Citation:

Martin Hepp: *Products and Services Ontologies: A Methodology for Deriving OWL Ontologies from Industrial Categorization Standards*, *Int'l Journal on Semantic Web & Information Systems* (IJSWIS), Vol. 2, No. 1, pp. 72-99, January-March 2006.

Official version available at

http://www.idea-group.com/articles/details.asp?ID=5577

Products and Services Ontologies: A Methodology for Deriving OWL Ontologies from Industrial Categorization Standards

Martin Hepp

Digital Enterprise Research Institute (DERI), Innsbruck, Austria
Florida Gulf Coast University, Fort Myers, FL, USA

mhepp@computer.org

http://www.heppnetz.de

Abstract

In order to use Semantic Web technologies for the automation of E-Business tasks, like product search, content integration, or spend analysis, we need domain ontologies for products and services. The manual creation of such ontologies is problematic due to (1) the high degree of specificity, resulting in a very large number of needed concepts, and (2) the need for timely ontology maintenance due to the high conceptual dynamics caused by innovation in the products and services domain; and due to cost, since building such ontologies from scratch requires significant resources. On the other hand, industrial products and services categorization standards, like UNSPSC[1], eCl@ss[2], eOTD[3], or the RosettaNet Technical Dictionary[4] reflect some degree of community consensus and contain, readily available, a wealth of concept definitions plus a hierarchy. They can thus be valuable input for creating domain ontologies. However, the transformation of existing taxonomies into useful ontologies is not as straightforward as it appears, mainly because simply taking the hierarchy of concepts, which was originally developed for some external purpose other than ontology engineering, as the subsumption hierarchy can yield useless ontologies. In this paper, we (1) argue that for the products and services domain, deriving ontologies from industrial taxonomies is more feasible than manual ontology engineering due to the large number of concepts and the high conceptual dynamics, (2) show that the interpretation and representation of the original semantics of the input standard, especially the taxonomic relationship, is an important modeling decision that determines the usefulness of the resulting ontology, (3) illustrate the problem by analyzing DAML+OIL, OWL, and RDF-S ontologies derived from UNSPSC and eCl@ss, (4) present a comprehensive methodology for creating products and services ontologies in OWL based on the reuse of existing standards like eCl@ss and UNSPSC, and (5) demonstrate this approach by transforming eCl@ss 5.1 into a practically useful products and services ontology.

Keywords: Ontology engineering, E-Commerce, E-Business, E-Procurement, Business Ontologies, Products and Services, Taxonomies, Reuse, OWL, UNSPSC, eCl@ss, eOTD, RNTD, RosettaNet

---

[1] http://www.unspsc.org/

[2] http://www.eclass-online.com/

[3] http://registry.eccma.org/eotd/

[4] http://www.rosettanet.org/

1 Introduction

Products and services categorization standards (PSCS), like the UNSPSC, eCl@ss, eOTD, or the RosettaNet Technical Dictionary (RNTD) form a valuable set of concepts from the products and services domain and reflect some degree of consensus. They are thus a promising foundation for the creation of product ontologies for the use in the Semantic Web, but are not by themselves fully usable ontologies.

There already exist examples of products and services ontologies derived from PSCS, e.g. the RDF-S versions of UNSPSC by Klein (Klein, 2002), and the DAML+OIL versions of UNSPSC by McGuinness (McGuinness, 2001) and by Klein (Klein, 2002). However, those ontologies do not properly reflect the specific semantics of the underlying standards. There also exists an early prototype of a RDF representation of eCl@ss 4.1 created by (Bizer & Wolk, 2003), which is based on an OWL ontology for representing taxonomies that itself is OWL Full, because it applies object properties to classes. All three ontologies are one-time captures of the reused standards, while the standards themselves have undergone significant change in the meantime (Hepp, Leukel, & Schmitz, 2005a, 2005b). These shortcomings in combination limit the usefulness of available ontologies.

Ontology engineering has been the subject of research for a long time, and multiple methodologies for the creation of ontologies have been proposed. However, the script-based (i.e. automated) reuse of concepts from hierarchically ordered standards, which were not created with the rigor of knowledge representation in mind, creates different requirements and demands novel approaches, mainly because we cannot partition the concept space into concepts at our own will. In other words, we must automatically capture as much semantics as possible out of the existing

standards and have only limited control over the actual definition of the concepts. In general, we have to take the concepts as they are, because the size of those standards (between 20,000 and almost 60,000 classes and between 3,000 and 20,000 properties) renders manual steps in the process of importing the concepts unfeasible. On the other hand, we have to carefully analyze the constraints and dependencies resulting from the implicit assumptions of the standards creators. Additionally, the intension of the reused concepts will often be influenced by our interpretation of the original standard. When transforming an informal categorization scheme into a formal ontology using a standard ontology language, we often narrow down the semantics, which includes some degree of freedom and thus several modelling decisions. Sometimes we might even want to have controlled changes in meaning between the original standard and the resulting ontology in order to create more useful ontologies. This, however, must be done with great care and we then must make respective decisions transparent to the ontology user.

Related Work

Related work to this paper can be classified into the following main groups:

Ontology engineering methodologies, implicitly or explicitly focusing on the manual creation of new ontologies based on knowledge engineering principles. A comprehensive discussion of all approaches in this field is beyond the scope of this paper; for an overview, see e.g. (Fernández-López & Gómez-Pérez, 2002) and (de Bruijn, 2003).

Analysis of the meaning of taxonomic relationships, especially the fundamental work of (Brachman, 1983). This yielded the insight that there are multiple types of taxonomic relationships, which should be represented distinctively.

Contributions regarding Electronic Commerce ontologies mainly from the perspective of catalog data management and ontology mapping. (Fensel, McGuinness et al., 2001) discuss the

advantages of ontologies for the integration of heterogeneous and distributed information in the E-Commerce field. (Schulten et al., 2001) and (Fensel, Ding et al., 2001) focus on the problem of product data integration in B2B relationships. (Ng, Yan, & Lim, 2000) introduce the important distinction between content standardization versus content integration as two approaches for overcoming information heterogeneity. (Fairchild & de Vuyst, 2002) detail the benefits of standardized products and services coding from a business perspective. (Obrst, Wray, & Liu, 2001) discuss the challenges associated with the creation of ontologies for the product and service knowledge space, stressing the task of mapping among ontologies. (Kim, Kim, & Lee, 2002) propose catalog data integration based on the controlled merging of category hierarchies. (Kim, Lee, Chun, & Lee, 2004) discuss the problematic aspects of a naïve view on product classification as one "ideal" way of ordering, and introduce the important notion of any classification being a purpose-specific order and thus not universally useable.

Prototypes of products and services ontologies in standard ontology languages derived from UNSPSC. To our knowledge, there are currently two examples of UNSPSC transformations into ontology representation languages: The DAML+OIL and RDF-S variants created by Klein (Klein, 2002) and the DAML+OIL variant from the Knowledge Systems Laboratory at Stanford, see (McGuinness, 2001). Also, there exists an early prototype of a RDF representation of eCl@ss 4.1 created by Bizer and Wolk (Bizer & Wolk, 2003). A slightly different approach is presented in (Klein, 2001): Instead of transforming UNSPSC into an ontology, he creates a small ontology of properties that supports the usage of UNSPSC codes in annotations as literal values.

Methodologies for and experiences with the reuse of consensus in existing standards for the creation of ontologies. This is the most relevant field of work for this paper. Rector et al. (2001) discuss the transformation of tangled hierarchies, as e.g. such derived from ambiguous "broader

than / narrower than" taxonomies in library science, into formal ontologies. Recently, (Giunchiglia, Marchese, & Zaihrayeu, 2005) introduced the notion of *Formal Classification* and proposed an approach of how to disambiguate the local labels of a given classification systems into logical formulae. (Paslaru Bontas, 2005) points to the importance of understanding the (often implicit) context, especially information about "scope, history, purpose, authors, or best practices of [its] usage" when reusing conceptualizations of a domain. Kashyap et al. (2003) describe the experiences gained while transforming the constructs of an existing Semantic Network in the medical domain into an OWL ontology. (Wielinga, Schreiber, & Sandberg, 2001) show the reuse and semantic enrichment of an existing taxonomy and demonstrate this for the Art and Architecture Thesaurus (AAT). (Wielinga, Wielemaker, Schreiber, & van Assem, 2004) and (van Assem, Menken, Schreiber, Wielemaker, & Wielinga, 2004) are consequent works of this stream of research. An important characteristic of (Wielinga et al., 2004) and (van Assem et al., 2004) is that they leave the limits of OWL DL in order to capture semantics contained in the original taxonomy, namely to be able to treat classes as instances and vice versa. An important distinction between the proposal of (Wielinga et al., 2004) and the work presented in our paper is that they treat the taxonomic relationship in the input taxonomy as a special form (i.e. rdfs:subPropertyOf) of rdfs:subClassOf), whereas in our approach, the original taxonomic relationship is regarded as a weaker, more general, and semantically less specific relationship. In other words, we regard rdfs:subClassOf as a specific form of a general taxonomic relationship, i.e. a rdfs:subPropertyOf of the more general idea of a taxonomic relationship. This approach is explained below in greater detail.

Our Contribution

In this paper, we (1) argue that for the products and services domain, deriving ontologies from industrial taxonomies is more feasible than manual ontology engineering due to the large number of concepts and the high conceptual dynamics, (2) show that existing DAML+OIL and RDF-S domain ontologies derived from products and services standards, namely UNSPSC, are based on incorrect assumptions about the meaning of the taxonomic relationship in those standards, limiting the practical use of the resulting ontologies; (3) detail how the interpretation and representation of the taxonomic relationship is an important modelling decision that determines the usefulness of the resulting ontology, (4) present a comprehensive methodology for creating products and services ontologies in OWL Lite based on the reuse of existing standards like eCl@ss and UNSPSC, (5) develop a pragmatic versioning mechanism that is compatible with existing OWL Lite constructs, (6) show how the valuable property recommendations for classes and the value recommendations for properties found in eCl@ss and RNTD can be represented without violating the limits of OWL Lite and the Open World Assumption (OWA). Finally, we demonstrate the application of our novel approach to eCl@ss 5.1 and provide a comprehensive and practically useful ontology for the products and services domain. To our knowledge, the resulting ontology is the first real-world domain ontology for products and services.

The structure of the paper is as follows. Section 2 summarizes the characteristics and components of prominent products and services categorization standards (PSCS). Section 3 elaborates on the representational challenges and presents a comprehensive proposal of transforming products and services categorization schemes into OWL ontologies. Section 4 evaluates and illustrates the proposal by applying it to the categorization standard eCl@ss.

Section 5 discusses the benefits, limitations, and implications of our approach. Section 6 summarizes the results.

2 Characteristics of Products and Services Categorization Standards

There are countless approaches for the classification of goods, ranging from rather coarse taxonomies, created for customs purposes and statistics of economic activities, like the North American Industry Classification System (NAICS) and its predecessor SIC (see (U.S. Census Bureau, 2004)), to expressive descriptive languages for products and services, like eCl@ss, eOTD, or the RosettaNet Technical Dictionary. The UNSPSC, widely cited as an example of a product ontology, is in the middle between those two extremes, providing an industry-neutral taxonomy of products and services categories, but no standardized properties for the detailed description of products.

It is out of the scope of this paper to list and compare all available standards in this area, but one can say that UNSPSC, eCl@ss, and eOTD are currently the most important horizontal standards (i.e. covering a broad range of industries), and RNTD should be included in the analysis because of its high degree of detail, albeit limited to a narrow segment of products.

Intended Usage

The basic challenge when deriving ontologies from existing taxonomies is to understand the implicit assumptions of its creators and its user community, and to then represent as much as possible of the original semantics using the allowed constructs of the respective ontology language. The hierarchies of both UNSPSC and eCl@ss were created on the basis of practical aspects of procurement, treating those commodities that “somehow” belong to a specific category, as descendents of this closest category. This makes “ice” a subcategory to “non-alcoholic beverages” in UNSPSC and “docking stations” a subcategory of “computers” in

eCl@ss. This is fine as long as the taxonomy is used for analytical purposes (e.g. spend analysis; the aggregation of spendings based on a corporate cost accounting scheme) or grouping products in electronic catalogs that will be used by humans, and this is exactly the typical form of intended usage. However, if we reuse the concepts from UNSPSC or eCl@ss for building an ontology, we have to carefully analyze the implications resulting from the originally intended purpose of those standards. As a consequence, the transformation of such taxonomies into useful ontologies is not as straightforward as it appears, because simply taking the hierarchy of concepts, which was originally developed for some external purpose other than ontology engineering, as the subsumption hierarchy using e.g. rdfs:subClassOf can yield useless ontologies. Often, the original meaning of the taxonomic relationship is “A is in some context a more specific category of B” rather than a strict subClassOf relationship with the typical semantics “For all A being a descendent of B, every instance of A shall also be an instance of B”.

Components

All of those standards reflect a varying combination of the following components, which can be reused for a derived ontology. It is crucial to observe that most standards were not created with the rigor of ontology engineering, or knowledge representation in general, in mind, but for very specific purposes, leading to several implicit assumptions and inconsistencies. Failure to understand the context of the underlying standard will yield inconsistent and almost useless ontologies.

Product Classes

All PSCS are based on a set of product categories that aim at grouping *similar* products. However, this grouping is often influenced by the purpose of the PSCS. For example, the

categories can aggregate products by the *nature* of the products or by their intended *usage*. This is most obvious when it comes to classifying chemicals: The same substance can be used for multiple purposes, and classification systems can be based on classes for the chemical substances, categories for the various usages, or a combination of both. This can create confusion, as there is an N:M correspondence between the nature of a product and product usages.

The intensions of the product categories are usually captured in a rather informal way, ranging from just very short class names to quite precise natural language definitions, sometimes available in multiple languages.

Hierarchy of Classes

Most PSCS arrange the classes in hierarchical order. It is crucial to understand that this hierarchy is directly connected to the intended usage of the PSCS. For example, eCl@ss was designed with the idea of grouping products from the perspective of a buying organization or a purchasing manager. So it regards products as similar that are (1) bought from the same range of suppliers, (2) needed by the same consumers inside the organization, (3) billed using the same cost accounting categories, or any combination of those.

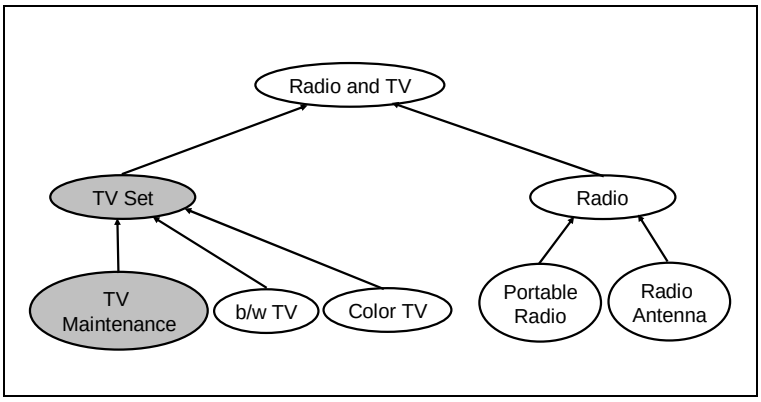

Figure 1: Example of a products and services taxonomy

A typical consequence of this is that related categories (e.g. service or maintenance) are often descendents of the general good category. The category “TV Set Maintenance” will thus be regularly a descendent of “TV Sets”, and “Oil for Sewing Machines” will be a descendent of “Sewing Machinery”. As a summary, the semantics of this relationship is generally not more specific than “Class A is some special kind of Class B”. Figure 1 gives an example of a taxonomy for products and services.

It is also important to note that, though all PSCS internally have unique identifiers for the classes, the visible identifier for a class is often just the position in the hierarchy (e.g. 1-3-0-0 for machinery and 1-3-1-0 for agricultural machinery). This is in general a bad practice, as it makes it hard to rearrange the hierarchy without destroying existing references to categories. Thus, it makes much more sense to use the true identifier of the classes as the concept identifier, for it also allows the use of multiple hierarchies for different purposes (e.g. one for cost accounting and one for purchasing management).

Dictionary of Properties

More sophisticated PSCS include a dictionary of standardized properties that can be used to describe product instances (i.e. an actual product that can be sold and bought) or product models (i.e. the brand of a commodity) in more detail and thus allow parametric search. Parametric search means querying the product space using restrictions on the properties of the product (e.g. “all TV sets that weigh less than 5 kg and have a screen size greater than 10 inches”). Usually, those property dictionaries contain a quite rich definition of the contained properties, including not only sophisticated data typing, but also references to international standards for the unit of measurement. From the perspective of ontology engineering, most of the properties are data type

properties, though there are also recommended sets of enumerated values for a small part of the properties.

Enumerated Property Values

For properties where an arbitrary string is not sufficient to capture the value in a semantically unambiguous way, most PSCS maintain a list of supported values in a separate collection. For example, eCl@ss contains the property “Design of door” (ID BAA106001 in version 5.1), with the lexical space set to an arbitrary string of up to 32 characters. The set of supported values contains two entries that are recommended for the use with this property, “Folding doors” (ID AAA376001) and “Sliding doors” (ID AAA377001). The mapping between recommended values and such properties is usually kept in a separate relation.

Class-Property Relation

Most PSCS with a dictionary of properties include a mapping between classes and recommended properties, i.e. property sets per each class, sometimes referred to as “attribute lists”. However, the semantics of this assignment varies between different standards. It can range from very loose recommendations (as in eOTD) to a rather strict definition of those properties necessary and sufficient to completely describe an product model or product instance of the respective class (as in eCl@ss). The creation and maintenance of such property assignments is a tremendous task, and none of the existing horizontal PSCS has specific property lists for more than 50 % of their categories (see (Hepp et al., 2005b) for a comprehensive analysis).

Keywords

In order to support a human user in finding the proper concepts, most PSCS contain keywords for classes, properties, and values. Those keywords are often no true synonyms and must thus be used with care. Quite frequently, more specific words (e.g. “ink-jet paper”) point to the more

generic category that fits best (e.g. “office paper”) and the user must keep in mind that the found concept is less specific than the initial keyword.

Size

UNSPSC, a standard hierarchs for products and services and often referred to as a business ontology, contains 20,789 categories (in version 7,0901), and eCl@ss, a similar but more expressive standard, contains 25,658 categories plus 5,525 precisely defined object and datatype properties (in version 5.1). The amount of concepts for services ranges from 21 % in UNSPSC (4313 categories) to 4 % in eCl@ss (1064 categories), which clearly shows that the scope of both standards is not limited to tangible products. For a quantitative analysis of the content and domain coverage of products and services hierarchies, see (Hepp et al., 2005a) and (Hepp et al., 2005b).

Versioning Dynamics

Due to the continuous innovation in the product and services domain, all PSCS are a work in progress with multiple releases per year. We know from practitioners that they would prefer stable categorization standards and long release intervals; however it is important to note that more change in a standard is usually better, since it reflects active processing of change requests caused by innovation dynamics in the various industries.

Table 1: Amount of change in eCl@ss, eOTD, RNTD, and UNSPSC (taken from (Hepp et al., 2005a)).

	Release	Previous release	New classes per 30 days	Mean	Modified classes per 30 days	Mean
eCl@ss	5.0	4.1	865.0	279.6	157.4	1271.4
	5.0SP1	5.0	47.8		10.2	
	5.1beta	5.0SP1	131.6		4918.0	
	5.1de	5.1beta	74.1		0.0	
eOTD	10-01-2003	01-17-2003	6.1	6.2	0.0	0.0
	11-01-2003	10-01-2003	4.8		0.0	
	03-01-2004	11-01-2003	18.3		0.0	
	06-01-2004	03-01-2004	1.6		0.0	
	08-01-2004	06-01-2004	0.0		0.0	
RNTD	2.0	1.4	0.7	1.3	6.4	1.5
	3.0	2.0	2.4		1.0	
	3.1	3.0	0.0		0.1	
	3.2	3.1	0.0		0.0	
	4.0	3.2	3.4		0.0	
UNSPSC	6,0315	5,1001	907.8	233.9	135.6	47.5
	6,0501	6,0315	304.5		53.0	
	6,0801	6,0501	97.5		15.0	
	6,1101	6,0801	69.1		50.2	
	7,0401	6,1101	13.8		29.4	
	7,0901	7,0401	10.8		2.0	

In a recent analysis we found out that, on average, eCl@ss grows by as much as 280 and UNSPSC by 230 new classes per 30 days, see (Hepp, 2004), (Hepp et al., 2005a) and (Hepp et al., 2005b). Table 1 shows the amount of change dynamics in eCl@ss, eOTD, RNTD, and UNSPSC in detail. The high amount of change dynamics in e.g. eCl@ss and UNSPSC is an indicator for active user communities and maintenance, while the low amount in eOTD and RNTD shows that those are rather dead collections than actively maintained standards.

As a consequence, transforming a PSCS into an ontology is no one time task, but a recurring activity, and the time available for this tasks is limited.

3 Challenges of Deriving OWL Ontologies from Products and Services Categorization Standards

The basic challenge when deriving ontologies from PSCS is to represent as much as possible of the original, informal semantics from the taxonomy using the allowed constructs of the respective ontology language. In this section, we explain that previous attempts do not properly capture the semantics of the reused PSCS, especially with regard to the taxonomic relationship. As a consequence, we propose a novel methodology and show in detail how the core components of PSCS, as introduced in section 2, should be represented in OWL Lite.

Product Classes and Hierarchy

When taking the categories found in an existing taxonomy as the basis for the creation of an ontology, we face a fundamental problem: Unless there is a formal definition of the semantics of the taxonomic relationship, the intensions of the category concepts (e.g. the product classes) *are tangled with the interpretation of the taxonomic relationship.* In other words: If we lack a formal definition of either the hierarchical relationships or the category concepts, then *how we understand the taxonomic relationship determines the shape of the category concepts* and vice versa. Our choice of the interpretation of the taxonomic relationship affects the intension of the category concepts, and a chosen definition of the intension of the category concepts is compatible with only a specific interpretation of the taxonomic relationship.

As a consequence, we have some degree of choice over the intension of the ontology classes derived from the categories in the source taxonomy by our choice of the interpretation of the taxonomic relationship.

Two examples might illustrate this problem: The hierarchies of both UNSPSC and eCl@ss were created on the basis of practical aspects of procurement, treating those commodities that

"somehow" belong to a specific category as descendents of this closest category. This makes, as said, "ice" a subcategory of "non-alcoholic beverages" in UNSPSC and "docking stations" a subcategory of "computers" in eCl@ss.

Now, we can simply read the taxonomic relationship as a strict "rdfs:subClassOf" relationship (i.e. each instance of "ice" is also an instance of "non-alcoholic beverages" and each instance of "docking station" is also an instance of "computers"). Then, however, the intension of the class "computers" is no longer any computer, but the concept "computer" solely from the perspective of cost accounting or spend analysis, where an incoming invoice for a docking station can be treated as an incoming invoice for a computer. Similarly will "non-alcoholic beverages" no longer represent all non-alcoholic beverages, but the union of non-alcoholic beverages and related commodities.

The disadvantages of this interpretation of the taxonomic relationship as being equivalent to rdfs:subClassOf is obvious: We can no longer use the resulting classes for fully automated buying processes, because a search for all instances of "computers" will also return docking stations, and ordering the cheapest available instance of non-alcoholic beverages will very likely return just ice cubes.

Given now the fact that our interpretation of the taxonomic relationship in the source taxonomy determines the intension of the classes, we have to *actively choose* the most useful shape of the product classes and then derive the required interpretation for the taxonomic relationship.

Basically, each source taxonomy contains two concepts for each category node (see Figure 2): First the generic concept of the respective product or service category (e.g. "home appliances", marked as (1) in Figure 2) and second the related taxonomic concept (e.g. "home appliances and related goods", marked as (2) in Figure 2).

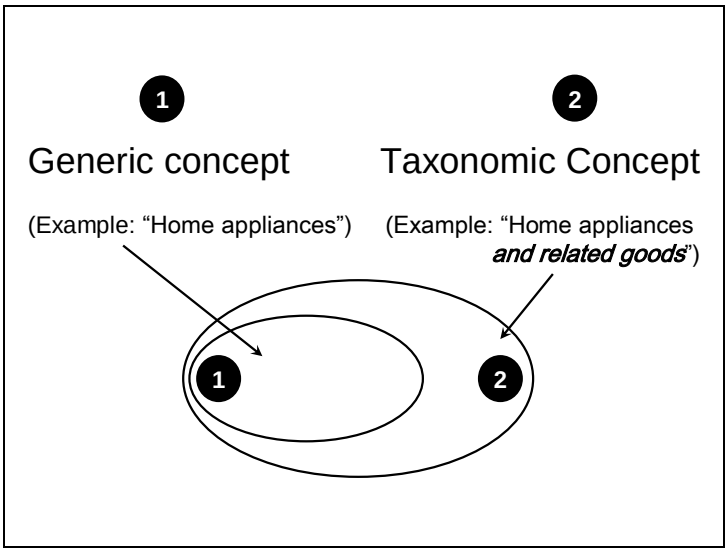

Figure 2: Relationship between generic concepts and taxonomy nodes

Depending on the implicit assumptions of the creators of the taxonomy, the relationship between the two concepts can deviate from the one shown in Figure 2, though this is the most frequently found one.

Now, even though the taxonomic relationship should not be interpreted as rdfs:subClassOf, we still want to capture the relationship as such, because there are applications like cost accounting where it is very useful. Thus, we have to find a way to represent generic categories for general use, while still preserving the hierarchical order for analytical purposes and similar applications. The most straightforward representation of UNSPSC and eCl@ss would be to create one ontology class for each category in the source standard, reflecting the *general* concept, and to capture the original hierarchy using a specific, transitive relationship “taxonomySubClassOf” between those classes. This approach is shown in Figure 3. This would allow having classes for the generally usable product categories while still being able to include the original hierarchy for queries (e.g. for finding all home appliances and all subcategories of home appliances in one query). Unfortunately, this is not possible in OWL Lite and DL, since relationships that link classes to classes can only be Annotation Properties in OWL, which cannot be made transitive.

Other ontology languages, like the WSML family of languages (de Bruijn et al., 2005), would support this, but are currently less widely in use.

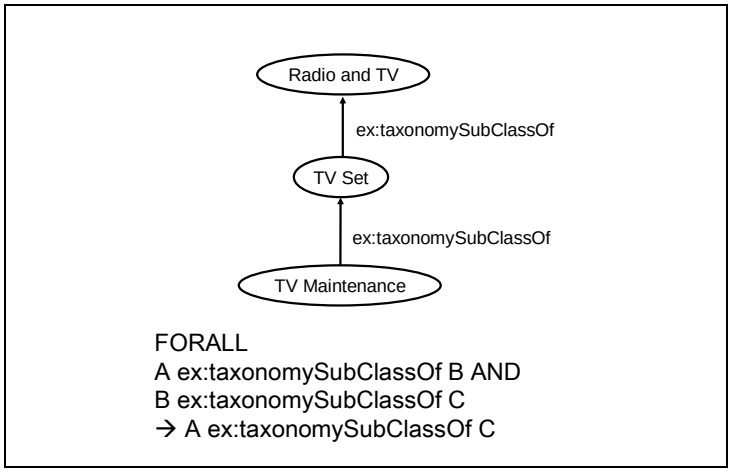

Figure 3: Representation using a new, transitive relationship „taxonomySubClassOf“

There are at least the following three known approaches of transforming a given taxonomy into an OWL Lite ontology, of which only the last two meet the stated requirements.

1. Create one class for each taxonomy category and assume that the meaning of the taxonomic relationship is equivalent to rdfs:subClassOf (Figure 4).
2. Create one class for each taxonomy category and represent the taxonomic relationship using an *Annotation Property* taxonomySubClassOf (Figure 5).
3. Treat the category concepts as instances instead of classes and connect them using a transitive *Object Property* taxonomySubClassOf (same as Figure 3 with the categories being instances instead of classes).

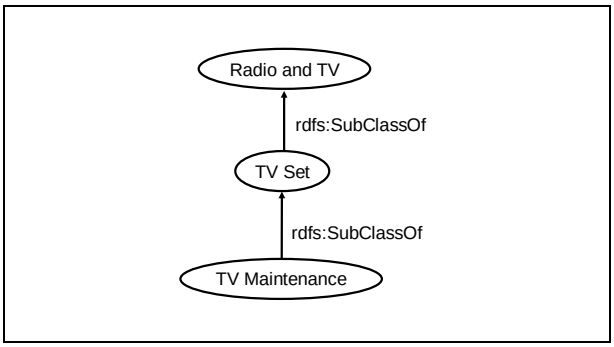

Figure 4: Option 1 in OWL: Understanding the concepts in the narrowest of all senses

Approach 1 is chosen by both available transformations of UNSPSC into products and services ontologies (it should be said that both ontologies base on different releases of UNSPSC and do thus not contain the exact same set of concepts). For example, the RDF Schema representation of UNSPSC created by Klein (Klein, 2002) contains statements of the following kind:

```
<rdfs:Class rdf:ID="Ice">
  <rdf:type rdf:resource="http://ontoview.org/schema/unspsc/1#Commodity"/>
  <rdfs:subClassOf rdf:resource="#Non_alcoholic_beverages"/>
  <unspsc:egci>014067</unspsc:egci>
  <unspsc:code>50.20.23.02</unspsc:code>
</rdfs:Class>
```

The DAML representation of UNSPSC created by the Knowledge Systems Laboratory at Stanford University (see (McGuinness, 2001)) uses the same structure:

```
<rdfs:Class rdf:ID="Docking-stations">
  <rdfs:subClassOf rdf:resource="#Computers"/>
  <unspsc-code>43171802</unspsc-code>
</rdfs:Class>
```

While the underlying approach is not necessarily incorrect, it does not yield a products and services ontology, but a set of cost accounting and purchasing management categories. Quite clearly, we want to make the resulting products and services ontology be useful for many different application areas, including the search for products and services, and not limit the usage to spend analysis.

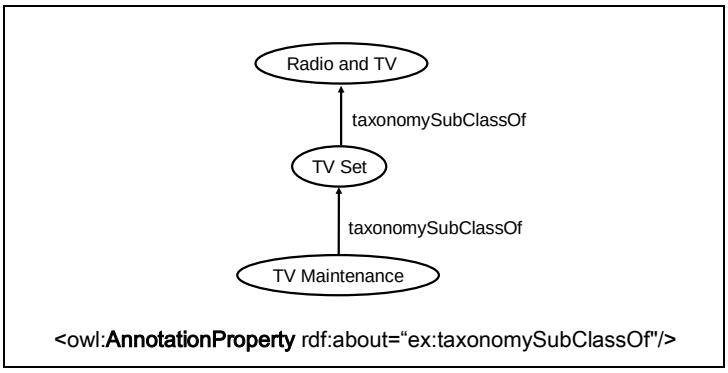

Figure 5: Alternative 2 in OWL: Using the non-transitive Annotation Property taxonomySubClassOf

The second approach is shown in Figure 5. This representation seems to be the most straightforward alternative, since the specific meaning of the taxonomic relationships is captured using a specific property and the classes can represent generic product concepts. The problem with this approach is that, in OWL Lite and OWL DL, a property that connects classes with classes can only be an *Annotation Property*. Thus, it cannot be made a transitive property, and an OWL Lite or OWL DL reasoner will only see explicit statements. In other words, if class A is a taxonomySubClassOf class B and class B is a taxonomySubClassOf C, the reasoner will not infer that class A is also a taxonomySubClassOf class C.

This limitation can be avoided by making the products and services concepts instances instead of classes, as described in solution 3. Then, the property "taxonomySubClassOf" can be an Object Property and can be made transitive (owl:TransitiveProperty). The downside of this approach is that one absolutely needs OWL Lite or OWL DL reasoning support in order to process the transitive nature of the property.

It might seem strange from a pure modeling perspective to impose both the limitations of OWL Lite (e.g. classes cannot be instances) and the limited reasoning of RDF-S (no transitive

properties other than rdfs:subClassOf and rdfs:subPropertyOf). However, this makes sense from a practical perspective, for the following reasons:

1. It works with faster but incomplete implementations of OWL reasoners, especially the use of an RDF-S reasoner on OWL ontologies (supported by some repositories), while being upward compatible to future implementations. Especially, it allows for a merge with other OWL Lite data without making the result of the merge leave the boundaries of OWL Lite.
2. It is possible to create an RDF-S variant of the ontology using the same constructs with only minimal changes. The low performance of current OWL implementations might make this step back necessary. We have to keep in mind that the resulting ontologies are rather big (tens of megabytes).

Thus we wanted to find a solution that does not require reasoning capabilities beyond rdfs:subClassOf.

Since all known representation patterns do not really meet the requirements of creating fully-fledged products and services ontologies based on UNSPSC and eCl@ss, we propose a novel approach, which is based on the idea of representing a concept in the source taxonomy using *two* concepts in the ontology; one generic concept and one taxonomy concept (see also (Hepp, 2005a) and (Hepp, 2005b)). The basic idea is as follows:

1. We create two separate concepts for (1) the *generic* product or service category and (2) the respective *taxonomy* category.
2. We arrange the *taxonomy* concepts in a strict rdfs:subClassOf hierarchy, but not the generic concepts. This allows for capturing the hierarchy of taxonomy concepts without linking the generic concepts to incorrect superordinate classes.

3. In order to ease annotation, we create one annotation class for each taxonomy node which becomes an rdfs:subClassOf of *both* the respective generic and the respective taxonomy concept. With this construct, a single rdf:type statement is sufficient to make a product an instance of both the generic and the taxonomy concept.

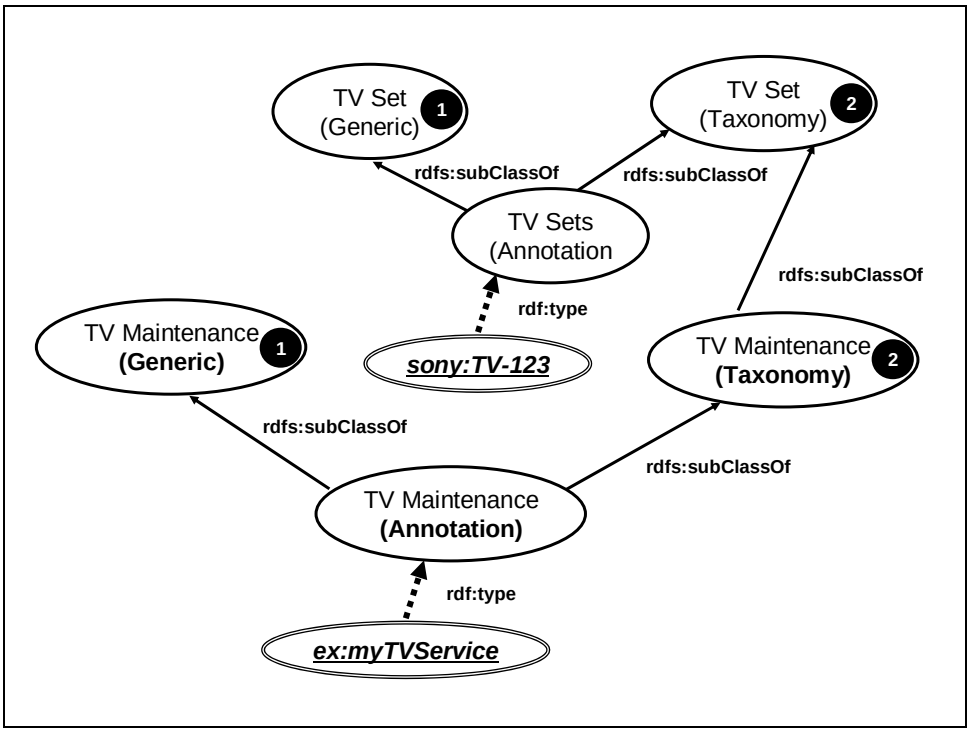

Figure 6: A novel approach: Separating the generic concept from the taxonomic concept

Figure 6 illustrates this approach. The concept numbering with (1) and (2) refers to the numbering in Figure 2. The application of this approach for describing products is shown in Figure 7: The TV maintenance service ex:tv-set-repair is an instance of the annotation class “TV Set Maintenance”. This makes it also an instance of the generic product class “TV Set Maintenance (Generic)” and the taxonomy concept “TV Set Maintenance (Taxonomy)”. The second is a subclass of “TV Set (Taxonomy)”, but the first is not a subclass of “TV Set (Generic)”. This yields exactly the distinction we want: When searching for a TV maintenance service, we look for instances of the *generic* class, and when looking for all items that belong to

the taxonomy category, we use the taxonomy concept. For example, a store manager might want to find all products in the TV set segment. In this case, he or she also wants to find TV set cabling and maintenance, so the query will be based on the taxonomy concept.

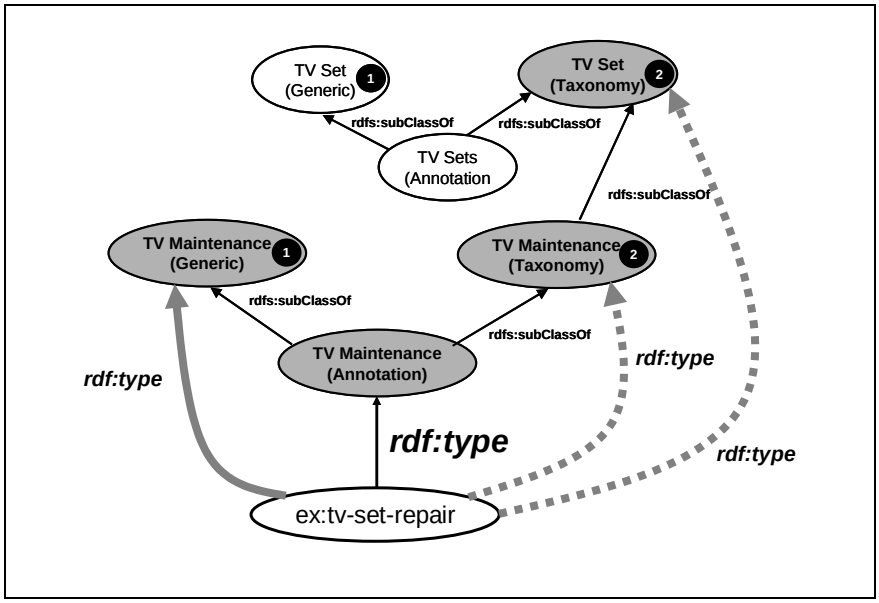

Figure 7: Example of using the proposed approach for product description

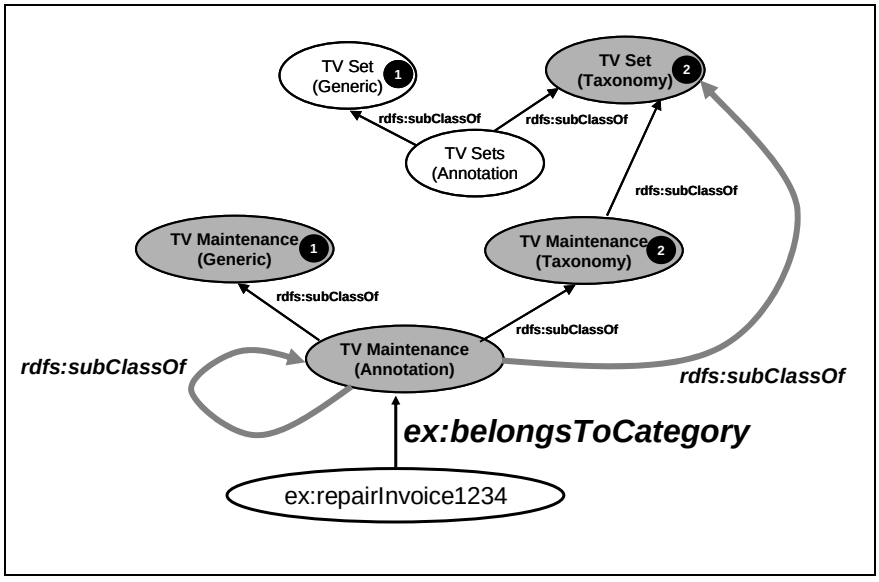

Figure 8: Example of using the proposed approach for annotating incoming invoices for spend analysis

The resulting ontology is not limited to describing product instances, but can also be used for annotating any kind of other entities. For example, incoming invoices can be linked to the respective classes with an annotation property “ex:belongsToCategory” (Figure 8). This allows capturing the fact that a specific invoice is referring to the respective category, but is not an instance. As every class in OWL is also an rdfs:subClassOf of itself and rdfs:subClassOf is transitive, one can easily search for all resources belonging to a taxonomy category on any level of the hierarchy (see Section 4 for the example of an RDQL query).

Product Properties and Property Data Typing

From an ontology engineering perspective, the set of properties in PSCS might seem rather trivial, because most of them are simple data type properties. However, their contribution to machine readable semantics in the Semantic Web is huge, for they provide standardized representations for concepts as generic as “weight” or as specific as “pump capacity”. Some of them can also be applied usefully when the class of a product is not known or no proper class exists. For example, it might be helpful to represent the weight of a novel product even though we do not yet have a proper class for this product.

The import of properties from PSCS into an OWL Lite ontology requires the following steps:

1. Determine whether it is a data type property or an object property, i.e. whether there are enumerated data values for this property or not.
2. Map the data type of the property to one XSD data type supported by current reasoners.
3. Add the unit of measurement (e.g. “inches”), usually stored separately in the PSCS, to the description of the property.
4. Create respective OWL properties for each property contained in the source PSCS.
5. Create instances of a new class “PropertyValue” for each enumerated data value.

It is important to know that the data types used in the PSCS cannot be directly mapped to standard XSD data types. eCl@ss, for example, uses over hundred different data type definitions based on ISO 9735 and ISO 6093, which are far more specific than available standard XSD data types. There are basically two options for this transformation:

1. One can use a rather coarse mapping from the specific ISO 9735 and ISO 6093 data types to standard XSD data types, loosing constraints of the original definition (e.g. field length, number of digits, …).
2. Alternatively, it seems possible to create specific XSD data types or use constraining facets (see (W3C, 2001), section 4.3) to further constrain the standard XSD data types. However, this is currently not supported by OWL, though there in ongoing work in this direction (e.g. (W3C, 2005)).

Thus, and given the yet limited support of reasoning for data types in Semantic Web tools, the first approach seems currently more appropriate. If possible, data types of the source PSCS should be mapped to either xsd:integer, xsd:float, or xsd:string.

Class-specific Property Lists and Value Recommendations

As explained in Section 2, most PSCS recommend the usage of certain properties for specific classes. In eCl@ss, this is a very strong component, as the set of recommended properties tries to reflect industry-wide consensus on which properties are necessary and sufficient to describe a product instance or model. Additionally, there is often a mapping between properties and enumerated values.

Both relations are worth being represented in the target ontology but do not fit into the open world assumption (OWA). It is not possible to restrict the usage of properties on certain classes or to restrict the usage of value instances to certain object properties, since the domain and range

constructs in OWL and RDF-S are used to infer class membership instead of constraining allowed usage.

The most straightforward approach to deal with this problem is creating two annotation properties “recommendedProperty” and “recommendedValue” and use those to capture the respective relation. It is then up to the application developer to use this mapping to validate instance data or to help the user finding suitable properties or values for a given annotation task.

Keywords

Keywords providing alternate search paths to classes, properties, or values can be easily captured using the Dublin Core field “subject” as an annotation property dc:subject.

Persistent URIs as Ontology Component Identifiers

Real-world usage of a products and services ontology requires persistent identifiers for all ontology components. This is achieved best by creating a persistent base URI according to the principles outlined by (Berners-Lee, 1998) and omitting the file name of the actual ontology serialization (e.g. no “eclass.owl” as part of the URI). The later eases having representations of the same ontology in various ontology languages.

Persistent Base URI

A persistent URI should be located in the domain space of the standards body maintaining the PSCS or the ontology. The following describes a generic scheme for such persistent URIs:

`http://organization/ontologies/schema-identifier/version`

organization

The domain name of the organization that manages the respective products and services categorization standard, e.g. “www.eclass.de” or, if that is not feasible or misleading, that of the organization maintaining the ontology variant.

schema-identifier

A uniform short name for the respective products and services categorization standard, e.g. “eclass”. This string must be unique within one organization, but in case a standard starts being maintained by multiple competing organizations (as it happened to the UNSPSC in the past), two organizations might use the same short name.

version

A unique identifier for the version of the underlying standard, preferably in the form yyyymmdd (year-month-day) reflecting the release date of the version, e.g. 20040920 in the absence of a unique version identifier.

As an example, the base URI for eCl@ss version 5.1 could be

```
http://www.eclass.de/ontologies/eclass/5.1
```

Concept Identifiers

The most reasonable approach for the individual concept identifiers is using the primary keys of the original taxonomy components as fragment identifiers (rdf:ID). This is the only way that assures fully automated transformations, because concept names and other fields of the taxonomy cannot be assumed to be unique.

However, the primary keys are in some standards unique only for the same type of component. In eCl@ss, for example, there is a small intersection of the primary key space between classes and properties. Failure to handle this properly would create an inconsistent ontology.

We propose adding a prefix reflecting the respective taxonomy component before the original primary key, e.g. “C_” for classes, “P_” for properties, and “V_” for values. For example, the product category with the primary key “AKJ644002” in eCl@ss 5.1 receives the fragment identifier “C_ AKJ644002”.

As we create *three* OWL classes per each taxonomy class, we have to distinguish them. An intuitive way is adding "-gen" to the identifier of the generic concept, "-tax" to the taxonomy concept, and to use the original identifier for the annotation concept.

As an example, the eCl@ss 5.1 category AKJ644002 will be represented by the following three OWL classes:

```
http://www.eclass.de/ontologies/eclass/5.1#C_AKJ644002
http://www.eclass.de/ontologies/eclass/5.1#C_AKJ644002-gen
http://www.eclass.de/ontologies/eclass/5.1#C_AKJ644002-tax
```

Retrievable Resources for Ontology Components

It can be argued whether the URI of each ontology component should be a retrievable resource by itself or not. We regard having retrievable resources containing a natural language definition of the concepts and probably images or other representations for each ontology component a huge advantage, because this will make it a lot easier for an ontology user to grasp the intension of the concept. As a consequence, it eases communication between the creator of the ontology and ontology users, and will likely contribute to a more consistent usage of the ontology.

We see two options of providing this desirable functionality:

1. The respective organization provides retrievable resources for each URI; this would mainly depend on a proper server configuration.
2. We include an annotation property rdfs:seeAlso for each ontology concept pointing to a retrievable resource where available.

Versioning

As discussed, there is a high versioning dynamics in most PSCS. This makes setting up a proper versioning mechanism a core task of ontology engineering. Without capturing the underlying version of the respective PSCS, the resulting ontology is of very limited use.

Our pragmatic solution to ontology versioning for products and services ontologies that are derived from PSCS is as follows:

1. We treat each new release of the ontology as a new ontology with new concepts, assuming no ex ante links between concepts in the old and new ontology.
2. For classes, properties, and values that are known to be equivalent (e.g. based on an unchanged primary key and/or internal version ID), we provide a (huge) set of explicit "owl:sameAs", “owl:equivalentProperty”, and “owl:equivalentClass” statements.

These explicit statements should be stored in separate ontology modules, so that they can be imported only when needed.

There are two motivations for this approach:

1. One can find many “slight” changes in meaning between the same category in two PSCS versions, especially caused by more precise (narrowing down the concept) or more generic labels (broadening the concept).
2. The notion of identity (e.g. the desired conceptual specificity) might vary between users of the same ontology or between fields of application. When it comes to aggregating invoice items for spend analysis, precision is not so much an issue, whereas for price comparison, we need a high specificity. With our approach, one can import different sets of identity assertions for different purposes.

The underlying rationale is that it seems much more important to not falsely assume identity than not to realize identity, for the later can be bridged later by mediation or established by explicit statements. This problem can be understood a trade-off decision between type 1 (false rejection) and type 2 (false acceptance) errors, or the calibration between precision and recall in Information Retrieval.

4 ecl@ssOWL: Transforming eCl@ss into an OWL Ontology

Based on the methodology described above, we have created an OWL Lite ontology of eCl@ss 5.1 and will do the same continuously for subsequent releases. We have chosen eCl@ss for the demonstration of our approach because it offers both a broad coverage of various industries and an impressive library of useful datatype and object properties. The same approach can be used for UNSPSC and existing taxonomies in other domains.

As eCl@ss is copyrighted material, we are currently preparing the legal framework for a public release of this ontology. Updated information and releases will available at http://www.heppnetz.de/eclassowl/.

Used Modeling Patterns in OWL

Table 2 shows the OWL representation patterns for each of the components of eCl@ss.

One can see how the category “AKK255002” in the original standard translates into three classes in the ontology, one for the generic concept “Agricultural Machine”, one for the taxonomy concept “Agricultural Machine as an eCl@ss Category”, and one class that is a subclass of both and can be used for the annotation of instances. The two properties “label” and “comment” are identical in the example because the English version of eCl@ss does not yet have longer textual definitions for a significant part of its classes. If the definition field in the original standard is empty, we used the label as a substitute.

Table 2: Modeling Patterns for eClassOWL

Ontology Component	Representation in OWL
Classes and Hierarchy	<owl:Class rdf:ID="C_AKK255002-**gen**"> <rdfs:label xml:lang="en">Agricultural machine [generic concept]</rdfs:label> <rdfs:comment xml:lang="en">Agricultural machine [generic concept]</rdfs:comment> </owl:Class> <owl:Class rdf:ID="C_AKK255002-**tax**"> <rdfs:label xml:lang="en">Agricultural machine [taxonomy concept]</rdfs:label> <rdfs:comment xml:lang="en">Agricultural machine [taxonomy concept]</rdfs:comment> <pcs:hierarchyCode>17080000</pcs:hierarchyCode> <rdfs:subClassOf rdf:resource="&pcs;C_AKJ644002-**tax**"/> </owl:Class> <owl:Class rdf:ID="C_AKK255002"> <rdfs:label xml:lang="en">Agricultural machine</rdfs:label> <rdfs:comment xml:lang="en">Agricultural machine</rdfs:comment> <rdfs:subClassOf rdf:resource="&pcs;C_AKK255002-**tax**"/> <rdfs:subClassOf rdf:resource="&pcs;C_AKK255002-**gen**"/> </owl:Class>
Properties	<owl:DatatypeProperty rdf:ID="P_AAA826001"> <rdfs:domain rdf:resource="&owl;Thing"/> <rdfs:range rdf:resource="&xsd;float"/> <rdfs:label xml:lang="en">Input signal range max. (Unit: V)</rdfs:label> <rdfs:comment xml:lang="en">Maximum value of the measured variable with a specified accuracy.</rdfs:comment> </owl:DatatypeProperty>
Values	<pcs:PropertyValue rdf:ID="V_WPB317003"> <rdfs:label xml:lang="en">Highly explosive (Highly explosive)</rdfs:label> <rdfs:comment xml:lang="en">Highly explosive</rdfs:comment> </pcs:PropertyValue>
Keywords	dc:subject for classes and properties
Class-Specific Property Lists	<owl:AnnotationProperty rdf:about="&pcs;recommendedProperty"/> <owl:Class rdf:ID="C_AAA361001"> <pcs:recommendedProperty rdf:resource="&pcs;P_AAA001001"/> <pcs:recommendedProperty rdf:resource="&pcs;P_AAA252001"/> </owl:Class>
Value Recommendations	<owl:AnnotationProperty rdf:about="&pcs;recommendedValue"/> <owl:ObjectProperty rdf:ID="P_AAA008001"> <pcs:recommendedValue rdf:resource="&pcs;V_WAA012001"/> <pcs:recommendedValue rdf:resource="&pcs;V_WAA013001"/> </owl:ObjectProperty>

Experiences

The eCl@ss standard is available at http://www.eclass.de in the form of separate CSV files containing categories, properties, values, class-property recommendations, property-value recommendations, and keywords. As the conversion process requires iterative queries, we first imported those files into a relational database. The actual conversion was then done by a Java application, accessing the database using JDBC.

The application of our methodology to eCl@ss creates only minor problems:

1. The rich data type definitions had to be mapped to the three XSD data types currently supported by reasoners, i.e. xsd:integer, xsd:float, and xsd:string. Generally spoken are the original data type definitions more restrictive. Thus any legal eCl@ss data can be properly represented in the OWL variant. Precision issues could result when OWL data

has to be converted back to fit into the original eCl@ss specification. We mapped all data types of the kind NR1* to xsd:integer, NR2* and NR3* to xsd:float, and all others to xsd:string.

2. The strings, especially labels and definitions, contain special characters, like the ampersand (&), apostrophe (‘), and quotation mark (“), which are not allowed within XML values. To make the RDF/XML serialization of our ontology valid XML, we had to make sure that those characters are translated into proper XML codings (e.g. & for the ampersand). This is not an issue when using proper tools for the XML serialization but must be observed when the XML code is written directly to a file.
3. The labels of enumerated values are often not self-contained. The meaning can sometimes only be grasped by looking at the recommended usage, i.e. for which property they are meant. For example, the value “WPA173003” is labeled “right”, but actually means the “door stop design type” “right”.
4. The resulting ontology is very big: About 25,000 categories in the source taxonomy result in more than 75,000 OWL classes, plus about 5,000 properties. Even though the current English version of eCl@ss does not contain long full text definitions of the classes (see above), the total ontology size in RDF/XML exceeds 25 MB. This does not include the property and value recommendations, which we store in two separate OWL files, because they are not always needed and can be easily imported on demand.

The size of the ontology imposes unexpected problems when trying to use standard ontology editors (e.g. Protégé), repositories/APIs (e.g. Jena 2), or validators (e.g. vowlidator). They all exit with error messages when trying to process the full ontology. It was possible, though, to validate and use a restricted version of the ontology that contains only a small subset of the actual eCl@ss

concepts. Also, during the preparation of the final version of this paper, we were able to load the full ontology plus all property recommendations plus a few instance data samples into an OWLIM[5] configuration.

While we started our work with version 5.0 of eCl@ss, we were able to generate new versions of our ontology based on new releases of eCl@ss in a fully automated fashion. The only manual steps required were importing the new CSV files into our RDBMS and updating the namespace for the new release. The total time for creating the new ontology was less than two hours. Running our script on other standards (e.g. RNTD and UNSPSC) would be basically require only slight modifications in the embedded SQL statements and possibly minor changes in the ontology patterns as shown in Table 2. Generating ontologies in other ontology languages than OWL (e.g. WSML) would just require expressing the ontology patterns from Table 2 in the respective ontology language, which should be easily possible.

Usage in e-Business Scenarios.

The following examples show how the resulting ontology can be used for various E-Business scenarios:

Product Description in the Semantic Web: We assume that “Fendt Supermower“ is an agricultural machine (eCl@ss category AKK255002), its weight is 125.5 kg, and the manufacturer name is "Fendt". Assumed that the ID for this product instance is “machine1”, the respective product description using the eCl@ss ontology would be as follows:

```
<pcs:C_AKK255002 rdf:ID="machine1">
 <pcs:P_BAD875001>125.50</pcs: P_BAD875001>  <!-- Net Weight -->
 <pcs:P_BAA001001>Fendt</pcs:P_BAA001001>  <!-- Manufacturer -->
 <pcs:P_BAA316001>Fendt Supermower1234</pcs:P_BAA316001> <!-- Name -->
</pcs:C_AKK255002>
```

[5] http://www.ontotext.com/owlim/

Now, we want to search for all agricultural machines in the ontology that weigh less than 160 kg. The respective RDQL query would be:

```
SELECT ?x, ?weight, ?productName, ?vendor WHERE
(?x, <rdf:type>, <pcs:C_AKK255002-gen>)
(?x, <pcs:P_BAA001001>, ?vendor)
(?x, <pcs:P_BAA316001>, ?productName)
(?x, <pcs:P_BAD875001>, ?weight)
AND ?weight <160
```

Because we want to get only instances of the generic product category, the class to be used in the query is C_AKK255002**-gen**, not C_AKK255002**-tax**. The later could be used to determine all products that fall in the respective taxonomy category. For example, a store manager might want to see all products in this product segment, including maintenance and spare parts for agricultural machines. Just changing the class ID in the query to C_AKK255002**-tax** would return exactly that.

Annotation of Incoming Invoices for Spend Analysis: As described, the usage of this ontology is not limited to product description. It can also be employed to tag incoming invoices for spend analysis and cost accounting. For this, we need the additional class „IncomingInvoice“ (a concept for paid invoices), the annotation property „costAccountingCategory“, and the data type property „totalInUSD“ for the total in US dollar.

```
<owl:Class rdf:ID="IncomingInvoice"/>
<owl:AnnotationProperty rdf:about="&pcs;costAccountingCategory"/>
<owl:DatatypeProperty rdf:ID="totalInUSD">
 <rdfs:domain rdf:resource="&pcs;IncomingInvoice"/>
 <rdfs:range rdf:resource="&xsd;float"/>
</owl:DatatypeProperty>
```

Then, we can annotate the incoming invoice over $ 1,200 for the above mentioned mower:

```
<pcs:IncomingInvoice rdf:ID="invoice1">
<pcs:totalInUSD>1200.00</pcs:totalInUSD>
<pcs:costAccountingCategory rdf:resource="&pcs;C_AKK255002"/>
</pcs:IncomingInvoice>
```

In order to find all incoming invoices related to the cost accounting category C_AKJ644002-tax (“Machine, device (for special applications)”, which is a superordinate class to C_AKK255002-tax) and its subclasses, we can use the following RDQL query:

```
SELECT ?x, ?total WHERE
(?x, <rdf:type> <pcs:IncomingInvoice>)
(?x, <pcs:costAccountingCategory>, ?y)
(?y, <rdfs:subClassOf>  <pcs:C_AKJ644002-tax>)
(?x, <pcs:totalInUSD>, ?total)
```

Especially the fact that we have a sophisticated set of properties in the ontology allows for rich descriptions of the items, which eases rule-based content integration significantly. For example, we can use *a combination* of (1) taxonomy class information and (2) property ranges to automatically find the proper cost accounting category. A realistic scenario is that we infer the cost accounting ledger from a combination of the supplier name and the products and service category. For example, invoices referring to the category “Services (unclassified)” will be treated as “IT services” if the supplier is “IBM”, and “Building maintenance” if the supplier is “Southwest Carpet Cleaning”.

Accessing Recommended Properties and Property Values: We can also easily determine the recommended properties for a given class or the recommended property value instances for a given property with a simple RDQL query. As per definition, the properties are assigned to the annotation class, and not to the generic or taxonomy concept.

Find recommended properties for C_AKK255002:

```
SELECT ?property WHERE
(<pcs:C_AKK255002>, <pcs:recommendedProperty> ?property)
```

Find recommended values for ObjectProperty P_XYZ001001:

```
SELECT ?value WHERE
(<pcs:P_XYZ001001>, <pcs:recommendedValue> ?value)
```

5 Discussion

We have presented a proposal of how eCl@ss and similar industrial taxonomies for products and services categories can be transformed into OWL ontologies, including dictionaries of properties, enumerative property values, and recommendations that map properties to classes and such that map property values to properties. While previous works, e.g. (Klein, 2002), (McGuinness, 2001), and (Bizer & Wolk, 2003) are rather early prototypes that were mechanically derived from the original standards without a more intensive analysis of the reused standards and their original application domain, we aimed at creating a fully-fledged, industrial strength ontology for the representation of products and services in the Semantic Web. To a certain degree, we were surprised how little previous attempts of serious ontology engineering for products and services exists, for we think that comprehensive ontologies in this domain are a prerequisite for almost any serious E-Business application of Semantic Web technology.

Our proposal comes not without cost. First, the resulting ontology provides quite limited reasoning support. Second, the resulting ontology serializations are very large and exceed the ontology size that most current ontology infrastructure can handle. In the following, we discuss these two aspects.

Limited Amount of Reasoning support

Readers from the traditional ontology engineering field might criticize the shallowness of the generic concepts. It is true that the generic products and services concepts do not support much reasoning, for they are just named classes. However, the semantic richness needed for most business scenarios will come from the usage of the huge collection of properties. An example is a parametric search like “find all TV sets made by Siemens with a screen size between 10 and 15 inches and 12 Volt power-supply”. Of course, we agree that a greater amount of e.g. axioms

would be generally desirable. On the other hand, we see no simple way of automatically adding these axioms, because they cannot be easily derived from the input standard. Additionally, we will have to put the resources necessary for the respective axiomatic enrichment in relation to the gain in automation and the resulting economies. In a domain as dynamic as products and services it should not be taken for granted that the business benefit will always outweigh the cost of the creation of the respective ontologies, nor that we are able to yield axiomatic richness during the lifespan of the concept, i.e. as long as the category is still relevant. Also, it has recently been pointed out by (Gruber, 2005) that such "semiformal" ontologies are practically very relevant.

Size

The resulting ontologies are very big, with file sizes of more than 25 MB. This causes a problem with most current OWL tools and repositories, resulting in memory errors or very slow processing. The Jena framework[6] and many Jena-based tools (e.g. vowlidator[7] and Protégé 3.0[8]) do not load an ontology of this size, but exit with an out-of-memory exception. This might be resolvable by changing the memory allocation for the JVM. However, the underlying reason seems to be that all popular DL reasoners require an in-memory model of the whole ontology, which does not work well with such big ontologies; and that OWL Lite as the simplest language variant of OWL still requires DL-based reasoner support (see (Fensel, 2004), p. 45).

There are approaches towards more scalable OWL reasoning (Bechhofer, Horrocks, & Turi, 2005) based on a hybrid reasoner/database architecture; however, the described architecture aims

[6] http://jena.sourceforge.net/

[7] http://owl.bbn.com/validator/

[8] http://protege.stanford.edu/

at improving the scalability in terms of instance data, while our ontologies are big already at the "TBox" level.

The advent of more suitable ontology languages, especially reductions of OWL (see e.g. (Bruijn, Lara, Polleres, & Fensel, 2005) or (Kiryakov, Ognyanov, & Manov, 2005)) and WSML (de Bruijn et al., 2005) as a family of languages that allows a trade-off decision between expressivity of the language and reasoning costs might improve the situation.

The proposed methodology creates three ontology classes per each node in the original taxonomy and thus directly contributes to large ontologies. This is obviously a disadvantage and there can be cases where the original taxonomy is already so big that tripling the number of ontology classes is rendered unfeasible. However, this weakness of the proposed approach has to be judged in the proper context. First of all, the size of the "schema part" (or TBox, if you want) will, in many cases, be significantly smaller than the instance data part. Thus, the increased number of ontology concepts will have a much lesser overall impact when compared to the total size of the ontology. Second, it is possible to partition the resulting ontology into to modules, one module containing the generic and the other module containing the taxonomy part and thus avoid the conceptual overhead in scenarios that do not require both views. In some cases, the hierarchy might be so inconsistent that it is not justified to take the effort of representing it. Then, it is recommendable to just derive named classes without a hierarchy in the ontology. Third, the proposal requires no reasoning support beyond rdfs:subClassOf and will thus work with faster, limited reasoners. Even an RDF-S reasoner would be sufficient, which means that the unwanted ontology growth has to be set in relation to the very much reduced reasoning costs.

A very advantageous property of the proposed methodology is that it can be applied mechanically without human intervention. From a business perspective, the unwanted ontology

growth has to be judged in comparison to the gain in automation of the ontology building process.

In general we think that, at least in the E-Business domain, the size of the resulting ontologies is not a serious argument against their usefulness. For three reasons, we assume that quite the opposite is true: First, we must stress that ontologies of this size will rather be the lower limit of what we will have to expect for non-toy ontologies in E-Business scenarios. Using traditional relational databases and COTS application software, even small companies are dealing with such amounts of data. Second, we conclude from the experienced limitations with current tools not that the ontology is too big, but that many prototypes yielded by the Semantic Web community have so far not achieved scalability beyond toy applications. It should be noted, though, that the situation has improved a lot since our first experiments (in April 2004) and November 2005. As of November 7, 2005 we were able to load the full ontology plus all property recommendations plus a few instance data samples into an OWLIM configuration. Query times for the sample queries described in section 4 remained below 100 ms.

Third, we expect our ontology to become a widely used benchmark among the developers of repositories and tools, and will thus hopefully contribute to a quick gain in scalability of Semantic Web infrastructure.

Implications

Our work presented in this paper has the following theoretical and practical implications:

Theoretical: We contributed to a better understanding of the reuse of taxonomies for the creation of ontologies. This is relevant, since existing taxonomies are likely the most valuable asset for reuse that is available for building ontologies. Especially, we described how the interpretation of the taxonomic relation is an important modelling decision.

Practical: Our work can be used as a large-scale benchmark for ontology infrastructure, especially repositories and editing and browsing components. Also, the resulting ontology eClassOWL can be used to build real business applications for E-Procurement, spend analysis, or catalog data integration and thus help demonstrate the business benefits of Semantic Web technology.

6 Conclusion

In this paper, we introduced a methodology for deriving fully-fledged OWL Lite ontologies from products and services categorization standards. We showed how the intension of the taxonomy concepts is directly influenced by the interpretation of the taxonomic relationship, and how the creation of a set of three separate ontology classes representing (1) the generic concept, (2) the taxonomy concept, and (3) an annotation concept for each taxonomy category yields far more useful ontologies than one single ontology class per taxonomy category, especially if the later is in combination with rigid mapping of the taxonomy relationship to rdfs:subClassOf, as currently in use. Additionally, we developed an approach of how property dictionaries, enumerated values, class-property recommendations, and property-value recommendations can be captured in an OWL ontology while staying well within the limitations of OWL Lite.

We then successfully applied this methodology to the comprehensive categorization standard eCl@ss and yielded a fully-fledged ontology for the product and services domain, reflecting more than 25,000 product types in 75,000 ontology classes plus a collection of more than 5,000 sophisticated properties. The resulting OWL version of eCl@ss 5.1 is currently the most comprehensive, horizontal products and services ontology that we know of, leaves the current "toy" level of many previous other ontologies behind, and is fully usable for product description

and spend analysis item tagging, which we regard as two very important usages of Semantic Web technology in business applications.

Acknowledgements: I would like to thank Jos de Bruijn, Axel Polleres, Amit Sheth, and the anonymous reviewers for very valuable comments on previous versions of this paper.

References

Bechhofer, S., Horrocks, I., & Turi, D. (2005, July 22-27). *The OWL Instance Store: System Description.* Paper presented at the 20th International Conference on Automated Deduction (CADE-20), Tallinn, Estonia.

Berners-Lee, T. (1998). *Cool URIs don't change*. Retrieved November 8, 2004, from http://www.w3.org/Provider/Style/URI.html

Bizer, C., & Wolk, J. (2003). *RDF Version of the eClass 4.1 Product Classification Schema*. Retrieved August 16, 2005, from http:////www.wiwiss.fu-berlin.de/suhl/bizer/ecommerce/eClass-4.1.rdf

Brachman, R. J. (1983). What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. *IEEE Computer*, *16*(10), 30-36.

Bruijn, J. d., Lara, R., Polleres, A., & Fensel, D. (2005, May 10-14). *OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web.* Paper presented at the 14th International World Wide Web Conference (WWW2005), Chiba, Japan.

de Bruijn, J. (2003). Using Ontologies. Enabling Knowledge Sharing and Reuse on the Semantic Web. *DERI Technical Report DERI-2003-10-29, October 2003*, 1-49.

de Bruijn, J., Lausen, H., Krummenacher, R., Polleres, A., Predoiu, L., Kifer, M., et al. (2005). *D16.1v0.21 The Web Service Modeling Language WSML. WSML Final Draft*. Retrieved November 7, 2005, from http://www.wsmo.org/TR/d16/d16.1/v0.21/.

Fairchild, A. M., & de Vuyst, B. (2002). *Coding Standards Benefiting Product and Service Information in E-Commerce.* Paper presented at the 35th Annual Hawaii International Conference on System Sciences (HICSS-35).

Fensel, D. (2004). *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce* (2nd ed.). Berlin etc.: Springer.

Fensel, D., Ding, Y., Omelayenko, B., Schulten, E., Botquin, G., Brown, M., et al. (2001). Product Data Integration in B2B E-Commerce. *IEEE Intelligent Systems*, *16*(4), 54-59.

Fensel, D., McGuinness, D. L., Schulten, E., Ng, W. K., Lim, E.-P., & Yan, G. (2001). Ontologies and Electronic Commerce. *IEEE Intelligent Systems*, *16*(1), 8-14.

Fernández-López, M., & Gómez-Pérez, A. (2002). Overview and analysis of methodologies for building ontologies. *The Knowledge Engineering Review*, *17*(2), 129-156.

Giunchiglia, F., Marchese, M., & Zaihrayeu, I. (2005, July 9-10, 2005). *Towards a Theory of Formal Classification.* Paper presented at the AAAI-05 Workshop on Contexts and Ontologies: Theory, Practice and Applications (C&O-2005), Pittsburgh, Pennsylvania, USA.

Gruber, T. (2005). Every ontology is a treaty - a social agreement - among people with some common motive in sharing. *AIS SIGSEMIS Bulletin, 1*(3), 4-8.

Hepp, M. (2004). Measuring the Quality of Descriptive Languages for Products and Services. In F.-D. Dorloff, J. Leukel & V. Schmitz (Eds.), *E-Business - Standardisierung und Integration. Tagungsband zur Multikonferenz Wirtschaftsinformatik 2004* (pp. 157-168). Göttingen: Cuvillier.

Hepp, M. (2005a, May 26-28). *A Methodology for Deriving OWL Ontologies from Products and Services Categorization Standards.* Paper presented at the 13th European Conference on Information Systems (ECIS2005), Regensburg, Germany.

Hepp, M. (2005b, November 7). *Representing the Hierarchy of Industrial Taxonomies in OWL: The gen/tax Approach.* Paper presented at the ISWC Workshop Semantic Web Case Studies and Best Practices for eBusiness (SWCASE05), Galway, Irland.

Hepp, M., Leukel, J., & Schmitz, V. (2005a, March 29 - April 1, 2005). *Content Metrics for Products and Services Categorization Standards.* Paper presented at the IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-05), Hong Kong.

Hepp, M., Leukel, J., & Schmitz, V. (2005b, October 18-20). *A Quantitative Analysis of eCl@ss, UNSPSC, eOTD, and RNTD Content, Coverage, and Maintenance.* Paper presented at the IEEE ICEBE 2005, Beijing, China.

Kim, D., Kim, J., & Lee, S.-g. (2002, February 24-25). *Catalog Integration for Electronic Commerce through Category-Hierarchy Merging Technique.* Paper presented at the 12th International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems (RIDE'02), San Jose, CA, USA.

Kim, D., Lee, S.-g., Chun, J., & Lee, J. (2004, July 6-9). *A Semantic Classification Model for e-Catalogs.* Paper presented at the IEEE Conference on E-Commerce Technology (CEC'04), San Diego, CA, USA.

Kiryakov, A., Ognyanov, D., & Manov, D. (2005, November 20). *OWLIM – a Pragmatic Semantic Repository for OWL.* Paper presented at the International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005), New York City, USA.

Klein, M. (2001). *Schema Definition for the Representation of the UNSPSC Classification in RDF-Schema*. Retrieved October 7, 2005, from http://139.91.183.30:9090/RDF/VRP/Examples/unspsc1.rdf

Klein, M. (2002). *DAML+OIL and RDF Schema representation of UNSPSC*. Retrieved April 23, 2004, from http://www.cs.vu.nl/~mcaklein/unspsc/

McGuinness, D. L. (2001). *UNSPSC Ontology in DAML+OIL*. Retrieved November 5, 2004, from http://www.ksl.stanford.edu/projects/DAML/UNSPSC.daml

Ng, W. K., Yan, G., & Lim, E.-P. (2000). Heterogeneous Product Description in Electronic Commerce. *ACM SIGEcom Exchanges, 1*(1), 7-13.

Obrst, L., Wray, R. E., & Liu, H. (2001, October 17-19). *Ontological Engineering for B2B E-Commerce.* Paper presented at the International Conference on Formal Ontology in Information Systems (FOIS'01), Ogunquit, Maine, USA.

Paslaru Bontas, E. (2005, June 13-14). *Using Context Information to Improve Ontology Reuse.* Paper presented at the Doctoral Consortium at the 17th Conference on Advanced Information Systems Engineering (CAiSE'05), Porto, Portugal.

Schulten, E., Akkermans, H., Botquin, G., Dörr, M., Guarino, N., Lopes, N., et al. (2001). The E-Commerce Product Classification Challenge. *IEEE Intelligent Systems, 16*(4), 86-89.

U.S. Census Bureau. (2004). *North American Industry Classification System (NAICS)*. Retrieved November 5, 2004, from http://www.census.gov/epcd/www/naics.html

van Assem, M., Menken, M. R., Schreiber, G., Wielemaker, J., & Wielinga, B. J. (2004, November 7-11). *A Method for Converting Thesauri to RDF/OWL*. Paper presented at the ISWC'04, Hiroshima, Japan.

W3C. (2001). *XML Schema Part 2: Datatypes Second Edition. W3C Recommendation 28 October 2004*. Retrieved November 15, 2004, from http://www.w3.org/TR/xmlschema-2/

W3C. (2005). *XML Schema Datatypes in RDF and OWL. W3C Working Draft 27 April 2005*. Retrieved Nov. 7, 2005, from http://www.w3.org/TR/swbp-xsch-datatypes/

Wielinga, B. J., Schreiber, A. T., & Sandberg, J. A. C. (2001, October 21-23). *From Thesaurus to Ontology*. Paper presented at the First International Conference on Knowledge Capture (K-CAP 2001), Victoria, British Columbia, Canada.

Wielinga, B. J., Wielemaker, J., Schreiber, G., & van Assem, M. (2004, May 10-12). *Methods for Porting Resources to the Semantic Web.* Paper presented at the Proceedings of the First European Semantic Web Symposium (ESWS'04), Heraklion, Greece.

Appendix

Modeling Patterns used in OWL

Ontology Component	Representation in OWL
Classes and Hierarchy	<owl:Class rdf:ID="C_AKK255002-**gen**"> <rdfs:label xml:lang="en">Agricultural machine [generic concept]</rdfs:label> <rdfs:comment xml:lang="en">Agricultural machine [generic concept]</rdfs:comment> </owl:Class> <owl:Class rdf:ID="C_AKK255002-**tax**"> <rdfs:label xml:lang="en">Agricultural machine [taxonomy concept]</rdfs:label> <rdfs:comment xml:lang="en">Agricultural machine [taxonomy concept]</rdfs:comment> <pcs:hierarchyCode>17080000</pcs:hierarchyCode> <rdfs:subClassOf rdf:resource="&pcs;C_AKJ644002-**tax**"/> </owl:Class> <owl:Class rdf:ID="C_AKK255002"> <rdfs:label xml:lang="en">Agricultural machine</rdfs:label> <rdfs:comment xml:lang="en">Agricultural machine</rdfs:comment> <rdfs:subClassOf rdf:resource="&pcs;C_AKK255002-**tax**"/> <rdfs:subClassOf rdf:resource="&pcs;C_AKK255002-**gen**"/> </owl:Class>
Properties	<owl:DatatypeProperty rdf:ID="P_AAA826001"> <rdfs:domain rdf:resource="&owl;Thing"/> <rdfs:range rdf:resource="&xsd;float"/> <rdfs:label xml:lang="en">Input signal range max. (Unit: V)</rdfs:label> <rdfs:comment xml:lang="en">Maximum value of the measured variable with a specified accuracy.</rdfs:comment> </owl:DatatypeProperty>
Values	<pcs:PropertyValue rdf:ID="V_WPB317003"> <rdfs:label xml:lang="en">Highly explosive (Highly explosive)</rdfs:label> <rdfs:comment xml:lang="en">Highly explosive</rdfs:comment> </pcs:PropertyValue>
Keywords	dc:subject for classes and properties
Class-Specific Property Lists	<owl:AnnotationProperty rdf:about="&pcs;recommendedProperty"/> <owl:Class rdf:ID="C_AAA361001"> <pcs:recommendedProperty rdf:resource="&pcs;P_AAA001001"/> <pcs:recommendedProperty rdf:resource="&pcs;P_AAA252001"/> </owl:Class>
Value Recommendations	<owl:AnnotationProperty rdf:about="&pcs;recommendedValue"/> <owl:ObjectProperty rdf:ID="P_AAA008001"> <pcs:recommendedValue rdf:resource="&pcs;V_WAA012001"/> <pcs:recommendedValue rdf:resource="&pcs;V_WAA013001"/> </owl:ObjectProperty>