

Date: 2002-4-24

Version: 1.0

Ontological Analysis of Metamodel for Registering Business Objects

April, 2002

He Keqing

Jing Yixin

Zhou Yi

Email: hekeqing@public.wh.hb.cn

State Key Laboratory of Software Engineering

Wuhan University

P.R.China

Contents

| | |
|---|-----|
| <i>Foreword</i> | iii |
| <i>Introduction</i> | iv |
| <i>Scope</i> | iv |
| | |
| 1. Why Ontology? | 1 |
| 1.1 What is Ontology? | 1 |
| 1.2 Metamodel Framework & Ontology | 1 |
| | |
| 2. Ontological Analysis of Metamodel for Registering BO | 3 |
| 2.1 An Ontological Analysis | 3 |
| 2.1.1 Meta-properties | 3 |
| 2.1.2 Constraints | 4 |
| 2.2 Structure of Metamodel for Registering BO | 4 |
| 2.2.1 Core Metamodel for Registering BO | 5 |
| 2.2.2 Extension to the Core | 8 |
| 2.2.3 An Example..... | 10 |
| | |
| 3. Classification of BO with Taxonomic Properties..... | 10 |
| 3.1 Ontology-Based Taxonomy of Properties..... | 11 |
| 3.1.1 Binded Properties | 12 |
| 3.1.2 Intrinsic Properties | 12 |
| 3.1.3 Extrinsic Properties | 13 |
| 3.1.4 The Overview of Property Taxonomy | 14 |
| 3.2 Classifying BO with Properties..... | 14 |
| 3.3 An Example..... | 15 |
| | |
| 4. Conclusion | 16 |
| | |
| References..... | 16 |

Table of Figures

| | |
|--|----|
| Fig.1 Metamodel Framework..... | 2 |
| Fig.2 Four kinds of Ontology and the relation among them | 2 |
| Fig.3 Mapping Between Meta-Framework & Ontologies | 3 |
| Fig.4 The relationships between Registry_Object, Administered_Object, and Administered_Entry | 6 |
| Fig.5 Core Metamodel for Registering BO | 7 |
| Fig.6 A more detailed metamodel for registering BO | 9 |
| Fig.7 Example showing classification of Administered_Objecs..... | 10 |
| Fig.8 Overview of Packages | 11 |
| Fig.9 Basic Property Group | 12 |
| Fig.10 Binded Properties | 12 |
| Fig.11 Intrinsic Properties..... | 13 |
| Fig. 12 Extrinsic Properties | 13 |
| Fig. 13 The Taxonomic Tree of Property Taxonomy..... | 14 |
| Fig. 14 The Taxonomic Tree of Object_Classification..... | 15 |
| Fig. 15 Example showing Classification with Taxonomic Properties..... | 16 |

Foreword

This document is prepared for the NWI proposal submitted by Japan National Body of ISO/IEC JTC1 SC32.

The NWI proposal is experiencing a new standard activity within ISO/IEC JTC1 SC32, which focused on developing a common framework of metamodel. The framework enables consistent registration of the various types of the business objects, extending ISO/IEC 11179 Specifications and the MOF that will be proposed by OMG to ISO JTC1 by the PAS procedure.

The metamodel for business object registering intends to provide a way describing a business object in the meta-framework. At the same time, the vocabularies for registering a BO should be organized to avoid confusion and misunderstanding, and the consistent among these vocabularies is required. For this goal, we apply an ontological analysis in the NWI proposal.

Introduction

Since the progress of Internet and various resources appear in World Wide Web, the development of software, especially in the business domain, has been improved. The way of producing an application has changed from code programming to a high-level software development applying the cooperation among the business objects (BO) that not only frees the programmers from hard repeating work but enhances efficiency and quality in the whole developing process. These business objects come from many business domains in the different manners, such as protocols, process models etc. To cope with requirements for the harmonized exchanging of the business objects, we first need to establish a metamodel to register and manage them, and describe the property of BO with a methodology that provides the hierarchy and order of the structure.

Nowadays, Ontology, a philosophical concept, is becoming increasingly popular in practice, such as modeling engineering, AI, knowledge system and so on. Ontology offers the benefit to providing a commitment among the research contexts so that it can eliminate the conflict and confusion due to the different understanding. To bring an order to BO registering, we apply a principled ontological analysis on the structure of metamodel and propose the ontology classification of BO in this paper.

Scope

In section 1, we discuss why ontology can contribute to the establishment of framework and the relationship between ontology and the metamodel framework. In section 2, an ontological methodology is introduced and adopted to construct the registering metamodel. In section 3, we present the implement of the classification of business objects in the method mentioned in the above section. At the end of the document, we conclude our work and advance the future research direction.

1. Ontology Method?

1.1 What is Ontology?

Ontology is the branch of metaphysics that deals with the nature of being. The subject of ontology is to study the categories of things that exist or probably exist in some domain. The result of such a study, called *ontology*, is a catalog of the types of things that are assumed to exist in a interesting domain D from the perspective that is expressed in some language L for the purpose of talking about the domain D . By itself along, logic says nothing about anything. However, the logic combining with ontology can provide a language description in which the relationships among the entities in the domain of interest may be expressed.

In conceptual modeling world, although what is ontology is still a debated issue [1], the analysis based on the ontology is playing a more and more important role. Modeling engineering has got the benefit a lot in the ontology method observing the world, Using the ontology description, the capability of interaction between different knowledge systems has been enhanced. The research and practice of the ontology methodology clarify the order and structure of the information; as a result, the conflict and confusion of understanding it can be avoided.

1.2 Metamodel Framework & Ontology

The metamodel framework itself is a knowledge system for registering and exchanging the business objects. The business objects may be shared by the users those who intent to interchange each other the enterprise objects of their organizations. In the proposal submitted by Japan National Body of ISO/IEC/JTC1 SC32/WG2 the metamodel framework is reckoned on consisting of a standard meta-metamodel, a metamodel for registering the model elements and a metamodel for describing the mapping relationship [2].

The standard meta-metamodel is an extension of MOF that provides a facility for metamodeling. This meta-metamodel describes the general concepts needed by building the registering and mapping metamodels. The general concepts abstracted by the meta-metamodel are non-relevant to the concepts of registering and mapping. The concepts of level-metamodel have to be defined by the meta-metamodel, i.e. the meta-metamodel is a specification of conceptualization [1]. In this paper, based on the standard meta-metamodel, the registering and mapping metamodel would be built and implemented by the ontology mechanism.

The metamodel for registering objects provides a schematic metamodel to enable those who have to develop standards, to access and locate appropriate standards to be conformed. In this metamodel, different concepts are defined for BO registering and the properties required to classify the objects reside in a taxonomic tree, which describes the registered object from various viewpoints.

Metamodel for mapping models is the other domain-based extension of standard meta-metamodel. It provides a unified metamodel to realize the appropriate mapping between models which are developed with different modeling facilities or modeling constructs, which is necessary for approach the goal of interchanging BOs which are developed under different standards but have similar function.

The relationship among the three (meta-)metamodels above is shown in Fig.1.

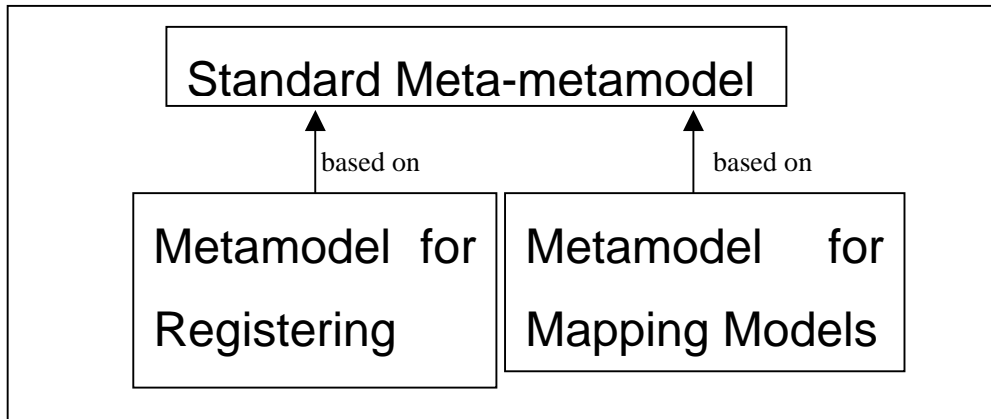


Fig.1 Metamodel Framework

We found a similar structure to Fig.1 in [3]. In Fig.2, there are four different kinds of ontology, which are divided into three levels. On the top of structure is Top-level ontology, which only presents the most general concepts, such as relations, time, space etc. It has nothing to do with any domain and doesn't provide resolution to any problem. Based on the Top-level ontology, domain ontology and task ontology are conceptually defined in the second level to specify Top-level ontology to a special domain and some particular action in that domain. They are considered as a vocabulary or knowledge repository of the domain they belong to. At the bottom of the structure is the application ontology. It focuses on the requirements of application following the rules in the ontologies above it.

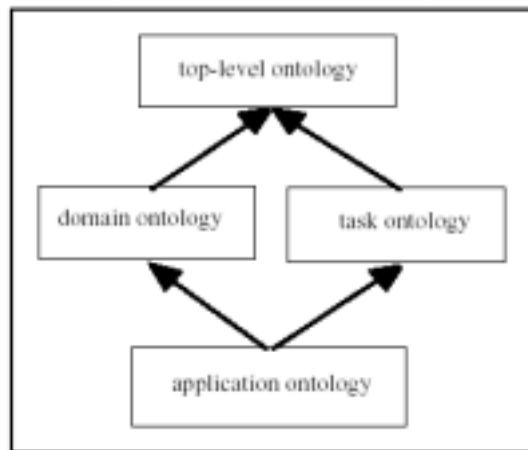


Fig.2 Four kinds of Ontology and the relation among them

Comparing Fig.1 and Fig.2, we clearly found the similarity between them. Although ontology is not explicitly applied in the metamodel framework, the mapping relationship between them is obvious, Fig.3.

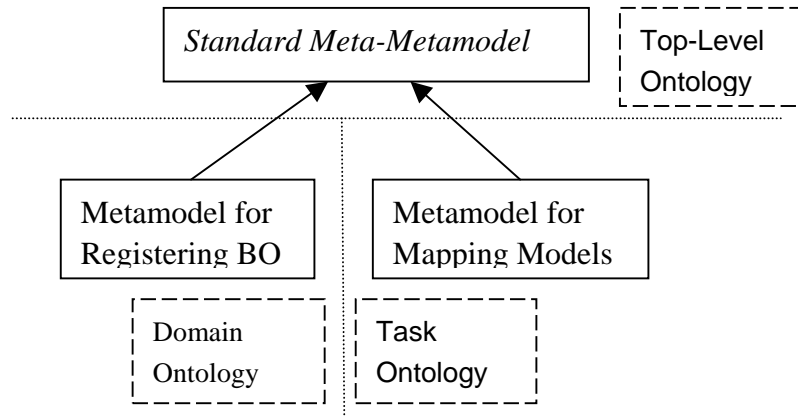


Fig.3 Mapping Between Meta-Framework & Ontologies

From the analysis above, we find standard meta-metamodel plays the role of Top-level ontology, while registering and mapping meta-models are domain and task ontology respectively belonging to the second level. Thus we'd like to construct metamodel framework with the help of ontology. Meanwhile, each (meta-)metamodel itself is a knowledge system where ontology can help, we believe that ontological analysis is important to not only the whole metamodel framework but also to establishment of each (meta-)metamodel.

2. Ontological Analysis of Metamodel for Registering BO

Taking the analysis above into account, metamodel for registering BO belongs to domain ontology. As a specification of conceptualization in a particular domain, principled methodology is needed to insure the semantic rigor and cognitive plausibility [4]. The structure of the metamodel would be ordered and refined with the help of ontological analysis. We first introduce a methodology supporting ontology before we apply it to build metamodel for registering BO.

2.1 An Ontological Analysis

During the process of building metamodel framework, we apply a formal tool of ontological analysis, which is described by Nicola Guarino and Christopher Welty in paper [5] to eliminate the confusion and poor quality in the practice of ontology. In that paper authors showed how a rigorous analysis of the ontological meta-properties of taxonomic nodes can help using the subsumption relation in a disciplined way. By this way, we hope to test the structure of metamodel framework and use this analysis as a guideline for us to classify the properties of business objects. Here we give a brief description of the methodology, more detailed definition could be found in relevant papers.

2.1.1 Meta-properties

The methodology is based on four fundamental ontological notions: Rigidity, Identity, Unity, and Dependence. The behavior of a property is represented with respect to these notions by means of a set of meta-properties. These meta-properties impose some constraints on the way subsumption is used to model a domain. Primitive meta-properties are denoted by bolded letters preceded by the sign “+”, “-” or “~” corresponding to carrying the meta-property, not carrying the

meta-property, and anti the meta-property.

- R Rigidity

A rigid property (+**R**) is necessary for all its instances. If it is not a necessary property of all its instances, we called it non-rigid property (-**R**). A stronger restriction is that a property is optional for all its instances. In this case, it is an anti-rigid property (~**R**). For example, a PERSON is a rigid property because if x is an instance of PERSON, it must be an instance of PERSON in every possible world, while STUDENT is ~**R** since any individual of student may not be a student in some situations, e.g. graduating.

- Identity

A property carries an identity condition (+**I**) iff all its instances can be (re) identified by means of a suitable sameness relation. A property supplies an identity condition (+**O**) iff such condition is not inherited by any subsuming property. A property marked with +**O** must be a +**I** property. For example PERSON is usually considered as supplying an identity condition, while STUDENT just inherits the identity condition of PERSON without supplying any further identity conditions. Since what we consider as an identity condition should be a global condition, student's registration number, which only has meaning in a certain studenthood experience, is not a further identity condition.

- Unity

A property P is said to carry unity condition (+**U**) iff there is a common unifying relation R such that all the instances of P are intrinsic wholes under R . A property carries *anti-unity* (~**U**) if all its instances can possibly be non-wholes. For instance, a ball is +**U**, while food is ~**U**.

- Dependency

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 x . For example PARENTS is +**D** since only some individual of CHILD must exist.

2.1.2 Constraints

Following the definitions above, we describe the constraints imposing on the meta-properties.

We take P and Q to be arbitrary properties.

- (1) (~**R**) can't subsume (+**R**),
- (2) (+**I**) can't subsume (-**I**),
- (3) Properties with incompatible ICs are disjoint ,
- (4) (+**U**) can't subsume (-**U**),
- (5) (~**U**) can't subsume (+**U**),
- (6) Properties with incompatible UCs are disjoint,
- (7) (+**D**) can't subsume (-**D**).

2.2 Structure of Metamodel for Registering BO

Before we build metamodel for registering BO, a standard meta-metamodel is needed on which our metamodel is based. MOF [6] standardized by OMG defines a common abstract syntax for defining metamodels. We take MOF as our standard meta-metamodel although its specification would be extended to enhance the classification and registration of metmodels. Meanwhile some work have be done to build some metamodels for registration, such as ISO/IEC/JTC1 FDIS 11179-3 [7] and Registry Information Model [8] (we call it Information

Model for convenience in this paper) adopted by ebXML/OASIS. We try to integrate the general concept in these two metamodels with the methodology mentioned above to perform a refined registering metamodel. In more detail, we will discuss how to classify business objects.

2.2.1 Core Metamodel for Registering BO

Both 11179 - 3 and Information Model are involved in constructing metamodel for registering metadata in form of conceptual data model. In their specifications some important classes relevant to registration are defined. We would use the ontological analysis to find the relationship between them according to their definitions.

One of these classes is Adminstered_Item in the structure of 11179-3. Adminstered_Item is a registry item for which administrative information is recorded in an Administration_Record, which is another class in metamodel. That means each instance of Adminstered_Item encapsulates its own Administration_Record. According the types of Adminstered_Item given by 11179-3, association class between classes doesn't belong to this concept.

Registry_Object is presented in the information model. It provides a base class for almost all objects in the model. Each instances of Registry_Object has a universally unique ID and the type of Registry_Object includes both entity and association class.

Another important class is Registry_Entry which is a subclass of Registry_Object. Since Registry_Object provides minimal metadata for an object, Registry_Entry, which always plays the role of a container, is used to describe a set of additional metadata for it. Now we can give the meta-properties for the classes above, see Tab.1.

| Name | Meta-Property | Notes |
|------------------|---------------|--|
| Adminstered_Item | +O+U~D+R | IC: same ID, UC: functional relationship |
| Registry_Object | +O+U-D+R | Everything in information model is Registry_Object, and a instance which is association class type must depend on some objects else. |
| Registry_Entry | +O+U~D+R | IC: same ID, UC: functional relationship |

Tab. 1 Meta-Properties of Adminstered_Object, Registry_Object and Registry_Entry

Considering the meta-properties constraints and the definition of each class, we put these concepts from different metamodels together with a generalization relation. We think the registry item mentioned in defining Adminstered_Item has the same meaning as Registry_Object, so the former should be a child of the latter. We keep the relationship between Registry_Object and Registry_Entry and see Registry_Entry as a special Adminstered_Item. For the consistency of name, we change the name of Adminstered_Item into Adminsitered_Object and Registry_Entry into Adminstered_Entry. See Fig.4

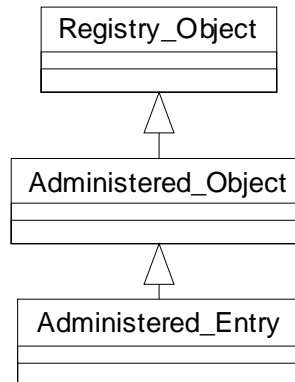


Fig.4 The relationships between `Registry_Object`, `Administered_Object`, and `Administered_Entry`

Fig.4 presents three key classes in metamodel for registering BO. We put this architecture into the MOF environment, thus we get the core of metamodel for registering BO. See Fig.5.

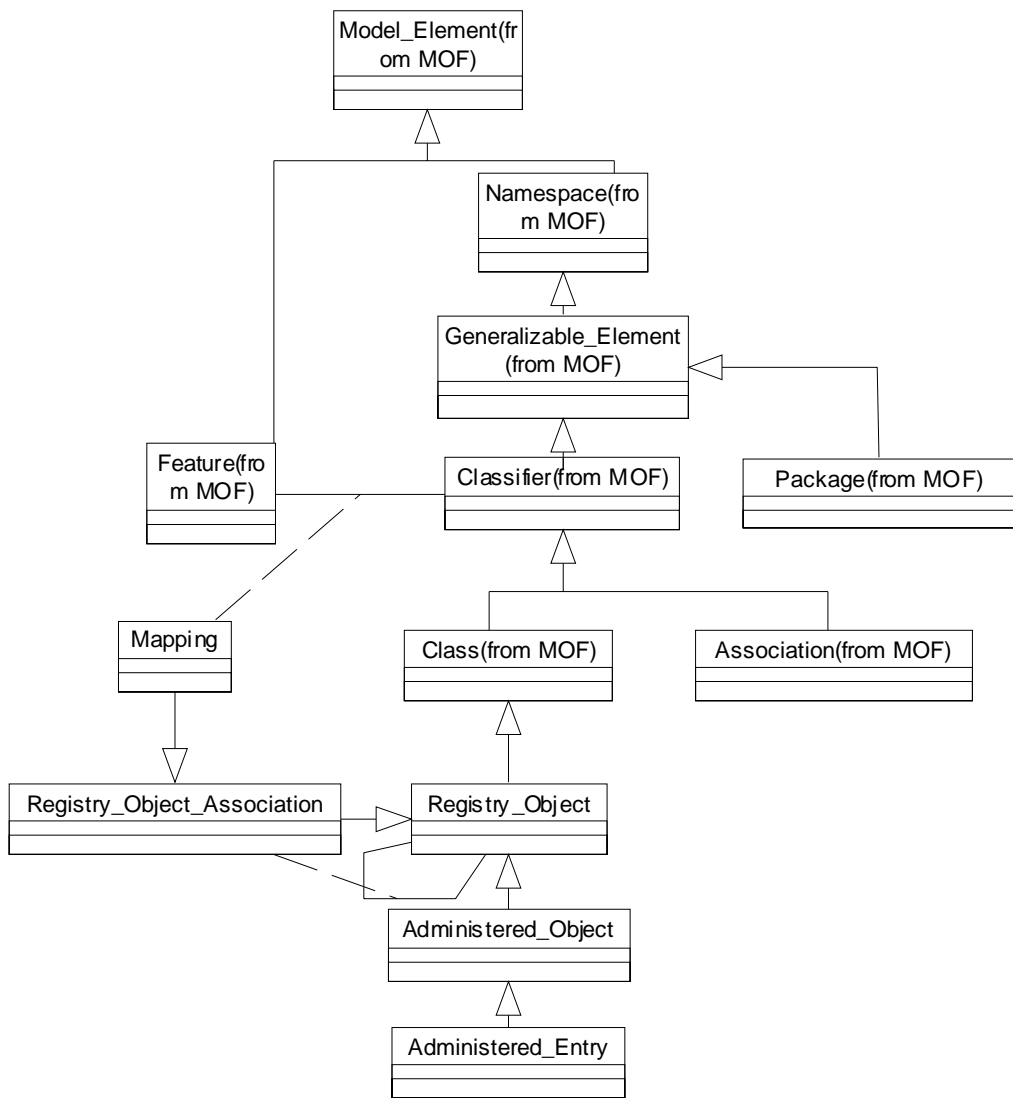


Fig.5 Core Metamodel for Registering BO

(1) Registry_Object

Registry_Object is the root class of the most classes in metamodel. A unique ID distinguishes an instance of Registry_Object with others under the same namespace. If an object can't be identified through this way, it belongs to Association or something else.

Attribute

| Attribute name | Occurrences | Datatype |
|----------------|---------------------------------|----------|
| description | Zero or One per Registry Object | String |
| object_Id | One per Registry Object | String |
| object_Name | Zero or One per Registry Object | String |

(2) Registry_Object_Association

Two or more Registry_Objects can be tied together with one or more

Registry_Object_Association, which itself is a Registry_Object and is the ancestor of all association classes in metamodel.

| Attribute name | Occurrences | Datatype |
|------------------------------|-------------------------------------|----------|
| association_Type_Description | One per registry object association | String |
| associated_Object_ID | Two per registry object association | String |

(3) Administered_Object

Every Administered_Object owns an Administration_Record. Besides the information stored in the record, Administered_Object capsulate some basic attributes.

| Attribute name | Occurrences | Datatype |
|---|-----------------------------|-----------------------|
| administered_object_administration_record | One per Administered Object | Administration_Record |
| object_ID | One per Administered Object | String |
| object_Path | One per Administered Object | String |
| object_Version | One per Administered Object | String |

(4) Adminitered_Entry

As a special Administered_Object, Adminitered_Entry has a lifecycle. It is used as a base class for high level coarse grained objects in the metamodel. They typically have relatively fewer instances but serve as a root of a composition hierarchy consisting of numerous objects that are sub-classes of Registry_Object but not Adminitered_Entry.

| Attribute name | Occurrences | Datatype |
|----------------|----------------------------|----------|
| major_Version | One per Administered Entry | Integer |
| minor_Version | One per Administered Entry | Integer |
| stability | One per Administered Entry | String |

(Pre-defined types of stability is "Dynamic", "Dynamic-Compatible", and "Static", which detailed definition could be found in Information Model.)

(5) Mapping

Mapping is a special association class, which provides the disciplines of mapping between classifiers, classifier and feature, and features. How to realize this function is out of the scope of this paper.

| Attribute name | Occurrences | Datatype |
|----------------|-----------------|------------|
| source_ID | One per Mapping | String |
| target_ID | One per Mapping | String |
| function | One per Mapping | Expression |

2.2.2 Extension to the Core

Basing on the core metamodel above, we integrate the general concepts of 11179-3 and Information Model for a domain-aimed metamodel for registering BO. This detailed metamodel, shown in Fig.6, is not a complete one for the numerous contents involved in registering a BO. New element could and should be added in the architecture specified by the core. We just describe the important classes which we think benefit the management and registration of BO. Some of detailed definition could be found in 11179-3 or Information Model Specification.

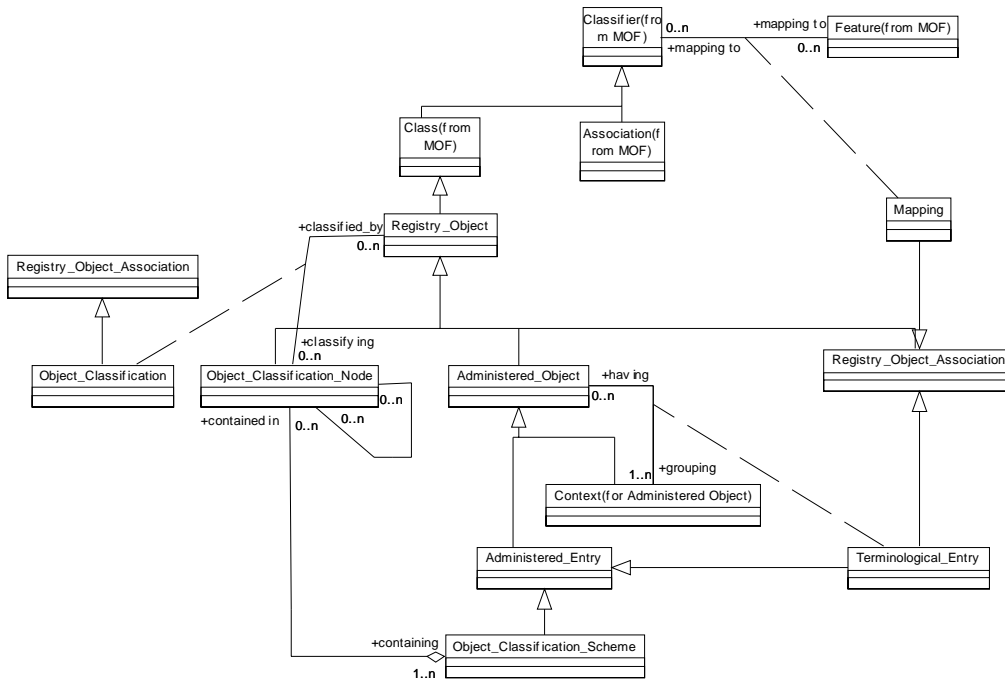


Fig.6 A more detailed metamodel for registering BO

(1) Context (for Administered Object)

Context is a class comes from 11179-3. Each Administered_Object is named and defined within one or more Contexts. A Context defines the scope within which the subject data has meaning. A Context may be a business domain, an information subject area, an information system, a database, file, data model, standard document, or any other environment determined by the owner of the registry.

| Attribute name | Occurrences | Datatype |
|---|-------------------------|-------------------------|
| context description | One per Context | String |
| context_description_language_identifier | Zero or one per Context | Language Identification |

(2) Terminological_Entry

A Terminological Entry applies to an Administered_Object in a particular Context. It provides a grouping of Designations and Definitions, which contained in a Language Section, allowing the Administered_Object to be named and defined within the Context in multiple languages. An Administered_Object may have one or more Terminological Entries each in a particular Context. Each Terminological Entry contains one or more Language Sections, so it is a sub-class of Administered_Entry and a sub-class of Administered_Object_Association.

| Attribute name | Occurrences | Datatype |
|----------------|------------------------------|----------|
| context_ID | One per Terminological Entry | String |
| object_ID | One per Terminological Entry | String |

(3) Object_Classification_Scheme

Object_Classification_Scheme is a generalizable container, which could contain other schemes in smaller grain or some Object_Classification_Nodes. We will use ontological analysis later to discuss how these scheme and nodes should be organized.

| Attribute name | Occurrences | Datatype |
|---|--|-----------------------|
| parent_ID | Zero or One per Object Classification Scheme | String |
| classification_scheme_administration_record | One per Object Classification Scheme | Administration Record |
| classification_scheme_type_name | One per Object Classification Scheme | String |

(4) Object_Classification_Node

An Object_Classification_Node present an individual item within a scheme, its instance is a property of a BO and its parent may be another node or the scheme it belongs to.

| Attribute name | Occurrences | Datatype |
|----------------|------------------------------------|----------|
| parent_ID | One per Object Classification Node | String |

(5) Object_Classification

An Administered_Object should be attached with its own properties, which are instances of Object_Classification_Nodes, through Object_Classifications, and the nodes' role is presented from the type of Object_Classification. We will discuss it in detail in following section.

| Attribute name | Occurrences | Datatype |
|------------------------|-------------------------------|----------|
| classification_Node_ID | One per Object Classification | String |
| classified_Object_ID | One per Object Classification | String |
| is_Binded_Property | One per Object Classification | Boolean |

2.2.3 An Example

Fig. 7 illustrate how a Administered_Object is classified with Object_Classification_Node through a Object_Classification

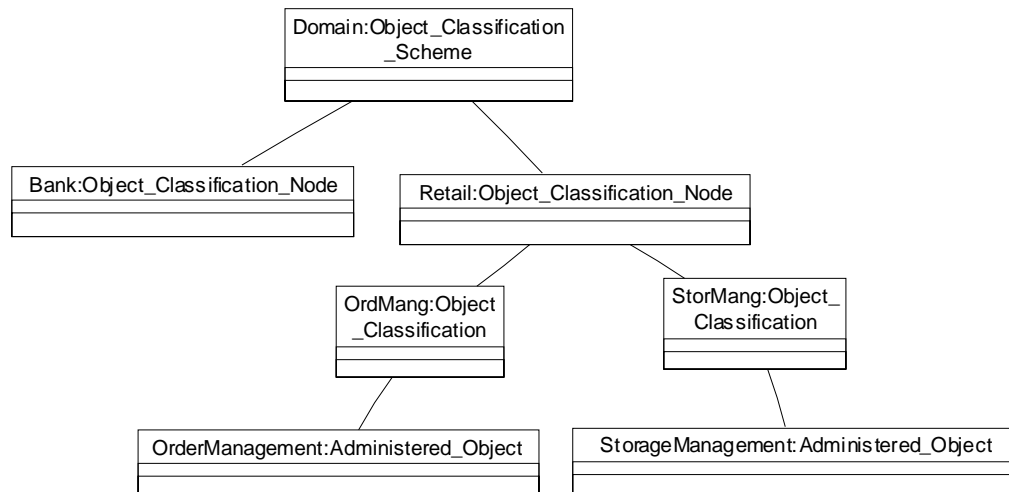


Fig.7 Example showing classification of Administered_Objects

3. Classification of BO with Taxonomic Properties

For the organization of the numerous properties of BOs, we draw out the Scheme, Node and

Classification to make up two packages, see Fig.8.

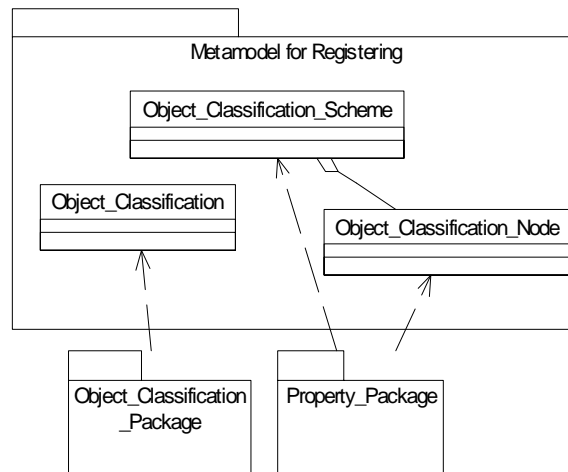


Fig.8 Overview of Packages

Property_Package contains Schemes and Nodes, which make up a tree-kind taxonomy, and each instance of Object_Classification_Node present a property of a BO. On the other hand, Object_Classification_Package contains Classifications, which make up a tree too.

3.1 Ontology-Based Taxonomy of Properties

Without a methodology to guide the structure of properties of BO, we can only put various properties together according to our experiences, even relying on our feeling! Here we apply the concept of meta-properties in ontological analysis mentioned above to describe the feature of different properties which are present with the instance of Object_Classification_Node within a particular Object_Classification_Scheme. We organize these schemes and nodes into a tree-kind taxonomy, whose leafs are classification node and the others are classification scheme (We remark the leaf with a N in the tree). In our opinion, all properties could be divided into two groups. One is Binded, and the other one is Unbinded. See Fig.9. The root is an instance of Object_Classification_Scheme. A Binded property, such as price or interface information, is the one that must depend on a business object, i.e. +D. If a business object doesn't exist, a Binded property which lose its host should be managed specially, may be collected in a history record. On the other hand, Unbinded properties, which are -D, could be split into smaller groups, Intrinsic and Extrinsic. The property belonging to Intrinsic means its value reflects the character of BO, any change of the value would influence where and how the BO is used by who. Extrinsic property is just some additional imformations of a BO, for example, a submitter.

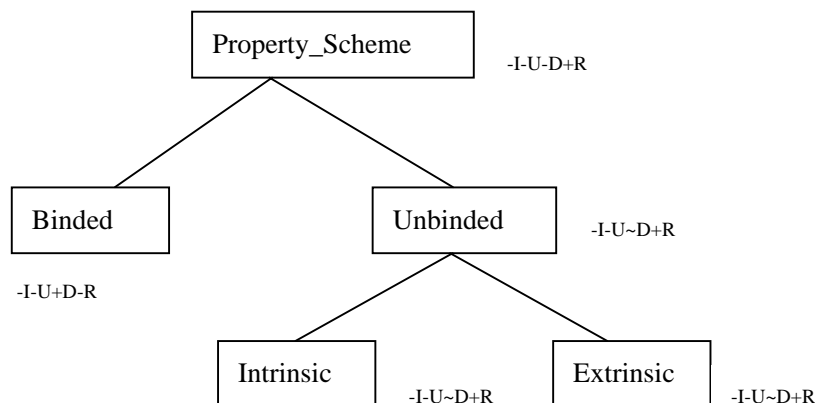


Fig.9 Basic Property Group

3.1.1 Binded Properties

Fig.10 presents some examples of Binded properties, and following tables give a more detailed definition respectively.

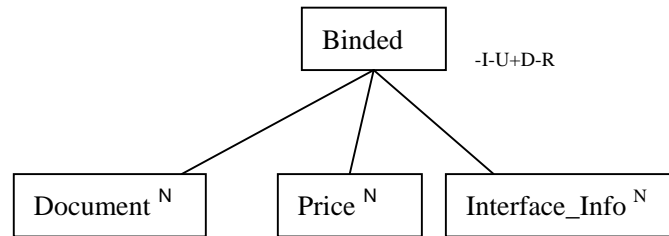


Fig.10 Binded Properties

(1) Document: Any of document relevant to a BO

| Attribute name | Occurrences | Datatype/Examples |
|-------------------|------------------|-------------------|
| document_ID | One per Document | String |
| document_Name | One per Document | String |
| document_Format | One per Document | String/word2000 |
| document_Location | One per Document | URL |

(2) Price: The price of a BO.

| Attribute name | Occurrences | Datatype/Examples |
|----------------|---------------|-------------------|
| Currency | One per Price | String/USDollar |
| Value | One per Price | String |

(3) Interface_Info: The information about BO's interface.

| Attribute name | Occurrences | Datatype/Examples |
|----------------|------------------------|-------------------|
| Input | Zero or One per Status | String/a date |
| Output | Zero or One per Status | String/order list |

3.1.2 Intrinsic Properties

Fig.11 presents some examples of Intrinsic properties, and following tables give a more detailed definition respectively. A Technology property describe the technological feature of a BO, such as manner, language written with, OS supported, and so on.

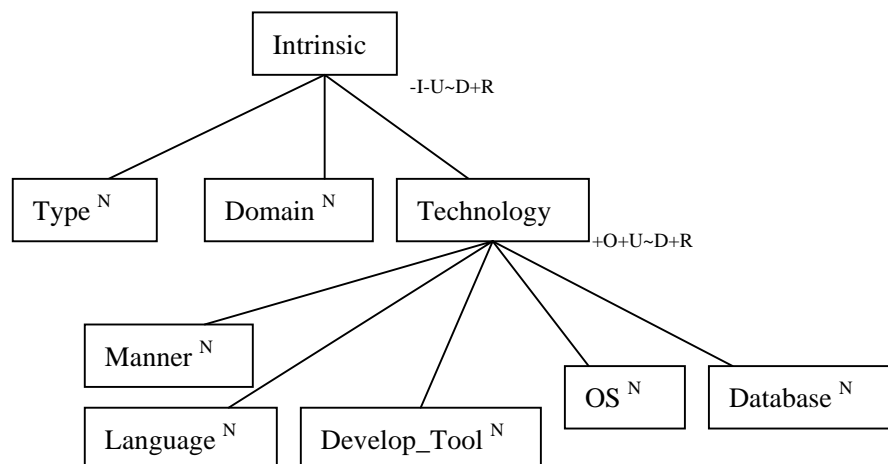


Fig.11 Intrinsic Properties

(1) Type: The type of a BO.

| Attribute name | Occurrences | Datatype/Examples |
|------------------|----------------------|-------------------------------|
| type_Name | One per Type | String/Business Process Model |
| type_Description | Zero or one per Type | String |

(2) Domain: The domain where a BO can work.

| Attribute name | Occurrences | Datatype/Examples |
|--------------------|------------------------|-------------------|
| domain_Name | One per Domain | String/Bank |
| domain_Description | Zero or one per Domain | String |

(3) Manner: The manner a BO presents.

| Attribute name | Occurrences | Datatype/Examples |
|----------------|------------------------|-----------------------|
| manner_Type | One per Manner | String/EJB, UML model |
| version | Zero or one per Manner | String |

3.1.3 Extrinsic Properties

Fig.12 presents some examples of Extrinsic properties, and following tables give a more detailed definition respectively. Responsible_Entity include those have responsible duty to a BO.

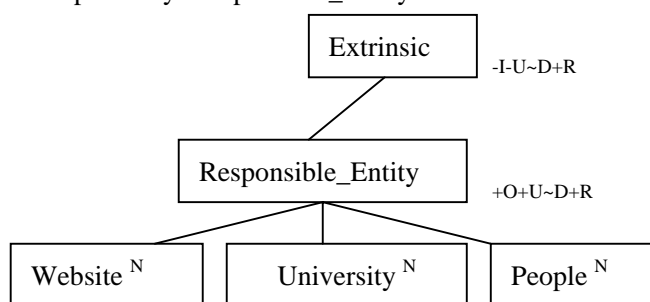


Fig. 12 Extrinsic Properties

(1) Website

| Attribute name | Occurrences | Datatype/Examples |
|----------------|-------------------------|-------------------|
| website_Name | One per Website | String |
| website_URL | One per Website | String |
| webmaster_Info | Zero or one per Website | String |

(2) University

| Attribute name | Occurrences | Datatype |
|----------------|------------------------------|----------|
| name | One per Organization | String |
| address | One per Organization | String |
| tel | One per Organization | String |
| fax | One per Organization | String |
| URL | Zero or One per Organization | String |

(3) People

| Attribute name | Occurrences | Datatype |
|----------------|----------------|----------|
| name | One per People | String |

| | | |
|---------|----------------|--------|
| address | One per People | String |
| tel | One per People | String |

3.1.4 The Overview of Property Taxonomy

In this sub-section, we present the overview of property taxonomic tree (Fig. 13). Beside each node of the tree, a number is remarked indicating the position of the node. The preceding character “P” means “Property” to distinguish with the node in Object_Classification tree (described later), and “N” means a leaf node.

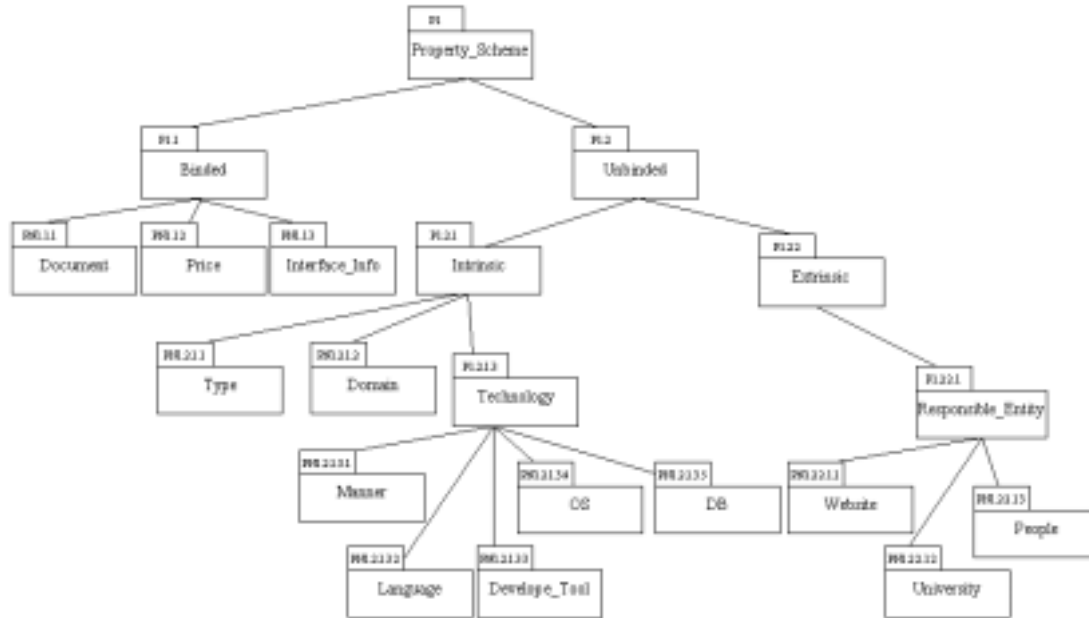


Fig. 13 The Taxonomic Tree of Property Taxonomy

3.2 Classifying BO with Properties

In our taxonomic tree of properties, these Object_Classification_Nodes can be a property only by associated to a business object through an Object_Classification. We may encounter the case that an instance of classification node may play different roles on one BO, for example, an organization is not only the developer of a BO, but also its submitter. We of course wouldn't like to create two instances which are the same individual. To deal with this problem, we classify the Object_Classification according the taxonomy of properties, then the role of a classification node is presented trough this association class. Consequently the definitions besides those inherited from Object_Classification are different. In the Object_Classification taxonomic tree, every node is a package except the leaf ones, which is a real Object_Classification class. Fig. 14 is the overview of the tree. Beside each node of the tree, a number is labeled indicating the position of the node. The preceding character “C” means “Classification” to distinguish with the node in property tree, and “N” means a leaf node.



Fig. 14 The Taxonomic Tree of Object_Classification

Tab. 2 is the description of some Object_Classification.

| Object_Classification Name | Note |
|----------------------------|---|
| Copyright | The document associated with a BO is copyright |
| Free | BO could be got free from the instances of Object_Classification_Node, such as Website. |
| Purchase | BO could be bought through the node it associated. |
| Written_with | The node is the language a BO is written with |
| Supported_DB | The node is the DB a BO support |
| Executing_Env | The node is the OS in which a BO can execute |

Tab.2 Notes of Some Object_Classification

3.3 An Example

Assume we have developed software component for a retailer, and then submitted it to a BO management organizer. This component is a set of EJBs and it has a commentary document. Fig. 15 shows how the description above is presented in metamodel framework. Each instance of classification and classification node is attached with the number that is the same as its class.

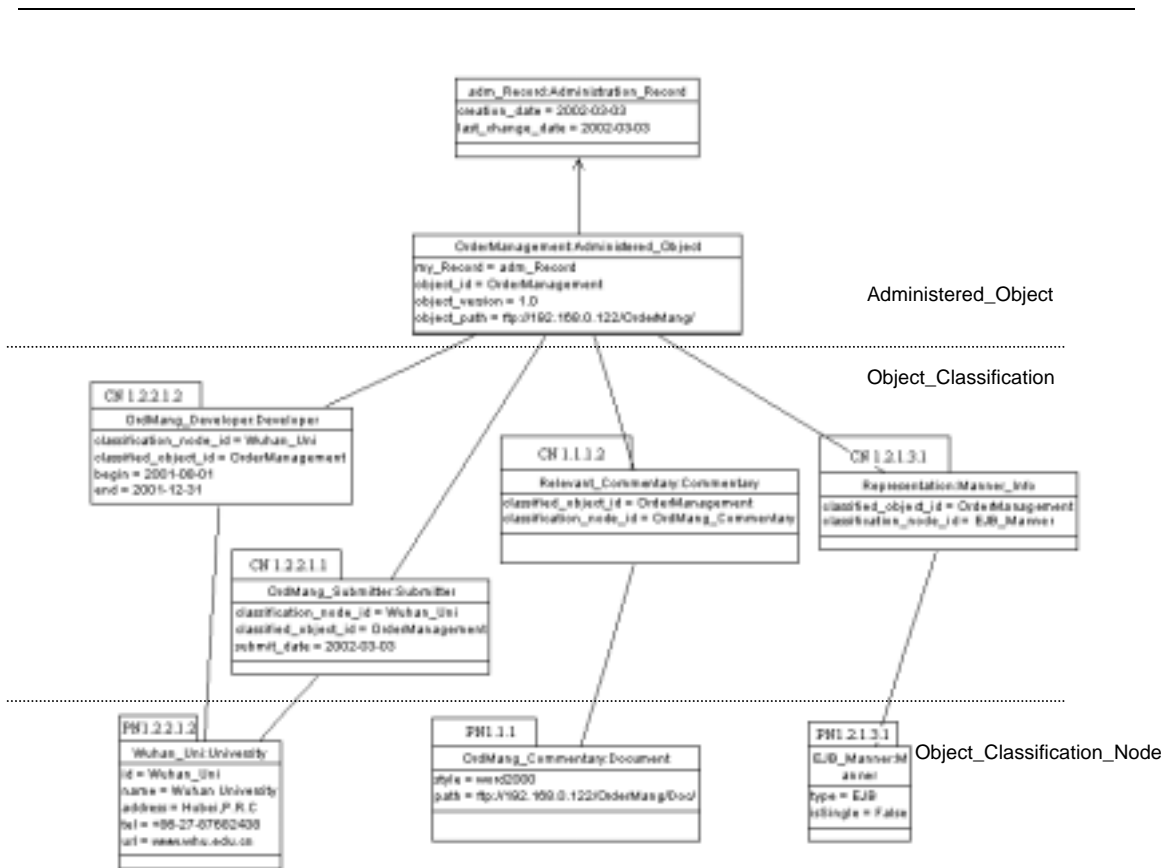


Fig. 15 Example showing Classification with Taxonomic Properties

4. Conclusion

Ontological analysis provides a way to check the strictness of inheriting and subsumption relation in metamodel for registering BO, which can bring order to the structure and eliminate confusion of vocabulary from taxonomy. At the same time, a Classification Node-Classification-BO structure offers a flexible way to classify various BO. In future, more detailed definition aiming to registering BO should be added in this metamodel, and the consistent between Object_Classification_Node and Object_Classification should be checked.

References

- [1] Guarino, N. 1997. Understanding, Building, and Using Ontologies: A Commentary to "Using Explicit Ontologies in KBS Development", by van Heijst, Schreiber, and Wielinga. International Journal of Human and Computer Studies, 46: 293-310.
- [2] Japan National body, ISO/IEC JTC1 SC32 WG2, September, 2001, Rationale for the NWI: Framework for Registering Business Object.
- [3] Guarino, N. 1998. Formal Ontology in Information Systems. In N. Guarino (ed.) Formal Ontology in Information Systems. Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998. IOS Press, Amsterdam: 3-15.
- [4] Guarino, N. 1997. Some Organizing Principles for a Unified Top-Level Ontology. LADSEB-CNR Internal Report 02/97, March 1997.
- [5] Guarino, N. and Welty, C. 2000. A Formal Ontology of Properties. In R. Dieng and O. Corby (eds.), Knowledge Engineering and Knowledge Management: Methods, Models and Tools. 12th International Conference, EKAW2000. Springer Verlag: 97-112.

-
- [6] OMG, Meta Object Facility (MOF) Specification, V1.3.
 - [7] ISO/IEC JTC 1/SC 32/WG 2, Editor's 2nd Draft of FDIS 11179-3
 - [8] OASIS/ebXML, Registry Information Model v2.0, Approved Committee Specification