

# Designing Metadata with Existing Application Ontologies

Heeryon Cho and Toru Ishida

Department of Social Informatics, Kyoto University

Kyoto 606-8501 Japan

cho@kuis.kyoto-u.ac.jp, ishida@i.kyoto-u.ac.jp

## Abstract

*Among various metadata design difficulties, 1) designing subsumption relation of two classes, 2) designing diverse properties of a given class, and 3) determining whether an item should be designed as class or property, are three common difficulties faced by human designers. We devised a metadata referring method, which provides example-based design information by organizing existing metadata defined in multiple ontologies in table formats. Our referring method provides 1) existing class hierarchy definitions, 2) similar and specific property definitions, and 3) overviews of class hierarchy and property definitions of a given class, after searching through numerous application ontologies. An evaluation experiment has revealed that our method is useful in design difficulties 1) and 2), but not in 3).*

## 1. Introduction

Dissemination of machine-understandable data through the Internet will facilitate the creation of an infrastructure for advanced information processing. Complex tasks, such as setting up schedules or searching for specific products that meet detailed criteria, will be automated by Internet agents once vast amount of machine-understandable data, or metadata, are shared by these agents [1]. To realize such metadata infrastructure, ontologies describing various application areas must be designed first.

Four kinds of ontologies with different levels of generality exist; they are top-level, domain, task, and application ontologies [6]. Some carefully constructed top-level [13], domain [10], and task [9] ontologies exist, and quite a few application ontologies are available on the Internet. Here, we focus on application ontologies that describe concepts depending both on a particular domain and task.

Although a number of ontology construction methodologies have been proposed to date [4] [11] [18], and applications using ontologies [12] can follow these methods to build ontologies, it is still difficult for metadata designer

who are at the application end to employ these methods when designing needed metadata.

Metadata designers usually browse and refer to existing ontologies, or select and refine the most relevant existing ontology to reuse it. Existing ontologies can be reused to create new ontologies by applying operations like inclusion, restriction, and polymorphic refinement [15], or they can be extended using a large-scale ontology [16]. Here, we focus on a method of referring to multiple ontologies so that metadata designers can find relevant metadata or obtain example-based design information.

In the next section, we lay out three specific metadata design difficulties, and elaborate on them using actual ontologies on the Internet. A metadata referring method, which addresses the design difficulties identified in section 2, is proposed in section 3, and evaluation of the proposed method is reported in section 4. Section 5 compares related works, and section 6 draws conclusion.

## 2. Metadata Design Difficulties

The terms ‘metadata’ and ‘ontology’ used in this paper are defined as follows: **Metadata** is data about data. In particular, it is machine understandable information shared over the Internet. Here, metadata specifically refers to classes and properties defined in ontologies. **Ontology** is an explicit specification of a conceptualization [5]. Or, it is a set of metadata composing a semantic structure. Ontologies can be described using ontology markup language such as OWL (Web Ontology Language).

We conducted a metadata design experiment on 41 human subjects, and identified the following common metadata design difficulties:

- Designing subsumption relation is difficult
- Designing diverse properties for a class is difficult
- Determining whether to design an item as class or property is difficult

We now explain each difficulty in detail using existing application ontologies on the Internet as examples. From here on after, underlined words starting with uppercase and

lowercase letters indicate classes and properties, respectively. The word Class may be omitted for classes. The same rule applies to the words in the figures.

## 2.1. Subsumption Relation Uncertainty

Subsumption relation, which is the basis of a taxonomy, is an extremely useful tool for imparting structure on an ontology [7]. Determining whether one class subsumes another is difficult, however, because IS-A relation between two generic concepts may have multiple meanings other than subclass/superclass relation [2].

Subsumption relations defined in actual ontologies can differ greatly as shown in Figure 1. The left and right graphs in Figure 1 represent different Person class hierarchies defined in two application ontologies. The graph on the left<sup>1</sup> shows Person having Biological\_Object and Actor as direct superclasses, whereas the graph on the right<sup>2</sup> shows four direct superclasses, GEO:SpatialThing, Agent, SWAP:Person and WN:Person. (GEO<sup>3</sup>, SWAP<sup>4</sup> and WN<sup>5</sup> denote outside ontologies and their abbreviated namespaces.) Although both ontologies define Person Class as the starting point of the superclass hierarchy, the overall class hierarchy design differs greatly between the two.

Now let us consider a design task where a superclass for Male and Female Class must be designed. Three kinds of superclasses, Person (Fig. 2 left), Gender (middle) or Animal (right), might be considered as possible candidates among many. One, two or all three superclass designs may be adequate, or none may be adequate. We describe this difficulty of designing appropriate class hierarchy as the problem of subsumption relation uncertainty.

## 2.2. Diverse Property Coverage

Part of the metadata design task includes a task of designing appropriate properties for a given class. Note that we define ‘properties of a specific class’ as ‘properties having that specific class as domain.’ Consider a task where properties of Person Class must be designed. Different design possibilities may exist as shown in Figure 3 (left graph<sup>6</sup> vs. right<sup>7</sup>). The left graph in Figure 3 shows Person having email, name\_Person, phone, address\_Person and homepage as its properties, while the right graph shows emailAddress, givenName, familyName and phoneNumber as Person properties. Of course, these are only two design possibilities. Other property designs are certainly possible.

<sup>1</sup>[http://simile.mit.edu/repository/ontologies/official/cidoc\\_crm.rdf](http://simile.mit.edu/repository/ontologies/official/cidoc_crm.rdf)

<sup>2</sup><http://simile.mit.edu/repository/ontologies/official/foaf.owl>

<sup>3</sup>[http://www.w3.org/2003/01/geo/wgs84\\_pos](http://www.w3.org/2003/01/geo/wgs84_pos)

<sup>4</sup><http://www.w3.org/2000/10/swap/pim/contact>

<sup>5</sup><http://xmlns.com/wordnet/1.6/Person>

<sup>6</sup><http://www.aifb.uni-karlsruhe.de/WBS/meh/mapping/data/swrc1a.rdf>

<sup>7</sup><http://www.isi.edu/webscripiter/person.o.daml>

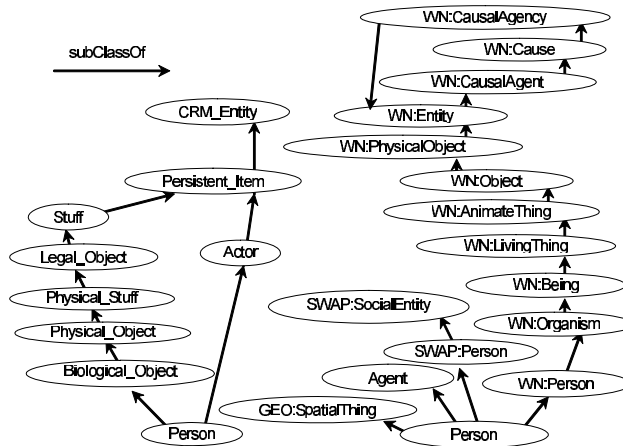


Figure 1. Different Person class hierarchies

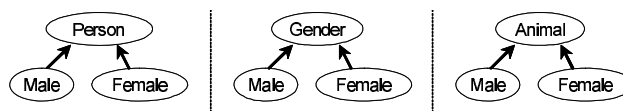


Figure 2. Three possible superclass designs for Male and Female

Sometimes superclass and subclass properties may give hints about property design. Figure 4 shows four such cases. Graphs (a)<sup>8</sup> and (b)<sup>9</sup> show superclass property definitions of Person. Graphs (c)<sup>10</sup> and (d)<sup>11</sup> show Person's subclass property definitions. (Graph (d) also shows direct Person properties.) We now look at each graph in detail.

Graph (a) shows Actor Class properties such as possesses, has\_right\_on, is\_current\_owner\_of, and surrendered\_custody\_through, which describe concepts used in cultural heritage documentation.<sup>12</sup> Graph (b) shows geographic location properties, GEO:long, GEO:lat, and GEO:alt, and a language property, SWAP:motherTongue, which describe the social network of friends.<sup>13</sup> Graph (c) shows Employee properties such as job\_title and expertise, which reflect the enterprise domain. Lastly, graph (d) shows Student properties such as advisor and takesCourse, which reflect the academic domain.

Many possible properties for a given class may exist, but when it comes to actually designing necessary properties for a class, metadata designer may fail to come up with various properties necessary for a specified class. For instance, the

<sup>8</sup>[http://simile.mit.edu/repository/ontologies/official/cidoc\\_crm.rdf](http://simile.mit.edu/repository/ontologies/official/cidoc_crm.rdf)

<sup>9</sup><http://simile.mit.edu/repository/ontologies/official/foaf.owl>

<sup>10</sup><http://www.daml.ri.cmu.edu/ont/homework/atlas-cmu.daml>

<sup>11</sup><http://www.cs.umd.edu/projects/plus/DAML/onts/univ1.0.daml>

<sup>12</sup>CIDOC Conceptual Reference Model (CRM), <http://cidoc.ics.forth.gr/>

<sup>13</sup><http://www.foaf-project.org/>

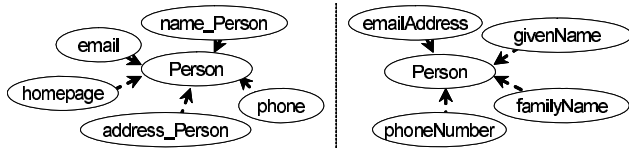


Figure 3. Different Person property designs

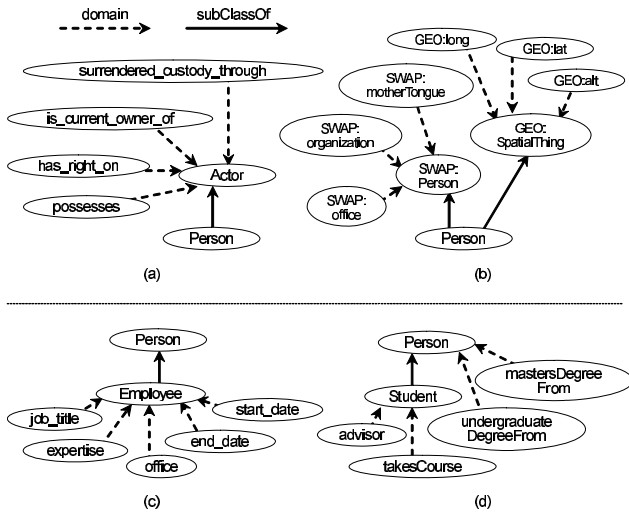


Figure 4. Various Person superclass properties (a)&(b) and subclass properties (c)&(d)

designer may come up with job\_title for Employee, but fail to define expertise as Employee property. We describe this difficulty as the problem of diverse property coverage.

### 2.3. Class or Property Design Dilemma

Classes provide an abstraction mechanism for grouping resources with similar characteristics, but whether or not to design an item as class is difficult to determine. Consider a task where items ‘webpage’ and ‘email’ must be designed. There could be three ways of designing these items: as class (Fig. 5 top), as both class and property (middle), or as property (bottom).

The decision whether or not to design an item as class will eventually be made during the metadata design process, but note that this decision in turn influences the property design especially for those properties defining these ‘class/not class’ items as range. For example, the two graphs placed in the middle in Figure 5 show properties hasWebpage (left) and hasEmail (right) defined as predicates linking the subject Person and objects Webpage and Email. But the property types will be different for the middle and bottom graphs when they are defined using such ontology language as OWL, since the middle case will define hasWebpage and

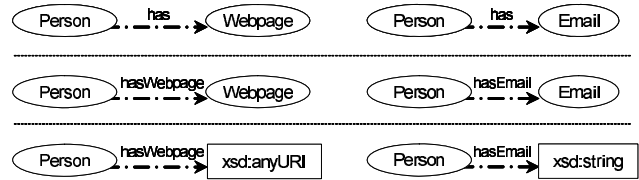


Figure 5. Class or property design dilemmas

hasEmail as owl:ObjectProperty whereas the bottom case will define them as owl:DatatypeProperty.

In sum, the decision whether or not to design an item as class can influence the outcome of the property design and vice versa; the property design can influence the class design. We describe this difficulty as class or property design dilemma.

## 3. Referring to Existing Metadata

We now propose a metadata referring method that organizes existing metadata defined in application ontologies in table formats. If somewhere on the Internet there exist metadata that relates to the metadata the designer has difficulty in designing, by referring to them, the designer may obtain various design ideas including those that the designer could not think of. These metadata examples may aid the designer to make better design decisions.

### 3.1. Referring to Class Hierarchies

An example of subsumption relation uncertainty discussed in section 2.1 was being unsure of what to design as superclass of Male and Female Class (Fig. 2). To address this problem, we propose a metadata table in the form of Table 1, which provides subsumption relation information collected from multiple ontologies. Table 1 shows seven Male class hierarchies defined in different application ontologies. The first column shows subclass metadata, which in this case is Male Class. The second column shows superclass metadata. The third column shows the ontology URL of the defined metadata.

In this table, we find that many ontologies define Animal Class as the superclass of Male Class. One reason for this is that a single ontology is being reused either partially or wholly by other ontologies. This result may be unexpected or even go against the intuition of some designers, but such repeated usage of the same class hierarchy design may indicate that a consensus has been formed by several designers about that class hierarchy definition. Being able to obtain such understanding may be helpful when designing class hierarchies.

**Table 1. Seven Male superclass definitions**

SUBCLASS	SUPERCLASS	ONTOLOGY URL
Male	Individual	<a href="http://www.tt.cs.titech.ac.jp/~fukatani/kadai/yasuda/genealogy.owl">http://www.tt.cs.titech.ac.jp/~fukatani/kadai/yasuda/genealogy.owl</a>
Male	Animal	<a href="http://www.daml.org/2000/10/daml-ex.daml">http://www.daml.org/2000/10/daml-ex.daml</a>
MalePerson	Person	<a href="http://www.cs.umd.edu/~evren/cm828y/hw1/ontology.daml">http://www.cs.umd.edu/~evren/cm828y/hw1/ontology.daml</a>
Male	Animal	<a href="http://www.daml.org/validator/examples/ont3.daml">http://www.daml.org/validator/examples/ont3.daml</a>
Male	Animal	<a href="http://www.atl.external.lmco.com/projects/ontology/ontologies/animals/animalsB.owl">http://www.atl.external.lmco.com/projects/ontology/ontologies/animals/animalsB.owl</a>
Male	Animal	<a href="http://www.w3.org/2000/10/swap/test/dpo/daml+oil-ex.daml">http://www.w3.org/2000/10/swap/test/dpo/daml+oil-ex.daml</a>
Male	Animal	<a href="http://www.srdc.metu.edu.tr/~yildiray/example.daml">http://www.srdc.metu.edu.tr/~yildiray/example.daml</a>

One may wonder why other superclass definitions such as Gender Class do not appear in the table. One reason may be that the concept ‘male’ may not be defined as subclass of Gender. For example, if ‘male’ and ‘female’ were defined as individuals of Gender Class rather than as subclasses, the subsumption relation will not be present.

What is important is that abundant examples on existing class hierarchy definitions are gathered and presented to the designer. By referring to these numerous examples, the designer may obtain design ideas including those that he or she could not think of. For instance, a designer designing metadata for a zoo system may initially consider Sex Class as a superclass for Male and Female, but after referring to the class hierarchy table such as Table 1, the designer may discover a new design direction, Animal Class, as more probable superclass for Male and Female.

### 3.2. Referring to Various Properties

The problem of diverse property coverage deals with the designer falling short of designing various properties for a given class. To address this problem, we propose a metadata referring method that aligns similar properties defined in numerous ontologies.

Table 2 shows an example of such aligned properties: various Person Class properties selected from seven different application ontologies<sup>14</sup> are organized and aligned in rows according to similar properties. Words in each cell indicate Person properties. Note that Person Class not necessarily points to the exact same class. Each column beginning with the second column (ONT1 to ONT7) indicates

<sup>14</sup>The URLs of the ontologies in Table 2 are as follows:  
 ONT1. <http://simile.mit.edu/repository/ontologies/official/foaf.owl>  
 ONT2. <http://daml.umbc.edu/ontologies/fitalks/person>  
 ONT3. <http://orlando.drc.com/daml/ontology/Person/current/>  
 ONT4. <http://www.aifb.uni-karlsruhe.de/WBS/meh/mapping/data/swrc1a.rdf>  
 ONT5. <http://dublincore.org/2000/06/07-org>  
 ONT6. <http://www.isi.edu/webscripser/person.o.daml>  
 ONT7. <http://www.openhealth.org/ASTM/simplified-model.rdf>

an individual ontology. Each row indicates similar properties defined across different ontologies. For example, each cell in the second row shows Person’s ‘name’ properties defined in different ontologies. Although each ontology defines different ‘name’ properties, for example, ONT3 defines six kinds of name properties (lastName, firstName, middleName, displayName, alias and nickName) whereas ONT5 defines a single name property, all aim to define metadata of Person’s name.

It is difficult to determine which design is better since metadata design objectives and intended domain may vary. However, it is possible to know what properties of a given class are missing or present if property definitions in multiple ontologies are compared to one another. For example in the last row of Table 2, Gender property is defined in ONT2 and ONT3, but missing in other ontologies. Similarly, properties related to ‘phone’ are defined in ONT2, ONT3, ONT4 and ONT6, but not in other ontologies.

By organizing properties according to similar properties, metadata designer can know what properties are commonly defined in many ontologies and what are discretely defined in few or a single ontology. Based on this, the designer may interpret that commonly defined properties constitute a set of fundamental properties of that specified class whereas discrete properties define special, non-fundamental properties. Moreover, such highlighting of property differences may enable the designer to discover which ontology is most relevant or suitable to refer to or reuse. All these discoveries are enabled by comparing properties of a given class defined in multiple ontologies.

### 3.3. Referring to Classes and Properties

The class or property design dilemma concerns with the difficulty of deciding whether to design an item as class and/or property. An example of this dilemma discussed in section 2.3 was whether to design ‘webpage’ and ‘email’ item as class or property (Fig. 5). Here, we focus on ‘email’ item to see how existing class and property examples can give ideas about class and/or property design.

Table 3 shows an overview of Person metadata defined in one ontology. The first row shows URL of the ontology. The second and third rows show class hierarchy definitions of Person; the left column shows subclasses and the right column shows direct superclasses of the subclasses on the left. The fourth and fifth rows show property definitions of the classes given in the third row. This metadata table organizes the basic class hierarchy and property information of a given class, here in this case, Person.

Now consider creating similar tables for other ontologies, and skimming through them. Focusing on the ‘email’ definitions in several tables, the designer can obtain knowledge about how existing metadata related to ‘email’ are de-

**Table 2. Overview of Person Class properties defined in seven different ontologies**

CATEGORY	ONT1	ONT2	ONT3	ONT4	ONT5	ONT6	ONT7
NAME	family_name surname first_name	lastName firstName	lastName firstName middleName displayName alias nickName	name_Person	name	familyName givenName	person.name
TITLE		title	title				
HOMEPAGE	homepage workInfoHomepage workplaceHomepage schoolHomepage	homepage		homepage	affURL		
EMAIL	mbox mbox_sha1sum	email	hasEmail	email	email	emailAddress	
PHONE		homePhone officePhone cellphone	hasTelephoneNumber	phone		phoneNumber	
ADDRESS		homeAddress officeAddress	hasAddress	address_Person			address
GENDER		gender	gender				

**Table 3. An overview of Person metadata**

URL	http://protege.stanford.edu/plugins/owl/owl-library/ka.owl	
SUBCLASS	SUPERCLASS	
Person	Object	
Employee	Person	
PROPERTIES	DOMAIN	RANGE
phone	Person	http://www.w3.org/2001/XMLSchema#string
firstName	Person	http://www.w3.org/2001/XMLSchema#string
lastName	Person	http://www.w3.org/2001/XMLSchema#string
name	Person	http://www.w3.org/2001/XMLSchema#string
middleInitial	Person	http://www.w3.org/2001/XMLSchema#string
address	Person	http://www.w3.org/2001/XMLSchema#string
email	Person	http://www.w3.org/2001/XMLSchema#string
photo	Person	http://www.w3.org/2001/XMLSchema#string
fax	Person	http://www.w3.org/2001/XMLSchema#string
headOfGroup	Employee	ResearchGroup
headOf	Employee	Project
affiliation	Employee	Organization
worksAtProject	Employee	Project

signed. In the case of Table 3, the designer can know that ‘email’ is designed as property having Person as domain and ‘xsd:string’ as range. All in all, overview tables like Table 3 will enable a designer to easily grasp the design patterns of a certain item. Surely, circumstances will dictate whether or not to design an item as class or property, but numerous metadata examples will provide knowledge of existing design patterns of the target items.

#### 4. Evaluation

To assess the effectiveness of our referring method, we conducted an evaluation experiment on human subjects. Subjects performed a same metadata design task twice, at first without referring to the tables and then referring to the tables. The two design performances were then compared.

#### 4.1. Method

**Subjects** were 41 undergraduate students in Artificial Intelligence course. They received course credit for their work done in the experiment. **A task** of designing semantic network of the university was assigned to the subjects. Nine domain keywords (university, student, name, undergraduate department, graduate student, Artificial Intelligence, Kyoto University, course, lecturer, research) were given at the start of the experiment to be incorporated into the network.

**Experiment procedure** was carried out as follows: Each subject individually designed the semantic network twice, each time for 20 minutes. During the first trial, the subjects did not receive any reference material. At the second trial, the subjects received a printout of the **metadata tables** that had similar layouts as Tables 1, 2 and 3, but that contained metadata about Student. This printout (from here on after *table*) was created from actual ontologies describing the academic domain.

**Data** collected from human subjects were 81 semantic networks: 41 of them were designed without any reference support; 41 were designed with the support of the *tables*. From each network, individual links were selected along with the two nodes connected to the link. These ‘[node]-link-[node] triples (from here on, triples)’ were then analyzed. **Triples** were divided into three groups: (i) triples that have ‘subClassOf’ links, (ii) triples that contain property definition, and (iii) triples that consist of ‘name’ related nodes and links. These three groups of triples were analyzed separately to assess whether the *tables* influenced human subjects’ performance of (a) class hierarchy design, (b) diverse property design, and (c) appropriate metadata design of the ‘name’ item.

**Table 4. Class hierarchy triple summary**

Class Hierarchy Category		Not Referred	Tables Referred
Correct Triple	Person	14	57
	Organization	10	13
	Activity	0	3
	Other	4	4
	Total	28	77
Wrong Triple	Instance	25	17
	Person	26	25
	Organization	27	31
	Activity	8	13
Total Triple	114	163	

## 4.2. Results and Discussion

Analyses of triples have revealed that, referring to the *tables* support class hierarchy and diverse property design, but does not support class and/or property design decision. Note that we regard the concepts and relations in semantic networks to be equivalent to classes and properties in ontologies, and analyzed triples accordingly.

**Analyses of Class Hierarchy Triples** A total of 277 class hierarchy triples were designed: 114 triples were designed without referring to the *tables*; 163 triples were designed while referring to the *tables*. One human referee judged each class hierarchy triple's correctness twice, placing two-month interval between the two judgments. Final judgments were made based on the two judgments. Table 4 summarizes the number of correct and wrong triples according to the class categories. Some class hierarchy examples are 'PhDStudent-Student' (an example of a correct triple. The triple is read, "PhDStudent Class is subclass of Student Class"), 'UndergraduateStudent-Department' (wrong triple) for person-related triples; 'University-ResearchInstitution' (correct), 'Department-University' (wrong) for organization-related triples; and 'Research-Activity' (correct), 'Lecture-Department' (wrong) for activity-related triples.

Notice how the number of person-related correct triples increased from 14 to 57 when the *tables* were referred. Moreover, the number of wrong triples has decreased for person-related class hierarchies, particularly those that define "person-related class to be subclass of organization-related class." The number decreased from 18 to 5 (these numbers are included in the person-related wrong triples 26 and 25 shown in Table 4.) We consider such increase in the design performance to be the effect of the *tables*.

**Analyses of Property Triples** A total of 433 property triples were designed of which 97 had undefined properties (triples with blank links). Table 5 summarizes property triples according to properties with similar domains.

**Table 5. Property triple summary**

Domain Category	Not Referred	Tables Referred	$\Delta$ (BL)
Person	98 (27)	183 (18)	+85 (-9)
Organization	43 (17)	47 (15)	+4 (-2)
Activity	20 (8)	22 (2)	+2 (-6)
Other	14 (6)	6 (4)	-8 (-2)
Total Triple	175 (58)	258 (39)	+83 (-19)

**Table 6. 'Name' related triple summary**

Triple Design Pattern	Not Referred	Tables Referred
[class]-name-string	0	2
[class]-name-instance	4	9
[class]-name-[name]	1	11
[class]-property-[name]	12	20
[name]-property-[class]	13	6
Total Triple	30	48

The numbers inside the parentheses indicate the number of blank link triples (blank properties). Some examples of property triples are 'Lecturer-teach-Course' (the triple is read, "Lecturer teaches course") for person-related property triples, 'Laboratory-member-GraduateStudent' for organization-related triples, and 'Research-hasResearcher-Lecturer' for activity-related property triples.

Notice how the number of person-related properties nearly doubled when the *tables* were referred to. The last column of the Table 5 shows the increase in number of triples before and after referring to the *tables*. Notice also how the blank link properties have decreased. We interpret this change to come from the influence of the *tables*.

**Analyses of 'Name' Related Triples** A total of 78 'name' related triples were designed, 30 of them before and 48 of them after referring to the *tables*. We defined '[some class]-name-string' to be an exemplary metadata design pattern for the 'name' related triples, and analyzed the triples. However, only one human subject succeeded in defining the 'name' related triple in the exemplary pattern. Table 6 shows other design patterns identified in the triples. We think that metadata examples given by the *tables* (similar to Table 3) contained unfamiliar xsd:string expression for 'string', and this hindered subjects' acquiring of the exemplary design pattern.

## 5. Related Works

As more ontologies are becoming available, ontology libraries and metadata search engines that store and retrieve ontologies are also becoming available: SchemaWeb<sup>15</sup>,

<sup>15</sup><http://www.schemaweb.info/search/Search.aspx>

Protege OWL Library<sup>16</sup> and DAML Ontology Library<sup>17</sup> are some examples. Although these ontology libraries can search part or whole ontology, they do not organize existing metadata in table formats. Ontology construction environments such as Ontolingua Server [3], SWOOP [8] and Protégé [14] allow direct manipulation or reuse of existing ontologies, but we do not provide an editing environment: rather, we focus on presenting overviews of existing metadata defined in numerous application ontologies. In terms of using table, [17] utilizes the relation represented in the structures of the table to extract ontology. Table data such as Excel files are used as input to produce ontology output. In contrast, our approach uses existing ontologies as input to generate metadata table output. In our case, tables are utilized as containers for organizing existing metadata information.

## 6. Conclusion

We identified the following metadata design difficulties:

- **Subsumption relation uncertainty**  
Designing class hierarchy is difficult.
- **Problem of diverse metadata coverage**  
Designing diverse properties for a class is difficult.
- **Class or property design dilemma**  
Determining whether an item should be designed as class and/or property is difficult.

To address these difficulties, a metadata referring method that provides table format overviews of existing metadata was proposed. Our method organizes metadata defined in numerous application ontologies in table formats to provide the following design information:

- existing class hierarchy definitions
- similar and specific property definitions
- overviews of class hierarchy and property definitions of a given class

Based on the evaluation experiment, we found our method to be useful in designing class hierarchies and diverse properties, but not in deciding whether to design an item as class and/or property.

## Acknowledgement

This work was supported by a Grant-in-Aid for Scientific Research (A)(15200012, 2003-2005) from Japan Society for the Promotion of Science (JSPS).

## References

- [1] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

<sup>16</sup><http://protege.stanford.edu/plugins/owl/owl-library/>

<sup>17</sup><http://www.daml.org/ontologies/>

- [2] R. Brachman. What IS-A is and isn't: an analysis of taxonomic links in semantic networks. *IEEE Computer*, 16(10):30–36, 1983.
- [3] A. Farquhar, R. Fikes, and J. Rice. The ontolingua server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707–727, June 1997.
- [4] M. Fernández-López and A. Gómez-Pérez. Overview and analysis of methodologies for building ontologies. *Knowledge Engineering Review*, 17(2):129–156, June 2002.
- [5] T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, June 1993.
- [6] N. Guarino. Formal ontology and information systems. In *Proceedings of the First International Conference on Formal Ontology in Information Systems (FOIS'98)*, pages 3–15, Amsterdam, July 1998. IOS Press.
- [7] N. Guarino and C. Welty. Evaluating ontological decisions with ONTOCLEAN. *Communications of the ACM*, 45(2):61–65, February 2002.
- [8] A. Kalyanpur, B. Parsia, and J. Hendler. A tool for working with web ontologies. *International Journal on Semantic Web and Information Systems*, 1(1), January-March 2005.
- [9] A. Kumar, P. Ciccarese, B. Smith, and M. Piazza. Context-based task ontologies for clinical guidelines. In *Proceedings of the Workshop on Medical Ontologies*, pages 81–94, Rome, Italy, 2003. IOS Press.
- [10] D. Lenat and R. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project*. Addison-Wesley, Reading, Massachusetts, 1990.
- [11] T. Miki, S. Nomura, and T. Ishida. Semantic web link analysis to discover social relationships in academic communities. In *IEEE/IPSJ Symposium on Applications and the Internet (SAINT2005)*, Trento, Italy, 2005.
- [12] K. Nakatsuka and T. Ishida. Content management for inter-organizational projects using e-mail metaphor. In *IEEE/IPSJ Symposium on Applications and the Internet (SAINT2006)*, Phoenix, Arizona, 2006.
- [13] I. Niles and A. Pease. Towards a standard upper ontology. In *Proceedings of the First International Conference on Formal Ontology in Information Systems (FOIS-2001)*, pages 2–9, Ogunquit, Maine, October 2001. ACM Press.
- [14] N. Noy, M. Sintek, S. Decker, M. Crubezy, R. Ferguson, and M. Musen. Creating semantic web contents with Protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.
- [15] R. Studer, R. Benjamins, and D. Fensel. Knowledge engineering: principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197, 1998.
- [16] B. Swartout, P. Ramesh, K. Knight, and T. Russ. Toward distributed use of large-scale ontologies. In *Proceedings of the Workshop on Ontological Engineering, AAI Spring Symposium (AAI'97)*, pages 138–148, Menlo Park, California, 1997. AAAI Press.
- [17] M. Tanaka and T. Ishida. Ontology extraction from tables on the web. In *IEEE/IPSJ Symposium on Applications and the Internet (SAINT2006)*, Phoenix, Arizona, 2006.
- [18] M. Uschold and M. Gruninger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11(2):93–155, June 1996.