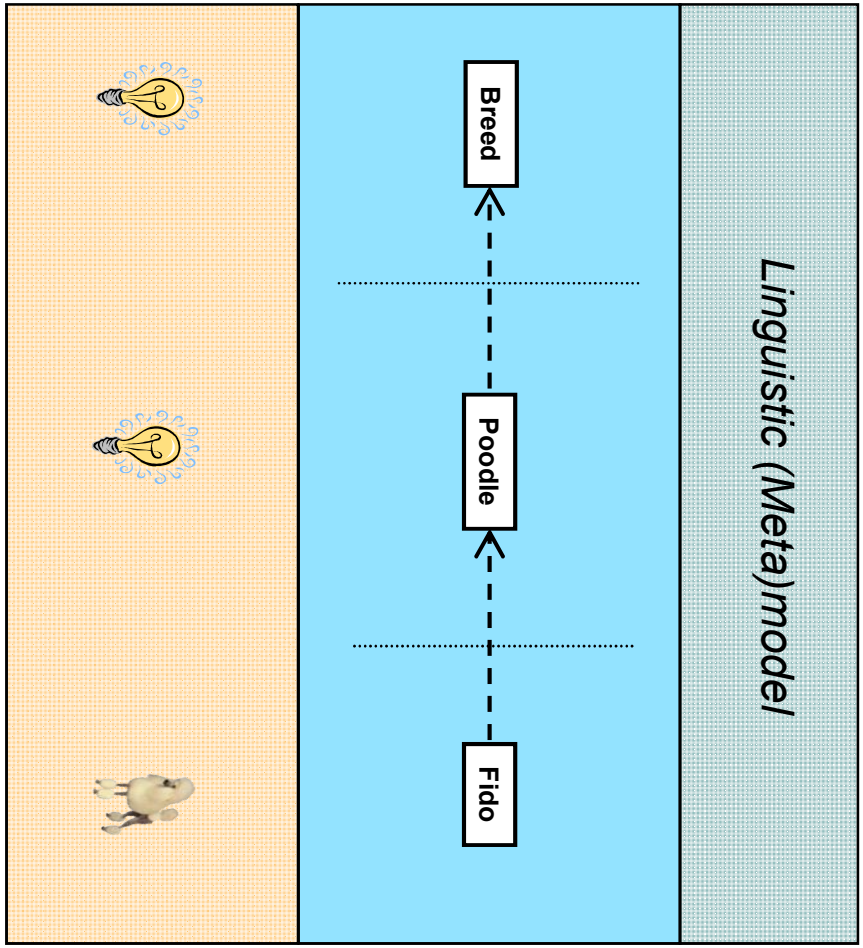
a concept can be an instance of another concept (counter to [EHA13])



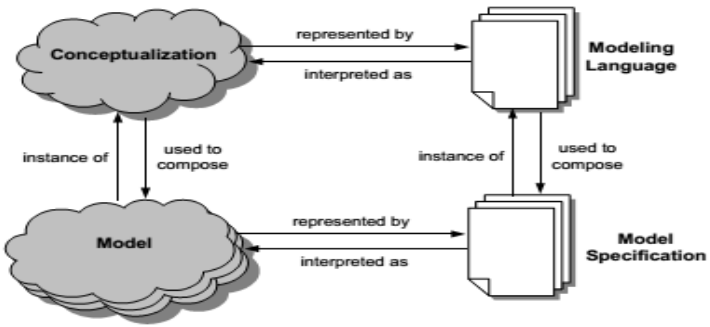
Deep Interpretation of Guizzardi's Square



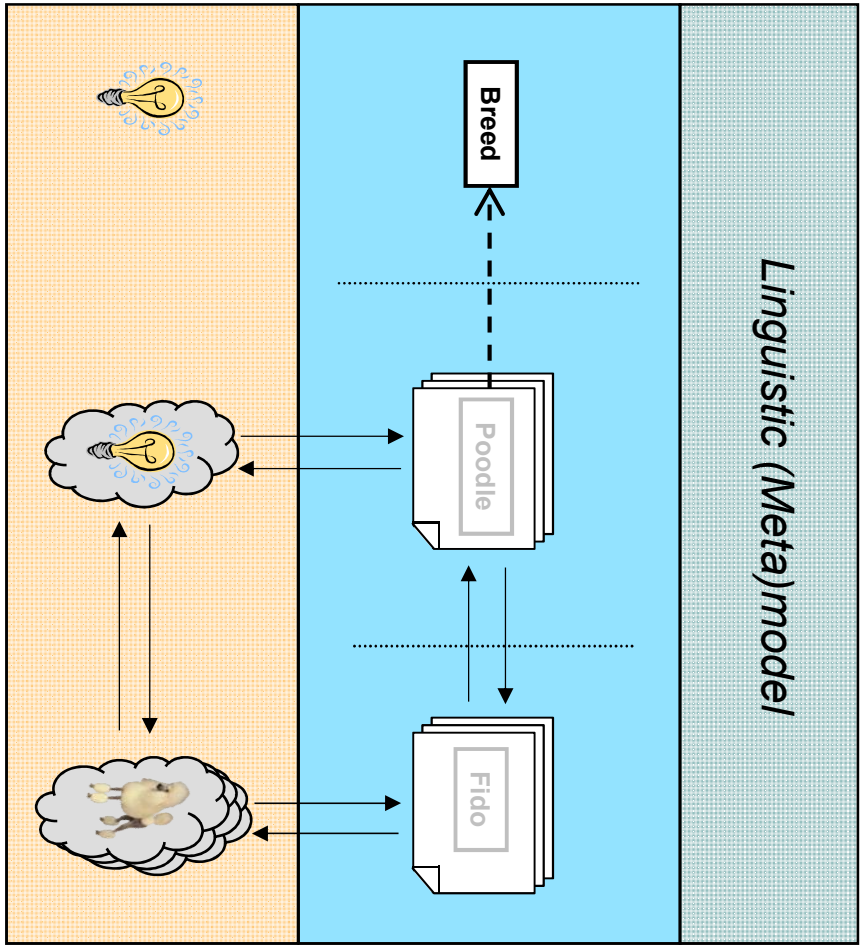
- model elements at one level are a language for the model specification at the level below
- a model language represents a conceptualization
- a model, a an instance of a conceptualization, is a model specification



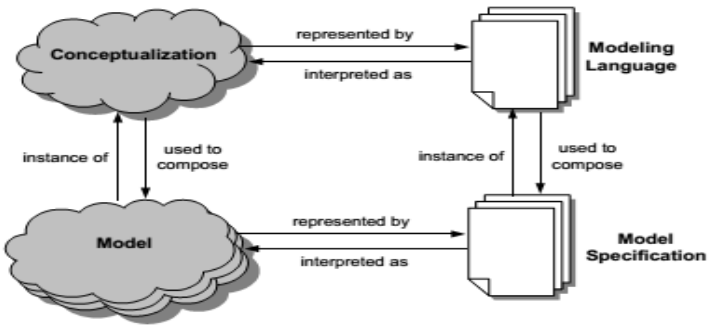
Deep Interpretation of Guizzardi's Square



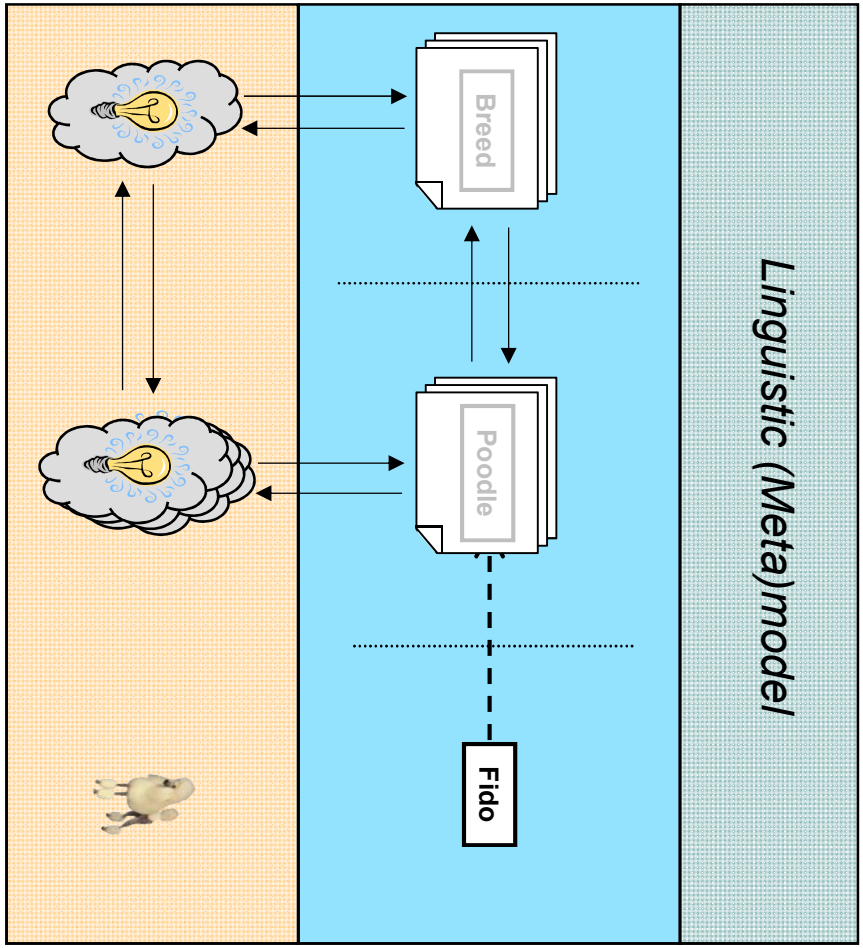
- model elements at one level are a language for the model specification at the level below
- a model language represents a conceptualization
- a model, a an instance of a conceptualization, is a model specification



Deep Interpretation of Guizzardi's Square



- model elements at one level are a language for the model specification at the level below
- a model language represents a conceptualization
- a model, an instance of a conceptualization, is a model specification



Conclusion

- FOs and deep modelling are at one level fundamentally incompatible, but at another level they are synergetic
  - an FO can be supported as a DSL in deep modelling
  - FO semantic models map well to a potency-based OCA (Ullmans's triangle, Guizzardi's square)
- "Linguistic" and "ontological" are highly misleading labels for the different forms of classification
  - infrastructural versus conceptual or domain
- The modeling community would benefit from a fundamental unification of FO and deep modelling
  - FOs needs to move beyond the two-level world view
  - Infrastructural model needs to be enhanced with FO concepts
- Deep modelling environment from Uni. Mannheim
  - www.melanee.org

Deep, SUM-based Environments