

## Ontology Modeling and Object Modeling in Software Engineering

Dr. Waralak V. Siricharoen  
*University of the Thai Chamber of Commerce (UTCC)*  
126/1 Dindeang, Bangkok, Thailand 10400  
(66)26976506-7, (66)816966425  
[waralak\\_von@utcc.ac.th](mailto:waralak_von@utcc.ac.th)

### **Abstract**

*A data model is a plan for building a database and is comparable to an architect's building plans. There are two major methodologies used to create a data model: the Entity-Relationship (ER) approach and the Object Model. This paper will be discussed only the object model approach. The goal of the data model is to certify that all data objects required by the database are completely and accurately represented. Ontologies are objects of interest (Universal of discourse). The objective of this paper is to simplify object models compare with ontologies models. There are some similarities between objects in object models and concepts sometimes called classes in ontologies. Ontology can help building object model. The object model is the center of data modeling; on the other hand ontology itself has the concept which is the basis of knowledge base. Because ontologies are closely related to modern object-oriented software design, it is good attempt to adapt existing object-oriented software development methodologies for the task of ontology development. Selected approaches originate from research in artificial intelligence; knowledge representation and object modeling are presented in this paper. Some issues mentioned in this paper are related with their connection; some are addressed directly into the similarities or differences point of view of both. This paper also presents the available tools, methods, procedures, language, reusability which shows the corporation with object modeling and ontologies.*

### **1. Introduction**

The object oriented paradigm is the structure in software engineering, influencing all attempts in information science. Data modeling is probably the most time consuming and labor intensive part of the development process and it need a lot of developer's experience. A common response by practitioners who write on the subject is that you should no more build a database without a model than you should build a house without blueprints<sup>1</sup>.

A data model is a conceptual representation of the data structures that are required by a database. The data structures include the data objects, the associations between data objects, and the rules which run operations on the objects. It serves as a bridge between the concepts that construct real-world events and processes and the physical representation of those concepts in a database. The database design process generally follows five steps: planning and analysis, conceptual design, logical design, physical design, and implementation. The data model gets its inputs from the planning and analysis stage. Here the software developer, along with analysts, collects information about the requirements of the database by reviewing existing documentation and interviewing end-users. The one output of data model is an object model (class diagram) or we called object model which represents the data structures in a pictorial

---

<sup>1</sup> <http://www.utexas.edu/its/windows/database/datamodeling/dm/design.html>

form<sup>1</sup>. Because the picture diagram is simply knowledgeable, it is valuable tool to communicate the model to the end-user.

The objects are then modeled and analyzed using an object diagram. The diagram can be reviewed by the software developer and the end-users to conclude its completeness and accuracy. What makes an object an object or attribute? For example, given the statement *Professor teaches in many courses which are assigned by the University*. Should *Professor* be classified as an object or attribute? The answer depends upon the requirements of the database. In some cases, *Professor* and *course* would be an object, in some it would be an attribute.

Ontology is the concept which is separately identified by domain users, and used in a self-contained way to communicate information. Combination of concept is the knowledge base or knowledge network. Ontologies themselves are rising as an important tool for coping with very great, compound and various sources of information. It has also been known that ontologies are advantageous for data modeling in software engineering [17]. Ontology is well known as description of declaration and abstract way the domain information of the application, it involved with vocabulary and how to constrain the use of the data [3] and they are used widely in the semantic web approach, which requires a significant degree of structure. In the area of ontology the concept have been supplemented above which allow to express the similarity of concept in ontology with object (or atom) in object oriented [4]. Ontology is actually well known in philosophy research area since 1960s, in the artificial intelligence arena, has been focused on knowledge modeling. The term ontology is used to refer to an explicit specification of a conceptualization [of a domain] is mentioned by Tom Gruber which we are already familiar with for quite sometimes. In other words, ontology refers to a formalization of the knowledge in the domain. Ontologies are used for various purposes: first is the documents in the document base are annotated and classified according to the ontology [20].

The main contribution of this paper is that it is seem appropriate to try to emphasize similarities as well as significant differences between data modeling and ontology modeling by describing the basic components of them. Ontology are intended to give details and explain the world, while object model(database) are meant to describe that part of the world whose representation has to be managed for some application purpose. Overcoming differences is a meaningful way to benefit one domain with results from the other domain. The main thought is that object-oriented software development methodologies show promise as a basis for ontologies methodologies. On the other hand, the well- produced ontologies themselves which are published online such as in ontology libraries. They are the very good starting point to use ontologies to facilitate discovering object modeling as well.

## 2. Ontology Modeling

Ontology describes basic concepts in a domain and defines relations among them. Basic building blocks of ontology design include: classes or concepts, properties of each concept describing various features and attributes of the concept, restrictions on slots (facets). This section will clarify each part of ontology modeling as following.

### 2.1. Concept/Class

Concepts that are objects are likely to be best represented by classes. Classes (Concepts) are abstract groups, sets, or collections of objects. Concepts in the ontology should be close to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain [9].

A class is defined for a conceptual grouping of similar terms. For example, a *Person* could be represented as a class which would have many subclasses such as *Professor*. Each class is described by a definition which specifies the slots and values that describe the class itself.

They may contain individuals, other classes, or a combination of both. Ontologies vary on whether classes can contain other classes, whether a class can belong to itself, whether there is a universal class (that is, a class containing everything), etc. The concepts arranged in an inheritance hierarchy.

## 2.2. Slots/Facets

Own slots are slots used to describe properties (or “is part of”) of the term itself. For most terms, the only slots they have are own slots. The example of slots on the class *Mother*, subclass-of is an own slot because it is used to indicate a property of the class *Mother*. Whereas, *has-children* would be an instance slot because it describes a property of instances of the class *Mother*<sup>9</sup>.

A slot is used to describe a relationship between two terms. The first term must be an instance of the class that is the domain of the slot and the second must be an instance of the class that is the Range of the slot. For example, *brother* could be represented as a slot such that its Domain was *Animal* and its Range was *Male-Animal*. A slot may also be referred to as a binary relation<sup>2</sup>. They are attached to classes or slots and contain meta-information, such as comments, constraints and default values. Slots represent the attributes of the classes. Possible slot types are primitive types (Integer, Boolean, String etc.), references to other objects (modeling relationships) and sets of values of these types [7].

For each *Professor*, we want to know her *address and salary*, and what *course* she works for. As we continue to generate terms, we are completely defining the scope of ontology, by what we finally decide to include and what to exclude. For example, in order to verify the status of *Professor*, a *Professor* has to work for at least one *Course*. Each *Professor* which inherits the slot from concept *Person* has an *address* slot of type *String*. Properties of the classes, such as *age* or *address* can be represented by slots, and restrictions on properties or relationships between classes and or slots, are represented by slot facets.

## 2.3. Axioms

An axiom is a sentence in first order logic (F-logic) that is assumed to be true without proof. In practice, axioms can be used to refer to the sentences that cannot be represented using only slots and values on a frame<sup>2</sup>. Axioms must be entered in prefix notation. Use => to indicate logical implication, <=> to indicate logical equivalence, **and** to indicate conjunction, **or** to indicate disjunction, **not** to indicate negation, and **exists** to indicate existential quantification<sup>2</sup>. For example in OWL, it is possible to state that two or more classes are equivalent<sup>3</sup>,

*axiom ::= 'EquivalentClasses(' classID classID { classID } )'*

## 2.4. Instances

Instances represent specific entities from the domain knowledge base (KB). An example KB based on the *Person* ontology might contain the specific *Professor Jane* and *Course SP233*

---

<sup>2</sup> <http://www-ksl-svc.stanford.edu:5915/doc/frame-editor/glossary-of-terms.html>

<sup>3</sup> <http://www.w3.org/TR/owl-semantics/syntax.html#2.3.1>

*Database.* Individuals (instances) are the basic components of ontology. The individuals in ontology may include concrete objects such as people, animals, tables, automobiles, molecules, and planets, as well as abstract individuals such as numbers and words.

## 2.5. Relationships

A relation is used to describe a relationship among two or more terms. If a relation represents a relationship between only two terms, it is called a slot or a binary relation. If the relation describes a relationship among  $n$  terms such that there is a unique  $n^{\text{th}}$  term corresponding to any set of the first  $n-1$  terms, then the relation is called a function (section 2.6). For examples of binary relations include: subclass-of and connected-to. If we introduce relationships to ontology, we find that this simple and well-designed hierarchy structure becomes complex and significantly more difficult to interpret manually. It is not difficult to understand why an entity that is described as 'part of' another entity might also be 'part of' a third entity. Consequently, entities may have more than one parent. The structure that emerges is known as a directed acyclic graph (DAG<sup>4</sup>). As well as the standard is-a and part-of relationships, ontologies often include additional types of relation that further refine the semantics they model. These relations are often domain-specific and are used to answer particular types of question. The is-a relationship can be used to inherit attributes and semantic relationships down (against the direction of the arrows) from higher nodes to lower nodes in the DAG. This is very similar to inheritance in Object Oriented Databases such as C++<sup>5</sup>. Example: If *Person* has the attribute *Name* then *Professor* would inherit *Name*. We don't have to specify that *Professor* has *Name*.

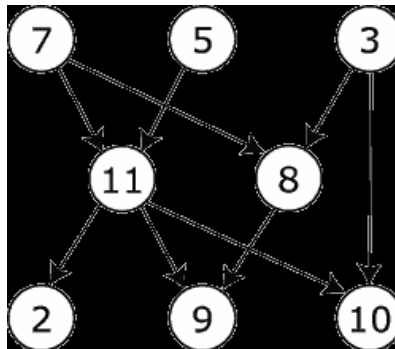


Figure 1. A simple directed acyclic graph<sup>4</sup>

For example in the domain of *Professors*, we might define a live-in relationship which tells us who each Professor lives in. So *Jane is live in Bangkok*. The ontology may also know that *Bangkok is-in Thailand* and *Thailand is-a country in Asia*. Software using this ontology could now answer a question like *Who (Professor) is live in Thailand?*

## 2.6. Functions

A function is a special type of relation which relates some number of terms to exactly one other term. That is, a function is a relation such that no two relationships of  $n$  terms in the relation have the same first  $n-1$  terms. For example, *Mother* is a function that relates an *Animal*

<sup>4</sup> [http://en.wikipedia.org/wiki/Directed\\_acyclic\\_graph](http://en.wikipedia.org/wiki/Directed_acyclic_graph)

<sup>5</sup> [http://web.njit.edu/~geller/what\\_is\\_an\\_ontology.html](http://web.njit.edu/~geller/what_is_an_ontology.html)

to exactly *one female animal*. A function may also be referred to as a slot if it relates only two terms [19].

## 2.7. Standards and Languages

One of the more recent developments with the Web is an activity known as the Semantic Web. The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation [17]. For the web, ontology is about the precise description of Web information and relationships between Web information. So the one important aspect of the Semantic Web is a set of ontologies. Three enabling technologies for the Semantic Web are XML, RDF and ontologies. Each of these has an important role to play in deploying and reusing learning objects on the Semantic Web. XML is used to markup the structure of a learning object in a machine readable way. It is also used to describe the metadata associated with objects. RDF allows the specification of metadata and other information associated with objects in a more flexible manner, facilitating the discovery and exchange of objects with limited information or more than one metadata specifications. Ontologies allow the specification of concepts in a domain as well as the terms used to markup content in a learning object. Shared ontologies allow for different systems to come to a common understanding of the semantics of an object.

The ontology also was developed using OWL (Web Ontology Language). OWL was proposed by the W3C for publishing and sharing data, and automating data understanding by computers using ontologies on the Web. OWL is being planned and designed to provide a language that can be used for applications that need to understand the meaning of information instead of just parsing data for display purposes [17]. OWL became a W3C (World Wide Web Consortium) recommendation in February 2004. W3C Recommendation is understood by the industry and the web community as a web standard. A W3C Recommendation is a stable specification developed by a W3C Working Group and reviewed by the W3C Membership<sup>6</sup>. OWL is an ontology language for the Web, which builds on a rich technical tradition (of both formal research and practical implementation), including SHOE (Simple HTML Ontology Extensions), OIL, and DAML+OIL. The technical basis for much of OWL is the part of the formal knowledge representations field known as Description Logic (aka "DL"). DL is the main formal underpinning of such diverse kinds of knowledge representation formalisms as semantic nets, frame-based systems, and others<sup>7</sup>.

---

<sup>6</sup> [http://www.w3schools.com/rdf/rdf\\_owl.asp](http://www.w3schools.com/rdf/rdf_owl.asp)

<sup>7</sup> <http://www.xml.com/pub/a/2003/08/20/deviant.html>

```
<owl:Class rdf:ID="Professor">
  <rdfs:label>Professor</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Faculty" />
</owl:Class>
<owl:Class rdf:ID="Faculty">
  <rdfs:label>faculty member</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Employee" />
</owl:Class>
<owl:Class rdf:ID="Employee">
  <rdfs:label>Employee</rdfs:label>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Person" />
    <owl:Restriction>
      <owl:onProperty rdf:resource="#worksFor" />
      <owl:someValuesFrom>
        <owl:Class rdf:about="#Organization" />
      </owl:someValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
... <owl:ObjectProperty rdf:ID="teacherOf">
  <rdfs:label>teaches</rdfs:label>
  <rdfs:domain rdf:resource="#Faculty" />
  <rdfs:range rdf:resource="#Course" />
</owl:ObjectProperty>
```

Figure 2. The example of OWL description about class Professor

From Figure 2 shows that “rdfs:subClassOf” is represent the inherit characteristics of ontology. This means *Professor* is subclass of *Faculty*, so *Professor* also inherits the ObjectProperty from *Person*. The development of ontologies for the Web has led to the apparition of services providing lists or directories of ontologies with search facility. Such directories have been called ontology libraries [16]. The following are static libraries of human-selected ontologies.

- The DAML Ontology Library<sup>8</sup> maintains a legacy of ontologies in DAML (Figure 3.).
- Protégé Ontology Library<sup>9</sup> contains a set of owl, Frame-based and other format ontologies. Protégé is a free, open source ontology editor and knowledgebase framework. The Protégé platform supports two main ways of modeling ontologies via the Protégé-Frames and Protégé-OWL editors. Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and XML Schema. Protégé is based on Java, is extensible, and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development (Figure 4).
- SchemaWeb<sup>10</sup> is a directory of RDF schemas expressed in RDFS, OWL and DAML+OIL (Figure 5).

---

<sup>8</sup> [www.daml.org/ontologies/](http://www.daml.org/ontologies/)

<sup>9</sup> <http://protege.stanford.edu/overview/protege-owl.html>

<sup>10</sup> <http://www.schemaweb.info/default.aspx>

## DAML Ontology Library

Summaries

- [Ontologies by URI](#)
- [Ontologies by Submission Date](#)
- [Ontologies by Keyword](#)
- [Ontologies by Open Directory Category](#)
- [Ontologies by Class](#)
- [Ontologies by Property](#)
- [Ontologies by Namespace Used](#)
- [Ontologies by Funding Source](#)
- [Ontologies by Submitting Organization](#)


Queries

- [Classes by Name](#)
- [Properties by Name](#)

## Ontologies by Keyword

Keyword	URI
academia	<a href="http://www.aktors.org/ontology/portal">http://www.aktors.org/ontology/portal</a>
academic department	<a href="http://www.cs.umd.edu/projects/plus/DAML/onts/cs1.0.daml">http://www.cs.umd.edu/projects/plus/DAML/onts/cs1.0.daml</a>
academic department	<a href="http://www.cs.umd.edu/projects/plus/DAML/onts/cs1.1.daml">http://www.cs.umd.edu/projects/plus/DAML/onts/cs1.1.daml</a>
academic departments	<a href="http://www.aktors.org/ontology/portal">http://www.aktors.org/ontology/portal</a>
Academic Positions	<a href="http://www.daml.ricmu.edu/ont/homework/cmu-ri-employmenttypes-ont.daml">http://www.daml.ricmu.edu/ont/homework/cmu-ri-employmenttypes-ont.daml</a>
access control primitives	<a href="http://www.w3.org/2000/10/swap/pim/doc.rdf">http://www.w3.org/2000/10/swap/pim/doc.rdf</a>

Figure 3. DAML Ontology library



[HOME](#) | [OVERVIEW](#) | [DOCUMENTATION](#) | [DOWNLOADS](#) | [SUPPORT](#) | [COMMUNITY](#) | [WIKI](#) | [ABOUT US](#)

### protégé ontologies

In addition to downloading Protégé, we encourage you to browse the library of ontologies on our Wiki. The ontologies listed on our Wiki were developed either here at Stanford or by our user community. Follow individual instructions to download the ontologies that are of interest to you.

*see also:*

- [Browse OWL ontologies](#)
- [Browse Frame-based ontologies](#)
- [Browse ontologies in other formats \(e.g., DAML+OIL, RDF Schema, etc.\)](#)

If you have developed an ontology using Protégé, we encourage you to add a link to our Wiki.

#### OWL ontologies

Information on how to open OWL files from Loading Projects section of the Getting Started page: <http://www.google.com/search?q=...>

- [AIM@SHAPE Ontologies](#): Ontologies Tools for the development of Semantic Web digital objects.
- [amino-acid.owl](#): A small OWL ontology.
- [Basic Formal Ontology \(BFO\)](#)
- [bhakti.owl](#): An OWL ontology for the development of Vedic consciousness engineering.
- [Biochemical Ontologies](#): Over 30 ontologies normalized into non-disjoint primitive skeletons, macromolecules and the proteome.
- [BioPAX](#): An OWL ontology for biological processes.

#### Frame-based ontologies

In the context of this page, the phrase "frame-based ontology" refers to an ontology that can be opened in the Protégé editor. For more information on how to open an ontology in Protégé, see the Protégé documentation.

- [Biological Processes](#): A knowledge model of biological processes, machine-interpretable, to allow reasoning.
- [CEDEX](#): Representation of CEDEX in Protégé. CEDEX is a knowledge model of biological processes.
- [Dublin Core](#): Representation of Dublin Core metadata in Protégé.
- [Engineering ontologies](#): A set of ontologies for representing engineering concepts.
- [GandriKB \(Gene annotation data representation\)](#): An ontology for representing gene annotation data.
- [GeneOntologyInProtégé](#): Knowledge acquisition, consensus, and representation of Gene Ontology.
- [Guideline Interchange Format \(GLIF\)](#): Representation of clinical guidelines.

#### Other ontology formats

- [Dublin Core](#): Representation of Dublin Core metadata in Protégé.
- [HL7-RIM](#): HL7-RIM as a Protégé ontology. Contributed by the HL7 community.
- [IPTC Subject Reference System](#): RDF Schema and Protégé ontology.
- [Learner](#): An ontology describing Learner features used in the ELena system.
- [PetriNet Semantic Web Infrastructure](#): Contributed by the PetriNet community.
- [Resource-Event-Agent Enterprise \(REA\)](#): An ontology for representing enterprise information systems.
- [Universal Standard Products and Services Classification](#)

Figure 4. Protégé Ontology Library

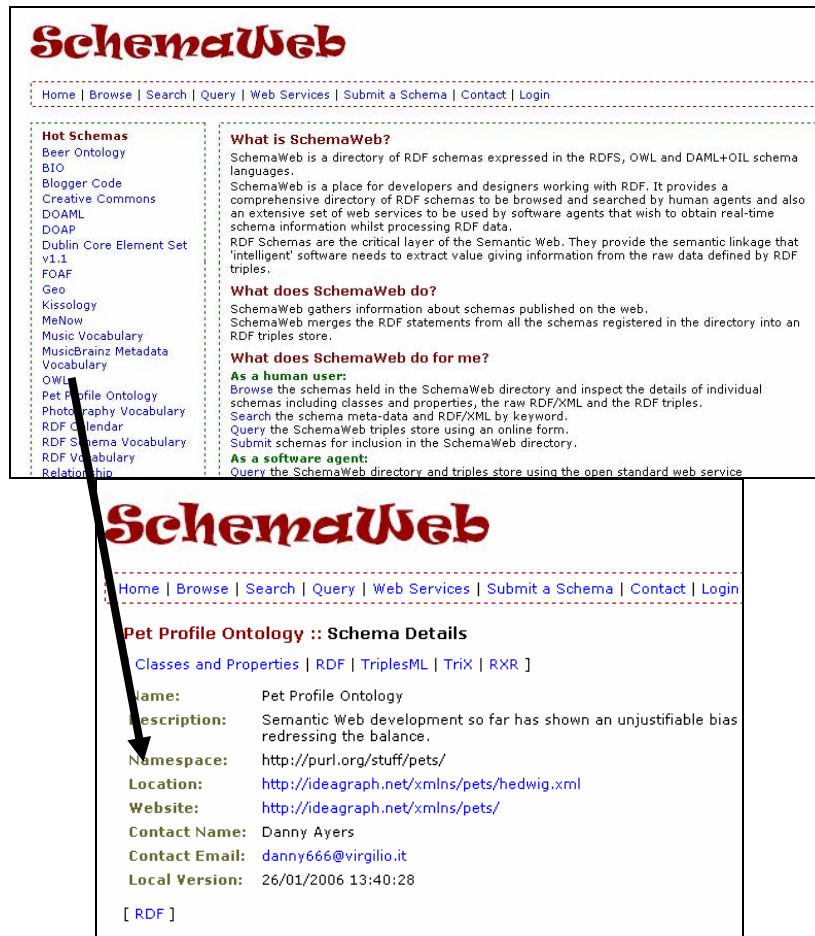


Figure 5. SchemaWeb

Finding ontologies is seen as important to avoid the creation of new ontologies where serviceable ones already exist [20]. This approach will guide to emergence of widely-used canonical ontologies. The following are both directories and search engines. They include crawlers searching the Web for well-formed ontologies.

- Swoogle<sup>11</sup> is a directory and search engine for all RDF resources available on the Web, including ontologies. Swoogle support querying for ontologies containing specified terms, this can be refined to find ontologies where such terms occur as classes or properties, or in some senses about specified terms[20].
- The OntoSelect Ontology Library<sup>12</sup> provides an access point for ontologies on any possible topic or domain that will be updated continuously, organized in a meaningful way and with automatic support for ontology selection in knowledge markup. Unlike the DAML and SchemaWeb ontology libraries, OntoSelect is not based primarily on a static registration of published ontologies, but includes a

<sup>11</sup> <http://swoogle.umbc.edu/>

<sup>12</sup> <http://olp.dfki.de/ontoselect/>



crawling procedure that monitors the web for any newly published ontologies in the following representation formats: RDF/S, DAML or OWL [21].

- Ontaria<sup>13</sup> is a searchable and browsable directory of semantic web data, with a focus on RDF vocabularies with OWL ontologies. Ontaria is primarily intended for people creating RDF content who want to better understand which vocabularies are available and how they are being used. Ontaria may be useful for finding and exploring arbitrary RDF content.



Figure 6a. Swoogle

Figure 6b. OntoSelect



Figure 6c. Ontaria

### 3. Object Modeling

In class diagram, classes are represented by a box with 3 parts: the name of class, the attributes of the class (specified by their name, type and visibility) and the operation of the classes as see in Figure 7.

#### 3.1. Object/Class

To emphasize that an object actually contains meaningful data, a term data object is sometimes used to refer to such an object. Object or Class is the tangible things. For example, the *University* system might contain the object *Professor* and instance *Jane*.

---

<sup>13</sup> <http://www.w3.org/2004/ontaria/>

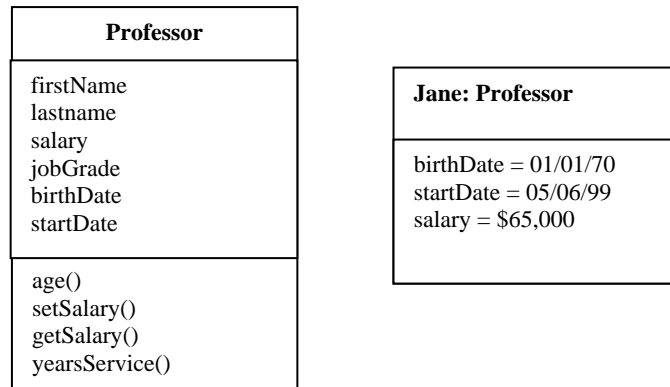


Figure 7. The example of class *Professor* and object *Professor: Jane*

### 3.2. Attributes/Properties

Attributes are data objects that either identify or describe entities. Attributes that identify entities are called key attributes. Attributes that describe an entity are called non-key attributes. Objects will have at least one attribute. Possible slot types are primitive types (integer, Boolean, string etc.), references to other objects (modeling relationships) and sets of values of these types. Properties generally associated with a class are its attributes and operations, which, following OMG terminology, we will collectively refer to as features. Thus, the class defines the features of *Professor* objects, each of which has a *birthDate* attribute, a *startDate* attribute, a *salary* attribute.

### 3.3. Method/Operations/Functions

They are attached to classes or slots and contain meta-information, such as comments, constraints and default values. For example, in order to verify the status of a *Professor*, a *Professor* has to assign at least one *Course*. They are attached to classes or slots and contain meta-information, such as comments, constraints and default values. As in Figure 7 the operations are *age ()* and *yearsService ()*.

### 3.4. Relationship/Relations

In object-oriented programming, is-a relationship exist through the definitions of inheritance, and other relationships exist via class membership. To construct a generalization hierarchy, all common attributes are assigned to the supertype. The supertype is also assigned an attribute, called a discriminator, whose values identify the categories of the subtypes. Attributes unique to a category, are assigned to the appropriate subtype. Each subtype also inherits. The inheritance mechanism forms one of the foundations of object technology, and is the primary feature that, at the language level, distinguishes object-oriented development from object-based development. From a modeling perspective, inheritance is used to capture the situation in which one type of abstraction is "like" another type of abstraction, but with some additional properties. Thus, instead of defining all features of *Professor* objects directly in the class *Person*, it is also possible to use inheritance to define some of them in a superclass of *Professor*. It means in describing Figure 8, "*An Professor is a person.*"[18]

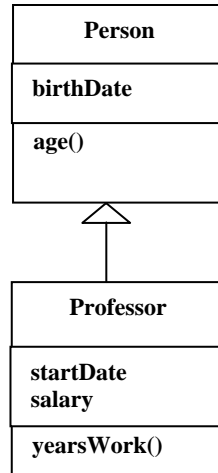


Figure 8. Hierarchy in object model

### 3.5. Instances

Each instance of the Employee class has values for each of these attributes, as illustrated as in Figure 7 and 8: "*Jane is a professor.*" This UML model captures the fact that the object *Jane* is an *instance-of* the class *Professor*, and has specific values for its attributes. Note: In this and the following examples, our focus is not on providing the optimal representation of every logical concept, for example whether *age* should be an attribute or a method, or whether role modeling should be used in place of inheritance but rather on illustrating the different effects of the meta-modeling and inheritance mechanisms on model elements[18].

### 3.6. Standards and Languages

Data (Conceptual) modeling languages have mainly been used in the development process of databases schema and have been supported by numerous case tools. Unified Modeling Language (UML) is well known and widely used object modeling that consist of concepts/entities/classes in a specification hierarchy, the description of concepts by attributes which have range and relationship between concepts. UML defines several types of diagram that can be used to model the static and dynamic behaviors of a system.

The business information model can be defined using an ontology. Concepts (or classes) are defined and related to each other, much like classes in a UML class diagram [10]. Unlike in commonly-known object-oriented data models attributes and associations are not defined with the class specification itself. Instead, class properties are first-class primitive themselves [11].

Conceptual (or Data) modeling deals with the question on how to describe in a declarative and abstract way the domain information of an application, its relevant vocabulary, and how to constrain the use of the data. Modeling languages like UML and ODMG have been developed for object oriented models in software engineering. Common to all of these newer models is the arrangement of concepts/entities/classes in a specialization hierarchy, the description of concepts by attributes which have ranges and relationships between concepts. Concepts, relationship types and attributes abstract from concrete objects or values and thus describe the schema (the ontology). On the other hand concrete objects populate the concepts, concrete values instantiate the attributes of these objects and concrete relations instantiate relationships [3].

Like UML, Object Data Management Group (ODMG) Object Model is intended to allow portability of applications among object database products. It provides a common model for

these products by defining extensions to the OMG object model that support object database requirements. UML and ODMG have been developed for object oriented models in software engineering. There are a lot of paper address about the object oriented standard to be used for ontology modeling for example as [5] the large user community and commercial support for object oriented standards warrants the investigation of standard object modeling technique for ontology development. We can see clearly that ontology modeling can be connecting to object model by this example: the ontology representation language used in this paper a UML class diagram (contain OCL constraints) in conjunction with an object diagram – contains both a highly structured model that could support automated reasoning (the basic class and object model, ignoring the constraints)

One approach for implementing objects is to have a class, which defines the implementation for multiple objects. A class defines what types the objects will implement, how to perform the behavior required for the interface and how to remember state information. Each object will then only need to remember its individual state. Although using classes is by far the most common object approach, it is not the only approach (using prototypes is another approach) and is really peripheral to the core concepts of object-oriented modeling [1]. With F-Logics we provide a clearly defined syntax and semantics to ontologies and the representation of these knowledge models is based on a well-understood logical framework. F-Logic allows to describe ontologies, i.e. classes, the hierarchy of classes, their attributes and relationships between classes in an object oriented style way. For simplicity, Assume that some of the objects you plan to use in your application will be developed by many different vendors using languages such as C, C++, and Java [12] as see the example of the class *Professor* applied in Java<sup>14</sup>.

```
public class Professor {  
  
    public String FirstName;  
    public String LastName;  
    private float Salary;  
    private int JobGrade;  
  
    public Professor() {  
        FirstName = "";  
        LastName = "";  
        Salary = 0.0f;  
        JobGrade = 0;    }  
    public Professor (String First, String Last) {  
        FirstName = First;  
        LastName = Last;  
        Salary = 0.0f;  
        JobGrade = 0;}  
    public Professor (String First, String Last, float salary, int grade) {  
        FirstName = First;  
        LastName = Last;  
        Salary = salary;  
        JobGrade = grade;}  
    public void SetSalary(float Dollars) {  
        Salary = Dollars;}  
}
```

---

<sup>14</sup> <http://livedocs.adobe.com/coldfusion/6.1/htmldocs/java30.htm>

```
public float GetSalary() {
    return Salary;}

```

#### 4. Modeling Comparison Issue

This following section shows the comparison between Semantic Web languages and object-oriented languages. In general the domain models of both ontology language and object-oriented language consists of classes, properties and instances (individuals). Classes can be arranged in a subclass hierarchy with inheritance. Properties can take objects or primitive values (literals) as values.

**Table 1. A Comparison of Ontology language and Object-Oriented Languages (UML)**  
 [15]

	<b>Object-Oriented Languages</b>	<b>Ontologies Language (specific XML, OWL, and RDF)</b>
<b>Classes and Instances</b>	Classes are regarded as types for instances.	Classes are regarded as sets of individuals.
	Each instance has one class as its type. Classes cannot share instances.	Each individual can belong to multiple classes.
<b>Properties, Attributes and Values</b>	Properties are defined locally to a class (and its subclasses through inheritance).	Properties are stand-alone entities that can exist without specific classes.
	Instances can have values only for the attached properties. Values must be correctly typed. Range constraints are used for type checking.	Instances can have arbitrary values for any property. Range and domain constraints can be used for type checking and type inference.
	Classes encode much of their meaning and behavior through imperative functions and methods.	Classes make their meaning explicit in terms of OWL statements. No imperative code can be attached.
	Classes can encapsulate their members to private access.	All parts of an OWL/RDF file are public and can be linked to from anywhere else.
	Closed world: If there is not enough information to prove a statement true, then it is assumed to be false.	Open world: If there is not enough information to prove a statement true, then it may be true or false.
<b>Role in the Design Process</b>	Some generic APIs are shared among applications. Few (if any) UML diagrams are shared.	RDF and OWL have been designed from the ground up for the Web. Domain models can be shared online.
	Domain models are designed as part	Domain models are designed to

	of software architecture.	represent knowledge about a domain, and for information integration.
	UML, Java, C# etc. are mature technologies supported by many commercial and open-source tools.	The Semantic Web is an emerging technology with some open-source tools and a handful of commercial vendors.
	UML models can be serialized in XMI, which is geared for exchange among tools but not really Web-based. Java objects can be serialized into various XML-based or native intermediate formats.	RDF and OWL objects have a standard serialization based on XML

## 5. Reusability

Once an object has been created and installed on a file system, implementing inheritance from that object using a new object is not possible. However, within the object, using a language such as C++, interfaces can be inherited from or overridden. Using any object-oriented language that supports inheritance, this kind of functionality is possible when building the object. To increase sharing and minimize duplicate efforts, the ontology libraries have been established [2]. Now there are many ontologies are produced and contained in the ontologies library for simply inquiry. As we you can see the example of semantic search engine in previous section that we can simple type your keyword of ontologies to those search engine, we will find the proper ontologies that be needed in very short period of time [17].

## 6. Conclusion

Ontology describes the types of things that exist, and rules that govern them; a data model defines records about things, and is the basis for a database design. Not all the information in ontology may be needed (or can even be held) in a data model and there are a number of choices that need to be made. For example, some of the ontology may be held as reference data instead of as entity types [6].

Ontologies are promised to bright future. In this paper we propose that as ontologies are closely related to modern object-oriented software engineering, it is natural to adapt existing object-oriented software development methodologies for the task of ontology development. This is some part of similarity between descriptive ontologies and database schemas, conceptual data models in object oriented are good applicant for ontology modeling, however; the difference between constructs in object models and in current ontology proposals which are object structure, object identity, generalization hierarchy, defined constructs, views, and derivations. We can view ontology design as an extension of logical database design, which mean that the training object data software developers could be a promising approach. Ontology is the comparable of database schema but ontology represent a much richer information model than normal database schema, and also a richer information model compared to UML class/object model. Object modeling focus on identity and behavior is completely different from the relational model's focus on information.

It is likely to adjust existing object oriented software development methodologies for the ontology development. The object model of a system consists of objects, identified from the text description and structural linkages between them corresponding to existing or established relationships. The ontologies provide metadata schemas, offering a controlled vocabulary of concepts. At the center of both object models and ontologies are objects within a given problem

domain. The difference is that while the object model should contain explicitly shown structural dependencies between objects in a system, including their properties, relationships, events and processes, the ontologies are based on related terms only. On the other hand, the object model refers to the collections of concepts used to describe the generic characteristics of objects in object oriented languages. Because ontology is accepted as a formal, explicit specification of a shared conceptualization, Ontologies can be naturally linked with object models, which represent a system-oriented map of related objects easily.

An ontology structure holds definitions of concepts, binary relationship between concepts and attributes. Relationships may be symmetric, transitive and have an inverse. Minimum and maximum cardinality constraints for relations and attributes may be specified. Concepts and relationships can be arranged in two distinct generalization hierarchies [14]. Concepts, relationship types and attribute abstract from concrete objects or value and thus describe the schema (the ontology) on the other hand concrete objects populate the concepts, concrete values instantiate the attributes of these objects and concrete relationship instantiate relationships. Three types of relationship that may be used between classes: generalization, association, aggregation.

## 6. Acknowledgements

The author would like to thank University of the Thai Chamber of Commerce (UTCC), Bangkok, Thailand for supporting the research funding.

## 7. References

- [1] ChiMu Corporation "Object Modeling Foundations of O-R Mapping", to appear, [Online] available: <http://www.chimu.com/publications/objectRelational/part0003.html>
- [2] W. V. Siricharoen, "Ontologies and Object models in Object Oriented Software Engineering", In Proceeding Conference: The International MultiConference of Engineers and Computer Scientists 2006 (IMECS 2006), Hong Kong, March 2006, pp 856 – 861.
- [3] Z. Hu, "Using Ontology to Bind Web Services to the Data Model of Automation Systems", In Journal Lecture Notes in Computer Science 0302-9743 (Print) 1611-3349 (Online) Springer Berlin / Heidelberg Volume 2593/2003, pp 154-168.
- [4] W. V. Siricharoen, "Ontologies and Software Engineering", In Proceeding International Conference on Computational Science (ICCS 2007) Lecture Notes of Computer Science, May 27-30, 2006, Beijing, China, Volume 44881105, pp. 1151-1162.
- [5] S. Cranefield, M. Purvis, "UML as an Ontology Modeling Language", In Proceeding of the IJCAI-99 Workshop on Intelligent Information Integration, Department of Information Science, University of Otago, New Zealand, 1999.
- [6] M. West, "Ontology Forum Database and Ontology mini-series Session-2", 2006, [Online] available: [From-Ontology-to-DataModel-- MatthewWest\\_20061116.ppt](#)
- [7] H. Knublauch, "Three Patterns for the Implementation of Ontologies in Java", Submitted to OOPSLA'99 Metadata and Active Object-Model Pattern Mining Workshop, Research Institute for Applied Knowledge Processing (FAW), Germany, 1999, [Online] available: <http://www.knublauch.com/publications/OOPSLA99-Metadata.html>

- [8] S. Strom, "Building a Large-Scale Generic Object Model: Applying the CYC Upper Ontology to Object Database Development in Java", to appear, [Online] available: [www.techtrader.com](http://www.techtrader.com).
- [9] N. F. Noy, and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880", March 2001. [Online] available: <http://mia.ece.uic.edu/~papers/MediaBot/ontology101.pdf>
- [10] D. E. Jenz, "It is High Time for Pursuing the Ontology-Centric Approach", 2003, [Online] available: [http://www.bpiresearch.com/Resources/RE\\_SWPr/A\\_OntologyCentricApproach.pdf](http://www.bpiresearch.com/Resources/RE_SWPr/A_OntologyCentricApproach.pdf)
- [11] R. Volz, D. Oberle, and R. Studer, "Views for light-weight web ontologies", In Proceeding of SAC 2003, Melbourne, Florida, USA. 1999,
- [12] B. Morgan, "Java and the Component Object Model", to appear, [Online] available: <http://docs.rinet.ru/ZhPP/ch16.htm#TypeLibrariesandObjectDescriptionLanguage>
- [13] P. Mohan, and C. Brooks, "Learning Objects on the Semantic Web", to appear, [Online] available: [http://www.cs.usask.ca/~cab938/icalt2003\\_mohan\\_brooks.pdf](http://www.cs.usask.ca/~cab938/icalt2003_mohan_brooks.pdf)
- [14] N. Cullot, and al. et., "Ontologies : A contribution to the DL/DB debate", to appear, [Online] available: [http://lbdwww.epfl.ch/e/publications\\_new/articles.pdf/Cullot\\_SW\\_DB2003\\_CR.pdf](http://lbdwww.epfl.ch/e/publications_new/articles.pdf/Cullot_SW_DB2003_CR.pdf)
- [15] H. Knublauch, and et al., "A Semantic Web Primer for Object-Oriented Software Developers", W3C Working Group Note, March 9, 2006, [Online] available: <http://www.w3.org/TR/2006/NOTE-sw-oosd-primer-20060309/>
- [16] Y. Ding, and D. Fensel, "Ontology Library Systems: The key to successful Ontology Re-use", In Proceedings of the International Semantic Web Working Symposium 2001: SWWS2001. pp. 93-112, 2001, [Online] available: <http://www.semanticweb.org/SWWS/program/full/paper58.pdf>
- [17] W. V. Siricharoen, "Ontologies and Object models in Object Oriented Software Engineering" In International Association of Engineers (IAENG) Journal of Computer Science (IJCS) Volume 33(1), pp. 1151-1162, 2006, [Online] available: [http://www.iaeng.org/IJCS/issues\\_v33/issue\\_1/IJCS\\_33\\_1\\_4.pdf](http://www.iaeng.org/IJCS/issues_v33/issue_1/IJCS_33_1_4.pdf)
- [18] C. Atkinson, B. Henderson-Sellers and T. Kühne, "To Meta or Not to Meta—That Is the Question", 2001 [Online] available: <http://www.adtmag.com/joop/article.aspx?id=207>
- [19] A Glossary of Ontology Terminology, to appear. [Online] available: <http://www-ksl-svc.stanford.edu:5915/doc/frame-editor/glossary-of-terms.html>
- [20] Davies, J., Studer, R., and Warren, P., Semantic Web Technologies Trends and research in ontology-based systems, John Wiley & Sons Inc., West Sussex, England, 2006.
- [21] P. Buitelaar, T. Eigner, T. Declerck, "OntoSelect: A Dynamic Ontology Library with Support for Ontology Selection", In Proceeding of the 3rd International Semantic Web Conference (ISWC 2004), Hiroshima, Japan, November 2004, [Online] available: <http://www.dfki.de/~paulb/iswc04.demo.OntoSelect.pdf>



## Author

### Dr. Waralak V. Siricharoen



Waralak Vongdoiwang Siricharoen is lecturer and researcher of Computer Science Department, School of Science, University of the Thai Chamber of Commerce (UTCC), Bangkok, Thailand. She received her master degree in Business Administration with Computer Information Systems Certificate in 1999 from University of Southern New Hampshire (New Hampshire College), NH, USA. And she received her Doctoral Degree of Technical Science in 2005 from Asian Institute of Technology (AIT), Thailand. Her previous works focus on creating the object models from ontologies. Her works have been reported in peer-reviewed papers published in international journals, conference proceedings, and books. Her research area specialists are ontologies, database technology, software engineering, object oriented, human computer interaction, etc.

Website: <http://waralak.mypage.utcc.ac.th>

