

---

# Ontological Analysis of Metamodel for Registering Business Objects

He Keqing, Jing Yixin, Zhou Yi

E-mail: [hekeqing@public.wh.hb.cn](mailto:hekeqing@public.wh.hb.cn)

State Key Laboratory of Software Engineering

Wuhan University

P.R.China

# Contents

---

- Ontology Method
- Ontological Analysis of Metamodel for Registering BO
- Classification of BO with Taxonomic Properties
- Conclusion

# 1. Ontology Method?

---

- 1.1 What is Ontology?
- 1.2 Metamodel Framework & Ontology

# 1.1 What is Ontology?

---

- Definition in Philosophy:  
Ontology is the branch of metaphysics that deals with the nature of being
- The subject of ontology is the study of the categories of things that exist or may exist in some domain.

# 1.1 What is Ontology?(Cont.)

---

## ■ Definition in Modeling Engineering

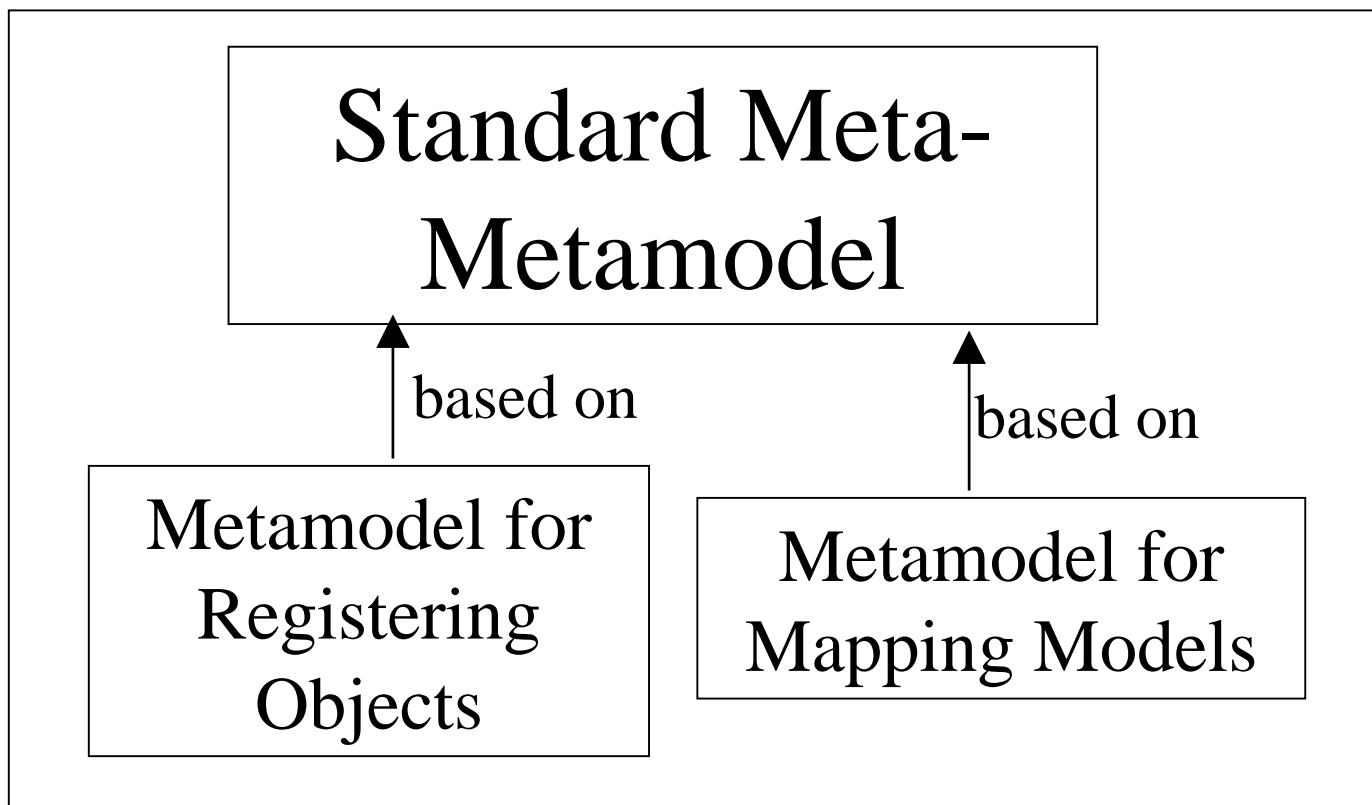
- (1) An ontology is a theory of what entities can exist in the mind of a knowledgeable agent.
- (2) An ontology is a explicit knowledge level specification of a conceptualization.
- (3) An ontology is an explicit, partial account of a conceptualization
- (4) etc.

## ■ A still debated issue

# 1.2 Metamodel Framework & Ontology

---

## ■ Metamodel Framework



# Standard Meta-Metamodel

---

- (1) Domain neutral
- (2) Describe general concepts
- (3) Be the base of metamodels

# Metamodel for Registering Object

---

- (1) Provides a schematic metamodel to be conformed
- (2) Different concepts are defined for BO registering and the properties required to classify the objects should be organized

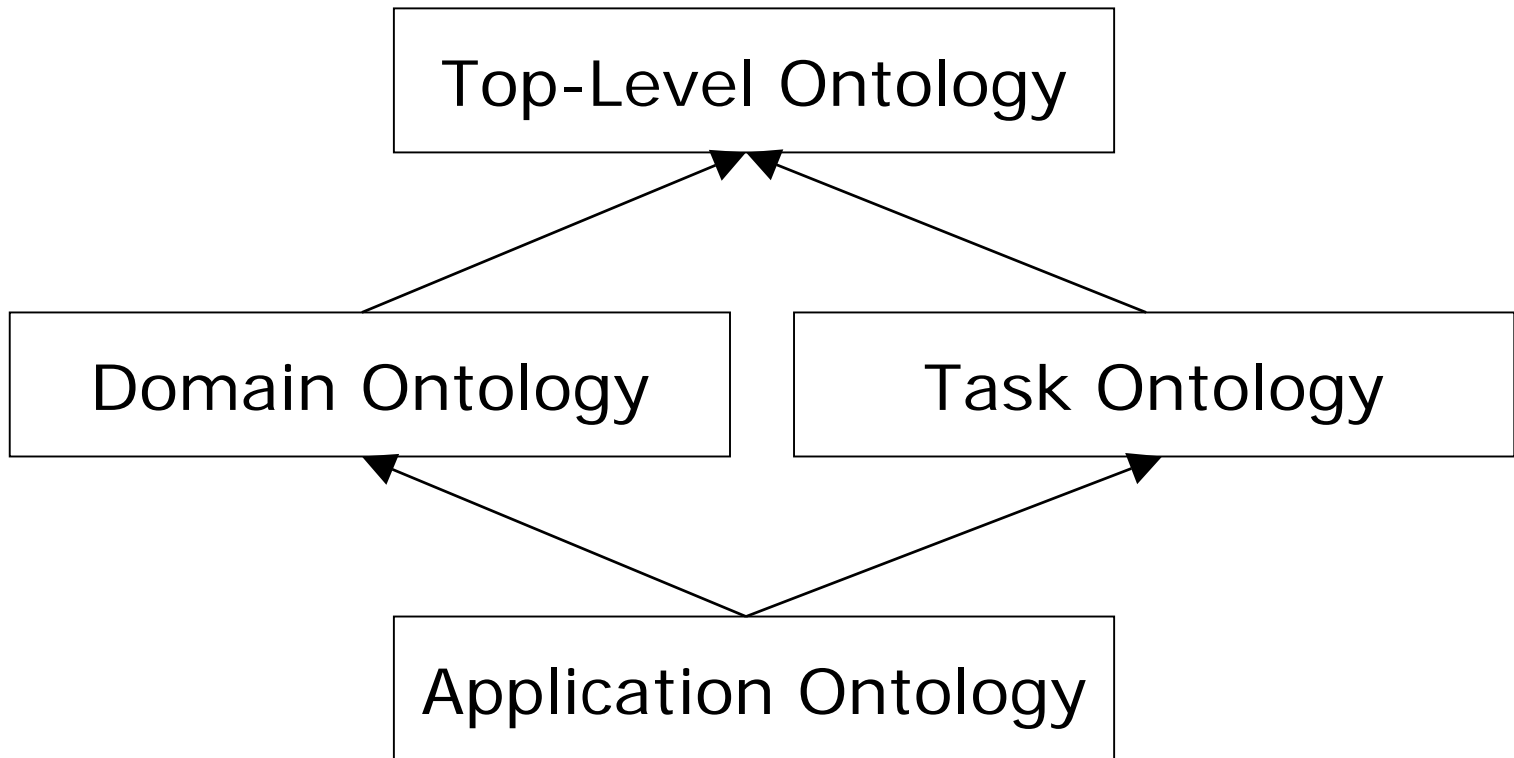
# Metamodel for mapping models

---

Realize the appropriate mapping between models which are developed with different modeling facilities or modeling constructs.

# Four Kinds of Ontology

---



# Top-Lever Ontology

---

- (1) Only describes the most general concepts, such as object, action, association
- (2) Independent of any domain
- (3) Don't provide resolution

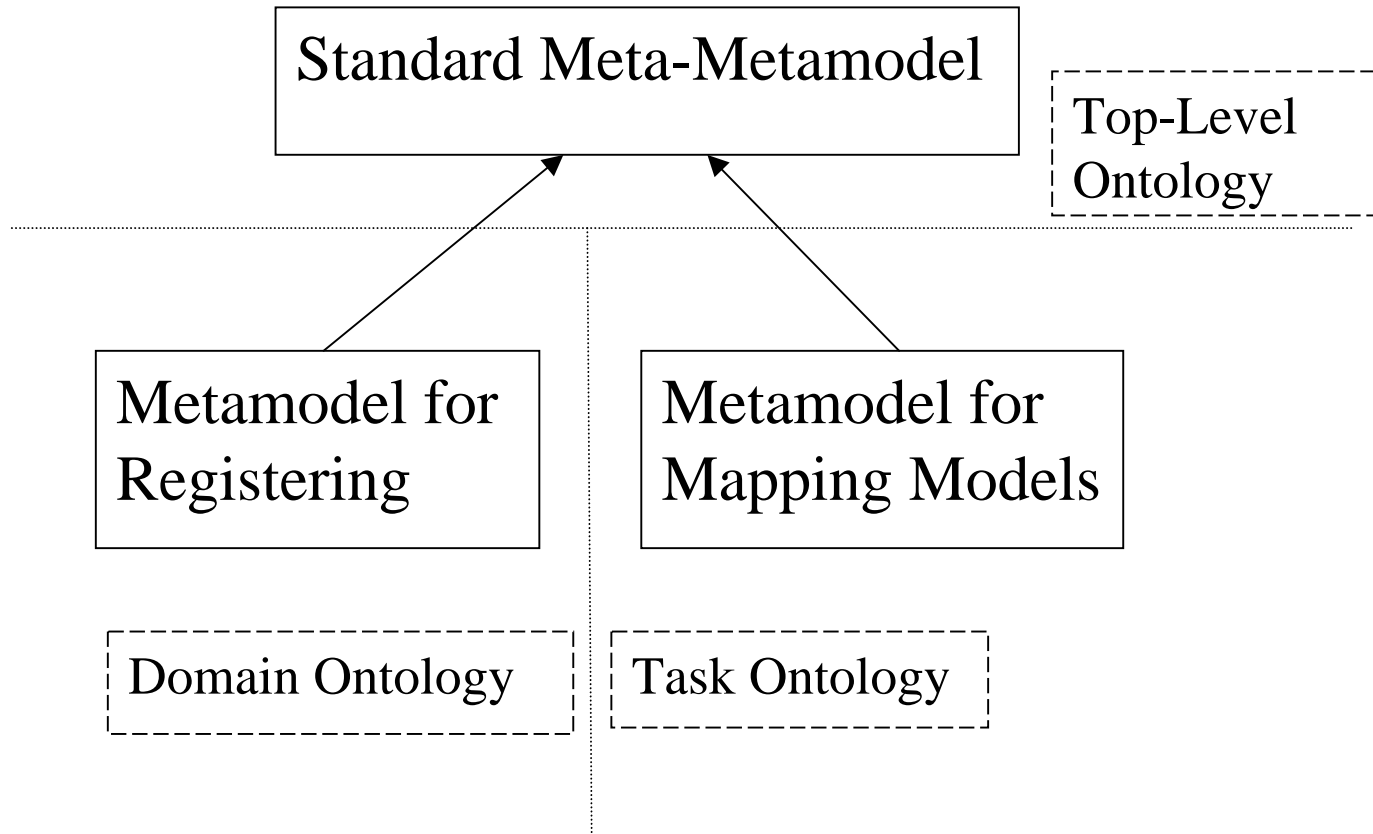
# Domain Ontology & Task Ontology

---

(1) Based on Top-Level Ontology

(2) Describe the vocabulary related to a generic domain or generic tasks or activities

# Mapping Between Meta-Framework & Ontologies



## 2. Ontological Analysis of Metamodel for Registering BO

---

2.1 Introduction of an ontological analysis

2.2 Structure of Metamodel for Registering BO

# 2.1 Introduction of An Ontological Analysis

---

- One of principled methodologies needed in the practice of ontology
- Bring order and taxonomic structure to information system
- Eliminate confusion and misunderstanding

# Meta-Properties

---

## (1) Rigidity

Rigid (+R) :A necessary property for all its instances.

Non-Rigid(-R):Not a necessary property for all instances

Anti-Rigid( $\sim$ R):An optional property for all instances

Example: PERSON (+R)

STUDENT( $\sim$ R)

# Meta-Properties(Cont.)

---

## (2) Identity

- +I : A property *carries* an identity condition(IC) iff all its instances can be (re) identified by means of a suitable sameness relation
- +O: A property *supplies* an identity condition iff such condition is not inherited by any subsuming property

Example: PERSON(+O) STUDENT(+I)

# Meta-Properties (Cont.)

---

## (3) Unity

- +U: There is a common unifying relation such that all the instances of the property are intrinsic wholes under this relation
- ~U: Every instance of the property is not an intrinsic whole

Example: Ball(+U)    Amount of matter(~U)

# Meta-Properties (Cont.)

---

## (4) Dependency

- +D: A property  $P$  is *externally dependent* on a property  $Q$  if, for all its instances  $x$ , necessarily some instance of  $Q$  must exist, which is not a part nor a constituent of  $P$
- D: Some instances of property should depend on something, but some are not
- ~ D: All instances of property depend on nothing

Example: PARENT(+D)

# Meta-Property Constraints

---

If  $P$  and  $Q$  are two properties then the following constraints hold:

(1)  $(\sim R)$  can't subsume  $(+R)$ ,

(2)  $(+I)$  can't subsume  $(-I)$ ,

(3)  $(+U)$  can't subsume  $(-U)$ ,

(4)  $(\sim U)$  can't subsume  $(+U)$ ,

(5)  $(+D)$  can't subsume  $(-D)$ ,

(6) Properties with incompatible ICs/UCs are disjoint.

# 2.2 Structure of Metamodel for Registering BO

---

2.2.1 Core

2.2.2 Extension

# Three Important Classes

---

## (1) Registry\_Object

- Provides a base class for almost all objects in the metamodel
- Each instances of Registry\_Object has a universally unique ID
- The type of Registry\_Object includes association class

# Three Important Classes(Cont.)

---

## (2) Administered\_Object

- Encapsulates its own Administration Record
- Don't include association class

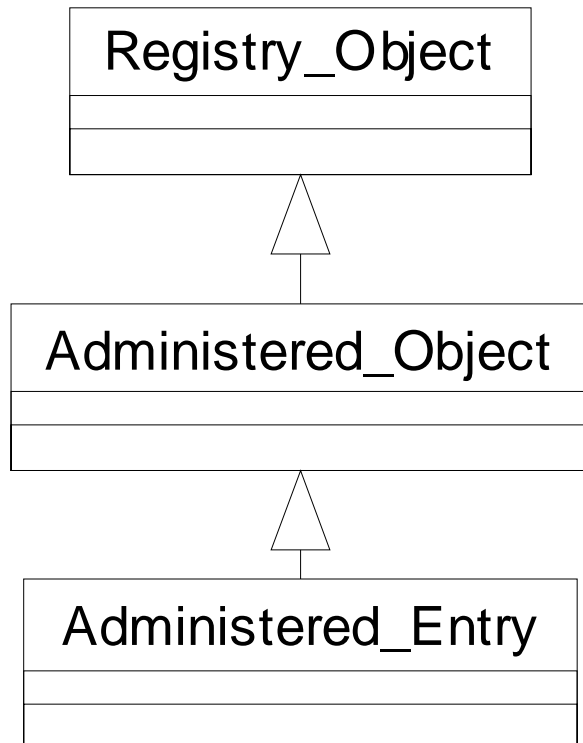
# Three Important Classes(Cont.)

---

## (3) Administered\_Entry

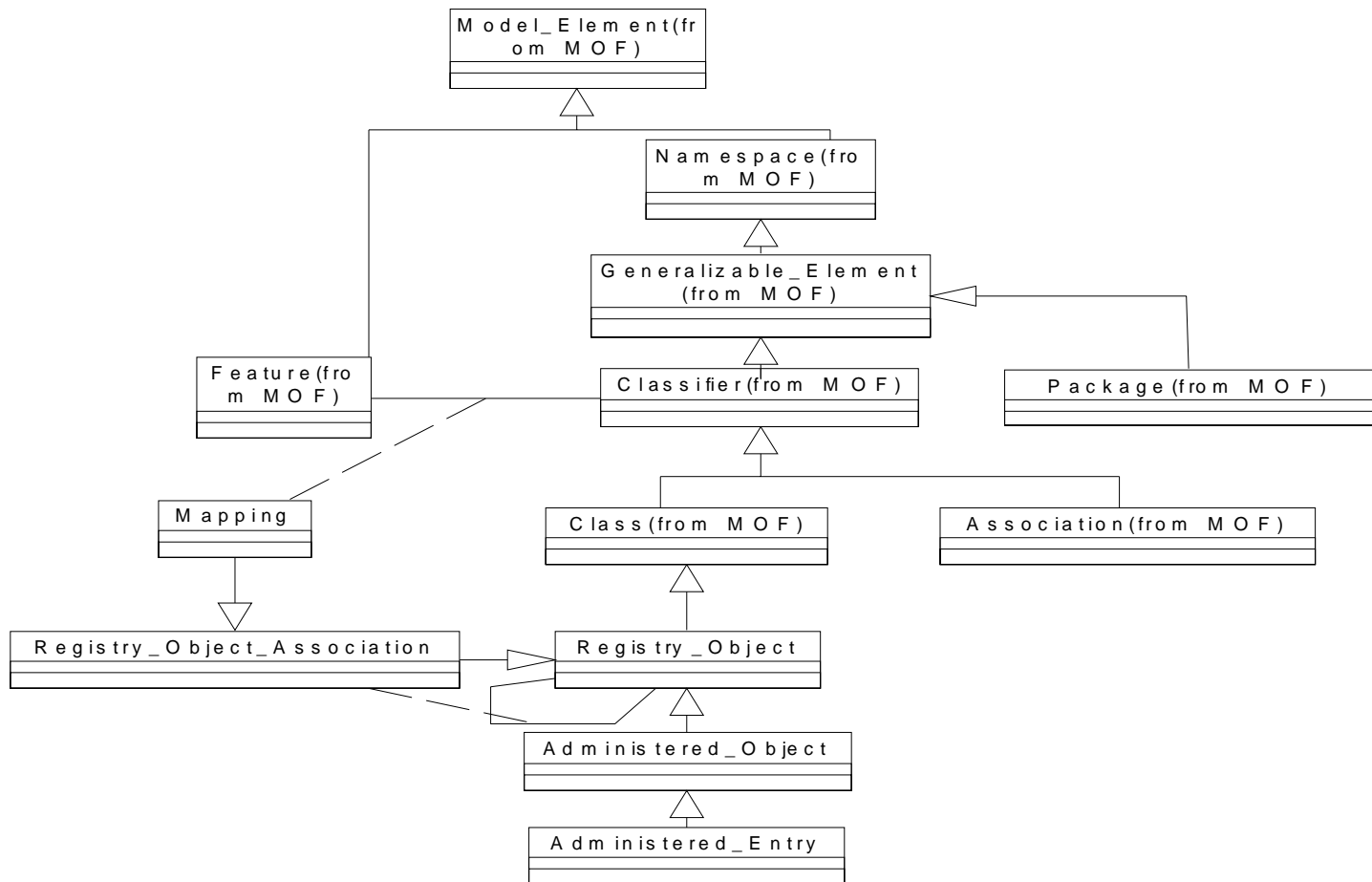
- It is used as a base class for high level coarse grained objects in the metamodel
- Have lifecycle

# Three Important Classes(Cont.)

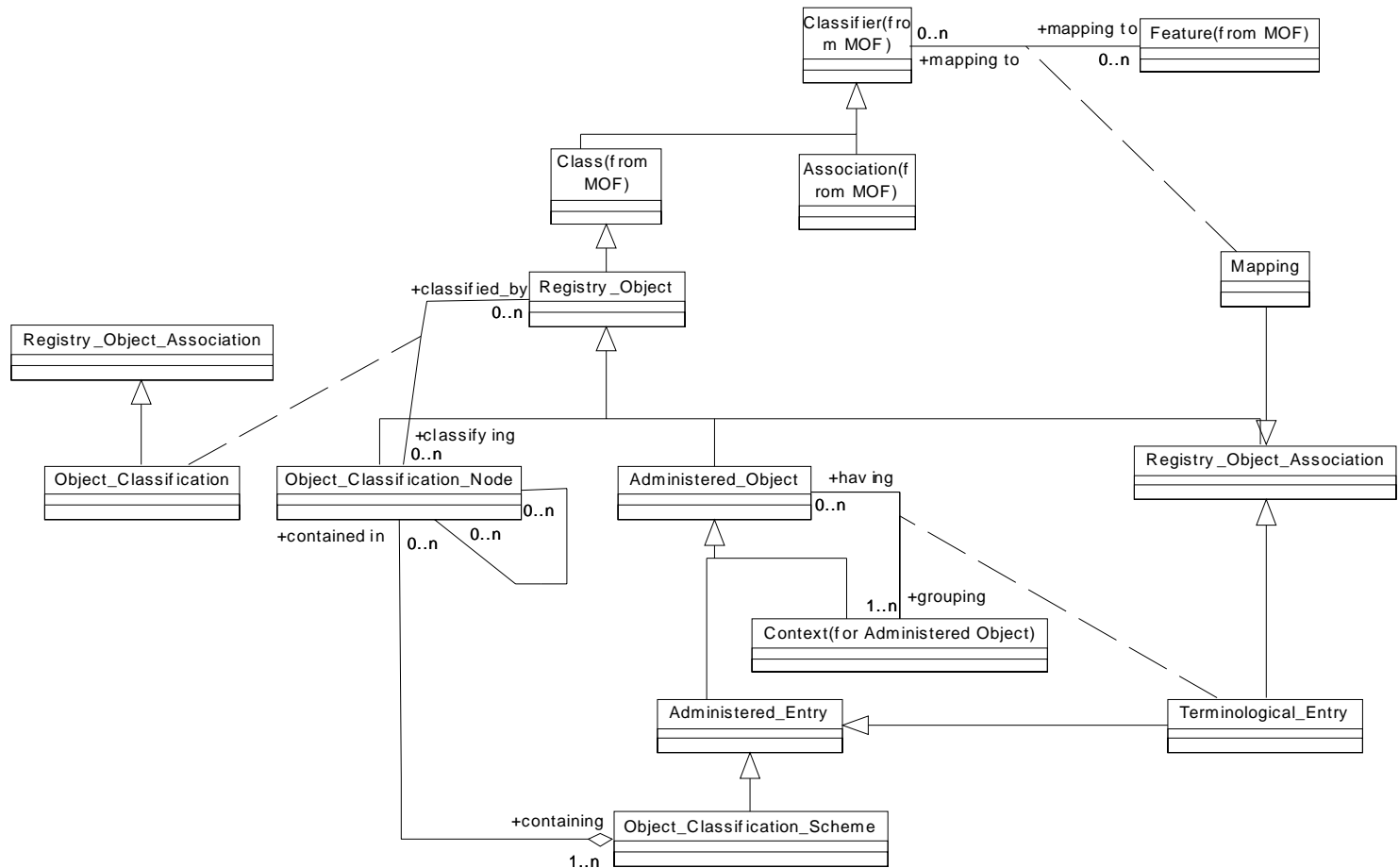


Name	Meta-Property
Registry_Object	+O+U-D+R
Administered_Object	+O+U~D+R
Administered_Entry	+O+U~D+R

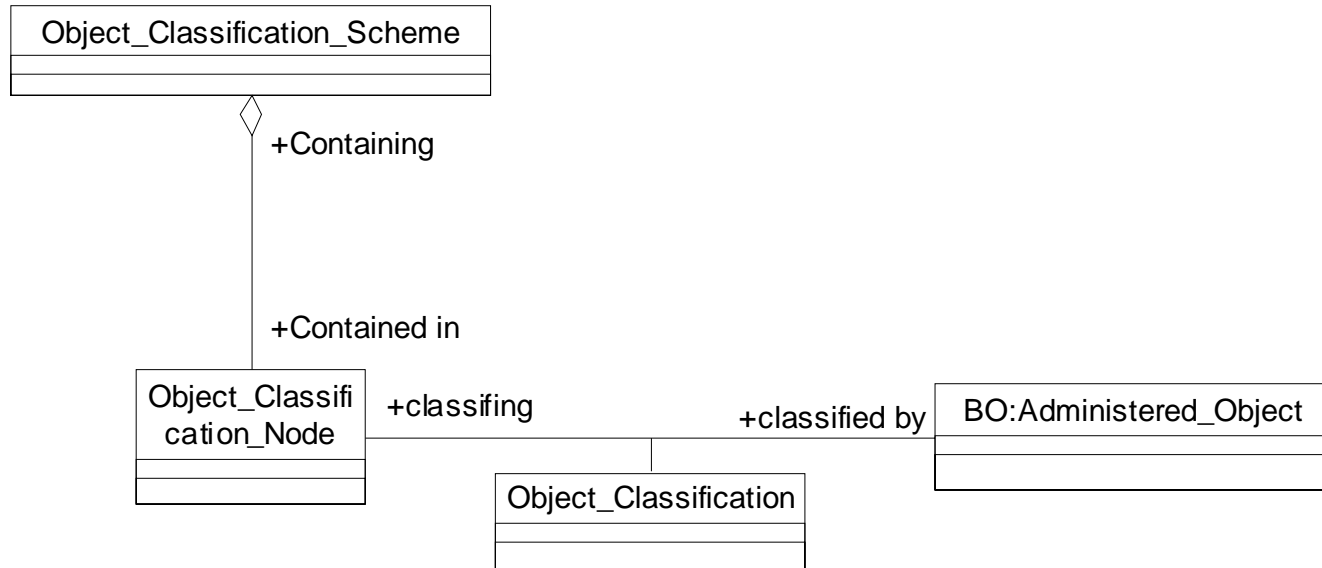
# Core Metamodel for Registering BO



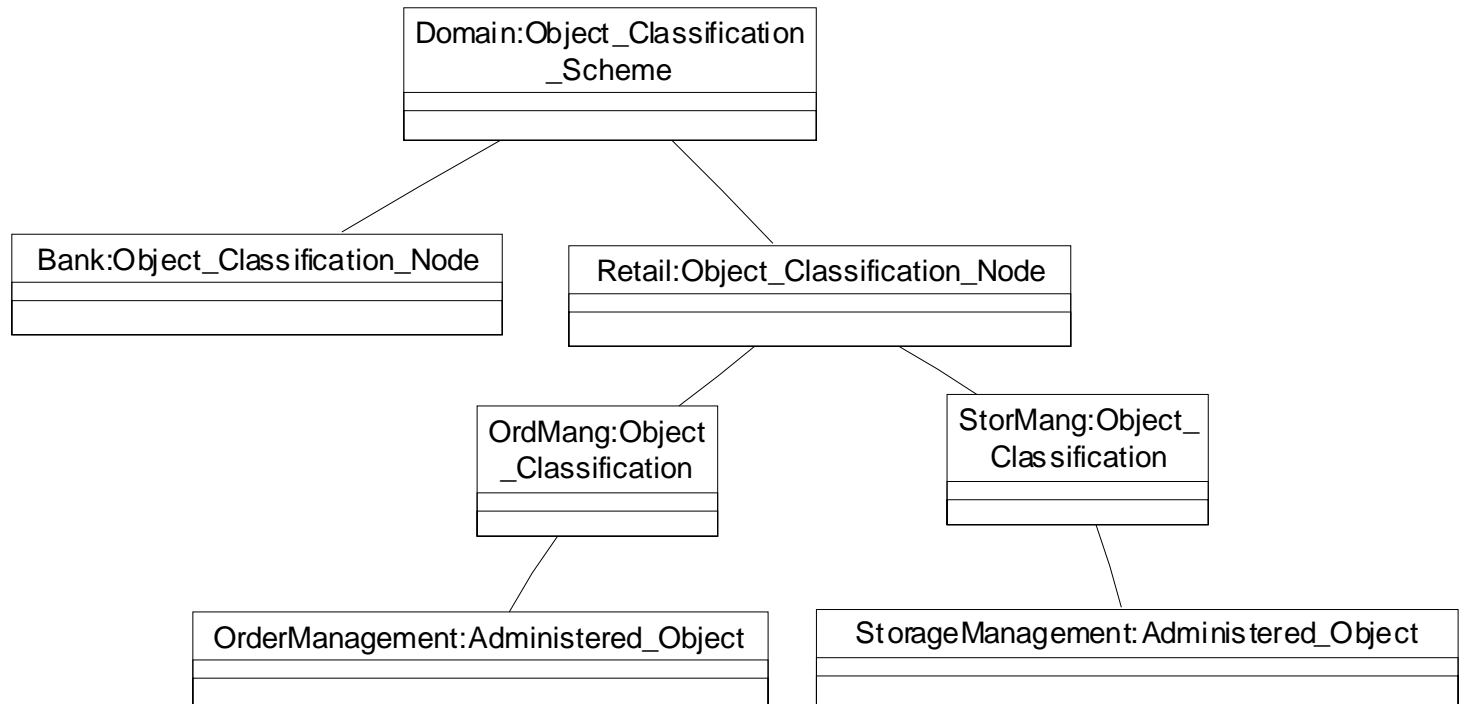
# A Detailed Metamodel



# Classification of BO



# Classification Example



# 3. Classification of BO with Taxonomic Properties

---

Why use ontology?

- Numerous properties of a BO need to be organized
- An instance of classification node may play different roles on a BO
- An instance of classification node could be shared by different BOs

# Overview of Classification Packages

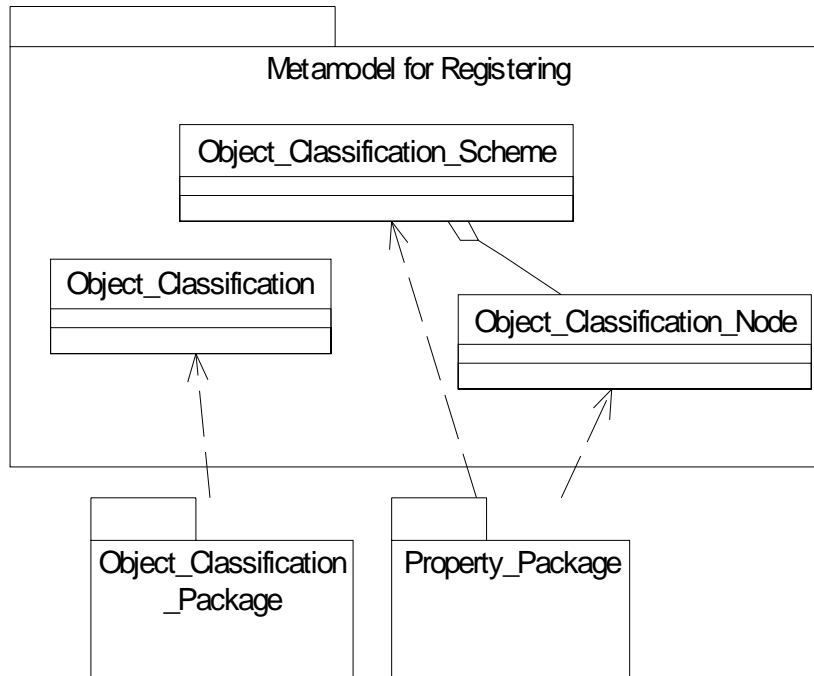


Fig. Overview of Packages

- **Property\_Package:**

Containing schemes and nodes, which make up a tree-kind taxonomy

- **Object\_Classification\_Package:**

Containing Classification, which make up a tree too.

# Property Package

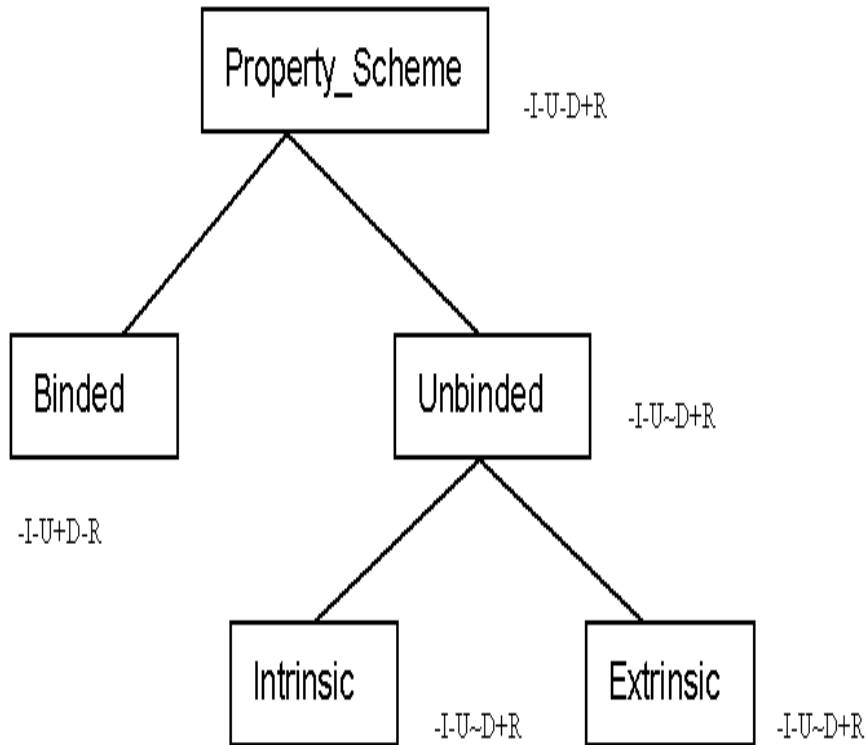
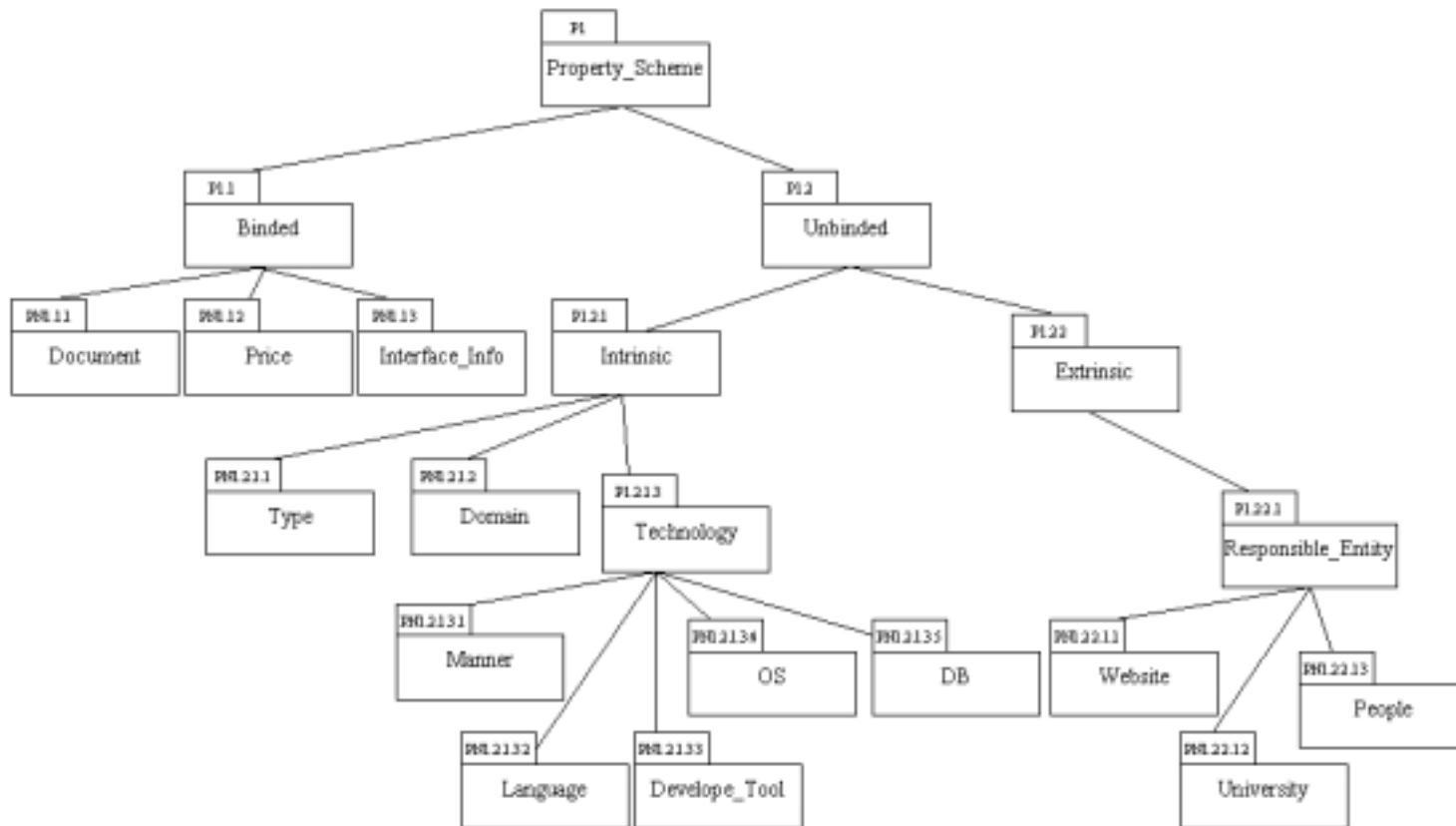


Fig. Basic Property Group

Note:

Property\_Scheme is the root of the taxonomy tree. Each tree node here is an instance of Object \_Classification\_Scheme.

# Overview of Property Taxonomy



Note:“P” means “Property” and “N” means “Object\_Classification\_Node”, i.e. leaf node.

# Classifying BO with Properties

---

- Classify the Object\_Classification according taxonomy tree of property
- Each Object\_Classification presents a role played by an Object\_Classification\_Node

# Classifying BO with Properties(Cont.)

Object\_Classification\_Package

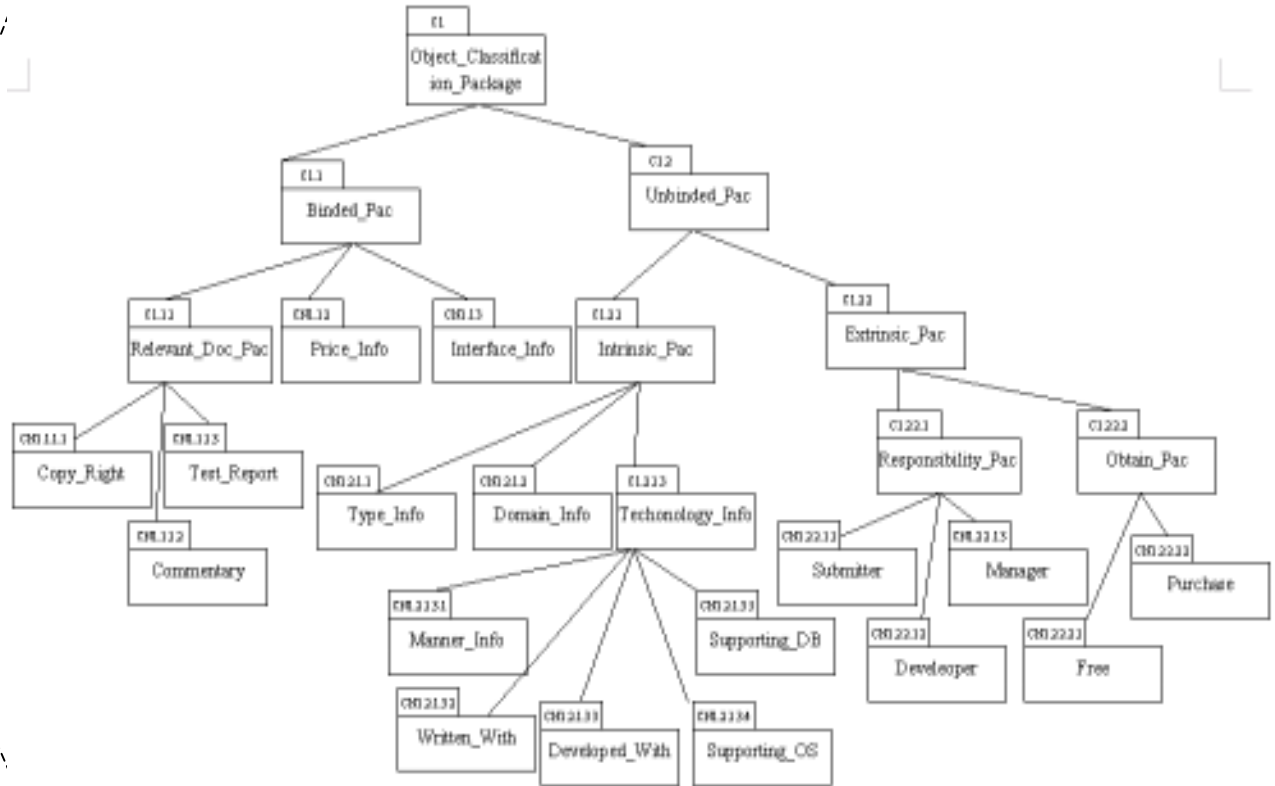


Fig. The Taxonomy of Object\_Classification

# Example

---

Assume we have registered a BO, we give a simple example to show the way classifying it

- (1) This BO is developed and submitted by the same responsible entity. These two role played on the BO are shown by two Object\_Classification instances (CN1.2.2.1.1 and CN1.2.2.1.2)
- (2) It has a relevant document which is a commentary

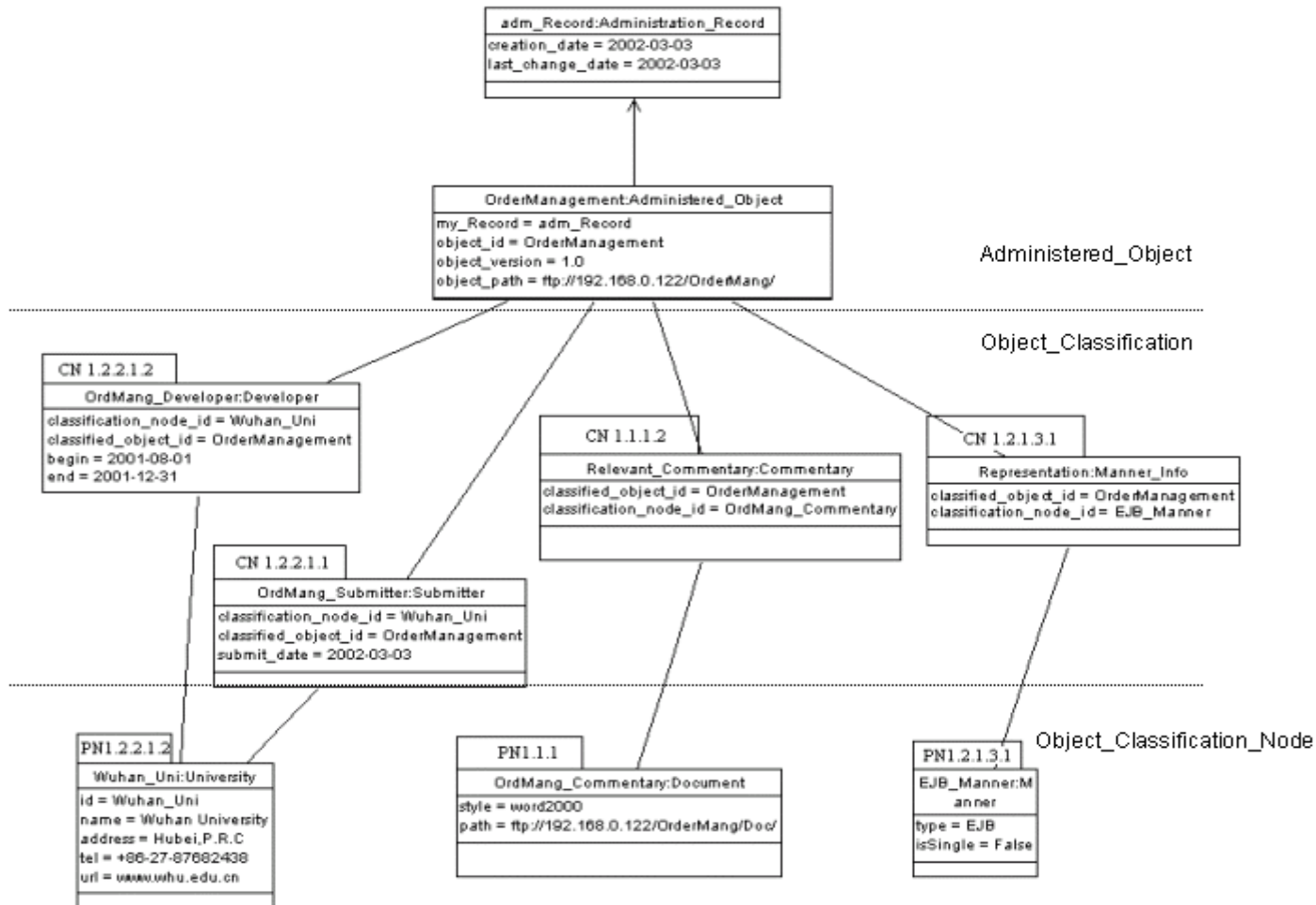
# Example(Cont.)

---

(3) Its manner is a set of EJBs

(4) Each instance of Object\_Classification or Object\_Classification\_Node is attached with a number that is the same as its class, which help to locate them in the taxonomic tree.

# Example(Cont.)



## 4. Conclusion & Future Research

---

- Ontological analysis provides a way to check the strictness of inheriting and subsumption relation in metamodel for registering BO
- “Classification Node-Classification-BO” structure offers a flexible way to classify various BO

## 4. Conclusion & Future Research(Cont.)

---

- Enrich the content of metamodel and property type.
- Strengthen the consistent check between Object\_Classification\_Node and Object\_Classification

---

Thank You!