GenTax: A Generic Methodology for Deriving OWL and RDF-S Ontologies from Hierarchical Classifications, Thesauri, and Inconsistent Taxonomies

Martin Hepp, Jos de Bruijn

Digital Enterprise Research Institute (DERI), University of Innsbruck
mhepp@computer.org, jos.debruijn@deri.org

Abstract. Hierarchical classifications, thesauri, and informal taxonomies are likely the most valuable input for creating, at reasonable cost, non-toy ontologies in many domains. They contain, readily available, a wealth of category definitions plus a hierarchy, and they reflect some degree of community consensus. However, their transformation into useful ontologies is not as straightforward as it appears. In this paper, we show that (1) it often depends on the context of usage whether an informal hierarchical categorization schema is a classification, a thesaurus, or a taxonomy, and (2) present a novel methodology for automatically deriving consistent RDF-S and OWL ontologies from such schemas. Finally, we (3) demonstrate the usefulness of this approach by transforming the two e-business categorization standards eCl@ss and UNSPSC into ontologies that overcome the limitations of earlier prototypes. Our approach allows for the script-based creation of meaningful ontology classes for a particular context while preserving the original hierarchy, even if the latter is not a real subsumption hierarchy in this particular context. Human intervention in the transformation is limited to checking some conceptual properties and identifying frequent anomalies, and the only input required is an informal categorization plus a notion of the target context. In particular, the approach does not require instance data, as ontology learning approaches would usually do.

Keywords. Ontology engineering, ontology learning, OWL, RDF-S, reuse, taxonomies, thesauri, classifications, UNSPSC, eCl@ss, e-business

1. Introduction

Hierarchical classification standards, thesauri, and such taxonomies that were not initially designed to be used as ontologies exist in many domains. They are likely the most promising sources for the creation of domain ontologies at reasonable costs, because they reflect some degree of community consensus and contain, readily available, a wealth of category definitions plus a hierarchy. For instance, UNSPSC [1], a standard categorization for products and services and often referred to as a products and services ontology for e-business, contains 20,789 categories (in version 7,0901) and the similar but more expressive industrial standard eCl@ss [2] defines 25,658 categories plus 5,525 precisely specified object and datatype properties (in version 5.1de). The products classification in eBay, as an additional example, includes more than 2,000 categories for computer and networking equipment alone.

For a quantitative analysis of the content and domain coverage of such standards in the products and services domain, see [3].

While it is tempting to write simple scripts that mechanically create ontology classes for the categories in the source standard and `rdfs:subclassOf` relations for the edges constituting the hierarchy, as has been done by [4] and [5], this straightforward approach often yields ontologies that are of limited practical use, since it implies a particular interpretation of the categories so that the original hierarchical order is a valid subsumption hierarchy. If, for example, "ice" is a subcategory of "beverages" in the original hierarchy, this naïve transformation forces us to read the category "beverages" as something like "beverages and related stuff from a purchasing manager's perspective", because only then holds that all instances of the former class are also instances of the latter. While this choice is a valid transformation, it does often not yield the most useful ontologies, as has been shown in [6] and [7]. In particular, the ontology classes would not be sufficiently narrow to describe actual products or services instances in an unambiguous way.

The main cause for this problem is that, due to the informal nature of the original schemas, the meanings of (1) the categories, (2) the hierarchical relations between them, and (3) the task of assigning an instance to a category are usually blurry, and the meanings of the three components are not clearly separated from each other. This means that such informal specifications entail multiple possible ontologies. For example, we can interpret the categories in a way so that the original hierarchy forms a consistent subsumption hierarchy and can be represented using `rdfs:subClassOf`, or we can interpret the categories in another way but must then use another transitive, binary relation of the kind "A is a subcategory of B in some context" in order to capture the hierarchy [6, 7].

Since most categorizations were not created under rigorous knowledge engineering methodologies, they often suffer from additional conceptual anomalies, e.g. local names or a varying semantics of the hierarchy relation by depth of branching. Such anomalies are sometimes found in only relatively small parts of the categorization schema. They may thus not become apparent by a quick view on a part of the specification.

Besides these difficulties in understanding the original semantics and selecting a useful interpretation for a given application, we are additionally constrained by the expressiveness of popular ontology formalisms. OWL DL, for example, does not allow the definition of transitive relations between ontology classes, which may force us to invent suitable ontology modeling patterns as workarounds.

Finally, it is highly desirable that the generation of derived ontologies is automated as much as possible, because of the high number of categories.

1.1. Classification, Thesaurus, and Taxonomy

The terms thesaurus and taxonomy are well established in ontology research. Basically, a *thesaurus* is a collection of concepts that are augmented by three types of relations: "broader term" (BT) and "narrower term" (NT), which may be read as a hierarchical order, and "related term" (RT), which is used to capture conceptual proximity [cf. e.g. 8]. An important characteristic of the NT/BT relation is that it is semantically less specific than a *subclassOf* relation used in ontology engineering for

building a subsumption hierarchy, since an instance that fits one subcategory of a thesaurus needs not to be an instance of a the respective parent category. For example, “ice cubes” may be a narrower term to “beverages”, but instances of the category “ice cube” are not instances of the category “beverage” if we read the categories literally.

A *taxonomy* is different from a thesaurus in that it contains a subsumption hierarchy in the form of transitive *subclassOf* relations, i.e. each instance of a class can be assumed to be also an instance of all parent categories. It should be noted that hierarchical classifications are sometimes imprecisely referred to as taxonomies even though they do not include a real subsumption hierarchy.

Classifications are sets of concepts and have been used for ages as a means of grouping entities by similarity. It is important to stress that the initial purpose of classification was not to capture the essence of things, i.e. modeling a part of the world, as in ontology engineering, but aggregating entities for some arbitrary purpose. Also, a hierarchical order is a frequent but not a mandatory property of classifications. Sometimes, classifications are assumed to be limited to hierarchical classifications, which are rooted trees where the semantics of the edges may vary widely depending on the purpose and context of usage [cf. 9].

In this paper, we will subsume all three types, i.e. taxonomies, thesauri, and hierarchical classifications under the term *hierarchical categorization schema*, which all have in common that they include a set of categories and some form of a hierarchical order. There are two main reasons for this unified view on the three variants. First, it may depend of the context of usage whether a given collection is a taxonomy, a thesaurus, or just a hierarchical classification. Second, we want to provide an approach that can be directly applied to all three types, thus allowing us to reuse the wealth of any such schemas for building domain ontologies, which are urgently needed for making the Semantic Web a reality.

1.2. Our Contribution

In this paper, we (1) show that it often depends on the context of usage whether an informal, hierarchical categorization schema is a classification, a thesaurus, or a taxonomy, (2) develop a novel methodology for mechanically deriving consistent, lightweight ontologies for a particular context from hierarchical classifications, thesauri, or taxonomies, even if they contain typical conceptual anomalies, (3) present suitable modeling patterns for RDF-S and OWL-DLP that require no reasoning support beyond `rdfs:subClassOf`, which allows for the use of the resulting ontologies with lightweight, scalable reasoners and repositories like OWLIM [10], and (4) demonstrate the usefulness of our approach by transforming the e-business categorization standards eCl@ss [2] and UNSPSC [1] into fully-fledged ontologies.

We also propose to use deductive statistics for the diagnosis of common anomalies and for selecting modeling options when handling large categorization schemas. This allows us to quantify the likelihood that the resulting ontology is consistent without the need to evaluate the complete schema manually.

The structure of the paper is as follows: In section 2, we present a unified model for hierarchical classifications schemas, thesauri, and taxonomies, which takes into account the role of contexts. In section 3, we present our methodology for deriving ontologies from hierarchical categorization schemas. In section 4, we show how our

approach can be successfully applied to the representation of eCl@ss and UNSPSC in RDF-S and OWL. In section 5, we discuss our findings and compare them to related works.

2. A Uniform Model of Classifications, Thesauri, and Taxonomies

In this section, we will present a unified formal model that fits any kind of hierarchical categorization schema, be it a domain classification, a thesaurus, or a taxonomy.

2.1. Overview

When taking the categories found in a hierarchical categorization schema as the basis for the creation of an ontology, we face two fundamental problems: First, the meaning of the categories may vary by context. With context we mean in here a domain of usage over which a category label is interpreted. Second, unless there is a formal definition of the semantics of the arcs constituting the hierarchy, the meaning of the category concepts is not determined independently of the meaning of that hierarchy relationship, i.e. both are tangled. For example, a category labeled "TV Set" can, depending on the context of usage, mean very different things, e.g. (1) any entity that is an actual TV set, (2) all TV sets and somewhat related items, (3) all invoices and cost statements that are related to TV sets, or (4) anything that can in any context be regarded as related to TV sets.

In the original fields of usage, this blurriness constitutes no serious problems, since one usually never expresses that an entity is *an instance* of a particular category, but rather *assigns entities to categories in well-defined contexts*. Thus, incompatible meanings of the categories do usually not become apparent. Since a relation like `rdf:type` is never used, it is no problem that in catalog data exchange contexts, actual TV set makes and models are assigned to the UNSPSC category "TV Set", while for spend analysis, invoices reflecting TV set and TV cabling purchases are put into the same category.

One could easily be tempted to trace back these problems to a lack of understanding of the original context and assume that there was one correct interpretation of the semantics of the categories. However, this is not the case, since we can observe that the very same categorization schemas are used in very different contexts with varying interpretations. When we want to build useful ontologies, however, we need to be clear about the semantics of the resulting ontology classes, i.e. what it means to be an instance of this very class.

Two examples might further illustrate this fundamental 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 "ice" a subcategory of "non-alcoholic beverages" in UNSPSC and "docking stations" a subcategory of "computers" in eCl@ss. Now, there exists at least one context in which the hierarchy relation can be read as a taxonomic relation in the sense of "`rdfs:subClassOf`", 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” from e.g. 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 negative consequence of interpreting the hierarchy as being equivalent to `rdfs:subClassOf`, is obvious: We can no longer use the resulting classes e.g. for 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. One could argue that exactly this narrow definition of the classes is the original semantics of the categories. However, this is not true, since the plain text descriptions for these classes in UNSPSC and eCl@ss define the categories in the generic sense.

In a nutshell, most hierarchical categorization schemas are used with varying semantics in multiple contexts, and depending on the respective context, the hierarchy relations may constitute a subsumption hierarchy or just “narrower then/broader then” relations. Our claim is that by restricting the interpretation of a categorization scheme to a particular context, we can derive more useful ontologies, even if that means that the hierarchical order of the original schema does not constitute a subsumption hierarchy in this particular context.

2.2. Formal Definition

We view a hierarchical categorization schema as a directed graph where nodes represent categories and edges represents the “narrower term” or “has subcategory” relation. Depending on the context, a set is related to each category. This set represents the items associated with the category in a particular context.

Formally, a hierarchical categorization schema *S* is a 6-tuple $S = \langle V, E, C, J, l^V, l^C \rangle$ with:

- *V* a set of categories,
- *E* a binary relation over *V:* $E : V \times V$ reflecting the original edges in the hierarchy,
- *C* a set of contexts,
- *J* a partial function which assigns to every context $c \in C$ a partial function which assigns to every category $v \in V$ a set of items such that *J(c)(v)* is the set of items associated with category *v* in context *c*,
- l^V a function which associates labels with categories: $l^V : V \rightarrow string$, and
- l^C a function which associates labels with contexts: $l^C : C \rightarrow string$.

We can see from the definition that a category is interpreted differently depending on the context of usage. We say that the *interpretation of a category* $v \in V$ *in a context* $c \in C$, denoted $S^c(v)$, is the set $S^c(v) = J(c)(v)$. The *interpretation of a*

category $v \in V$, denoted *S(v)* is the union of the interpretations of *v* at every context in *C*: $S(v) = \cup \{S^c(v) \mid c \in C\}$.

Using the formal model, we can specify a number of properties which a categorization schema may have. First of all, it is not clear whether a particular hierarchical classification is a consistent taxonomy or rather a thesaurus.

We would call a classification *S* a *taxonomy*, if the hierarchy is a valid subsumption hierarchy, i.e., for all pairs of concepts v_a, v_b holds that if v_a is a descendant of v_b then v_a is also a *subclass* of v_b. Formally, *S* is a taxonomy if, and only if, for all $v_a, v_b \in V$ holds:

$$if\ \langle v_b, v_a \rangle \in E\ then\ S(v_a) \subseteq S(v_b).$$

We call *S* a *taxonomy with respect to context c* if the interpretations of all categories in context *c* form a valid subsumption hierarchy. Formally, *S* is a taxonomy with respect to a context $c \in C$ if, and only if, for all $v_a, v_b \in V$ holds:

$$if\ \langle v_b, v_a \rangle \in E\ then\ S^c(v_a) \subseteq S^c(v_b)$$

Several of the hierarchical classification schemas we looked at are taxonomies only for some contexts.

We say a categorization is *cyclic* if there is a $v \in V$ such that $\langle v,v \rangle \in tr(E)$ with *tr(E)* the transitive closure of *E*. For the remainder of this paper, we assume the input categorization not to be cyclic.

3. Deriving OWL and RDF-S Ontologies from Hierarchical Categorization Schemas

In this section, we describe a novel approach of deriving consistent OWL and RDF-S ontologies from hierarchical categorization schemas. Our approach allows for the semi-automatic creation of meaningful ontology classes for a particular context while preserving the original hierarchy, even if the latter is not a consistent subsumption hierarchy in this particular context. The basic idea of our *GenTax* methodology is to derive two ontology classes from each category: one **gen**eric concept in a given context and one broader **tax**onomic concept which allows preserving the original hierarchy.

The input required is minimal and limited to (1) an informal specification of a hierarchical categorization schema as defined in section 2.2 and (2) a notion of the context in which the ontology should be used. In particular, we do not need instance data or any additional information, as most ontology learning approaches usually would. The transformation itself is semi-automatic in the sense that human intervention is limited to checking some conceptual properties and identifying frequent anomalies. In other words, the actual generation of the ontology can be done by a script that only needs to be configured properly by a human.

The resulting ontologies can be either RDF-S or OWL DLP; in fact, they require no reasoning support beyond `rdfs:subClassOf`, which allows for the use of lightweight, scalable reasoners, while still being able to merge an OWL DLP variant

with OWL DL data without leaving the boundaries of OWL DL (which would be the case if e.g. RDF-S meta-modeling was used).

3.1 Overview

The basic idea of our approach is as follows:

- We derive meaningful, generic ontology classes from the categories in the original classification by narrowing them down to their meaning in one particular context.
- We define taxonomic concepts for the categories of the original schema so that they form a consistent taxonomy when the edges in the schema are interpreted as a subsumption hierarchy.

Our algorithm depends on a number of external functions.

- The function **genS** takes as input a categorization hierarchy and returns a formal specification, as defined in Section 2.2. This function takes care of all the pre-processing, disambiguating labels of categories, etc.
- The function **getContextInfo** takes as input a formal categorization and returns a formal categorization which includes an interpretation for every category at every context, except for possibly c_{cat}.
- The function **genURI** takes as argument a context label and a category label and returns a URI based on this information.

The input to the algorithm is some categorization and a set of contexts *C*. The output of the algorithm is an RDF-S or OWL DLP ontology. There is a special context c_{cat} with $l^C(c_{cat})$ = *"Category"*. If this context is included in *C*, then the algorithm will create special category classes in the output ontology for each of the categories in the categorization.

Step 1: Pre-processing and creating a formal representation of the model

The input to this step is an arbitrary hierarchical categorization schema *H*. The output is $S = \langle V,E,C,J,l^C,l^V \rangle$ with V the set of categories, *E* such that $\langle v_1,v_2 \rangle \in E$ if there is an arc $\langle v_1,v_2 \rangle$ in the original categorization, *C* the set of input contexts, and *J* is not defined for any context. S is obtained as follows: *S* = **genS**(*H*).

Step 2: Deriving context information

This step defines the function *J* in *S* for each context $c \in C$ with $c \neq c_{cat}$. The (external) algorithm finds an interpretation $S^c(v)$ for each category $v \in V$. The output is *S'* = **genContextInfo**(*S*), where *J(c)(v)* is defined for every $c \in C, v \in V$ such that $c \neq c_{cat}$.

Step 3: Category context

If $c_{cat} \notin C$, proceed to the next step. Otherwise, choose $S^{c_{cat}}$ as follows: (a) for any $v \in V$, $S^{c_{cat}}(v) \supseteq S(v)$, (b) for all $v_{1,} v_2 \in V$ such that $\langle v_{1,} v_2 \rangle \in E$, $S^{c_{cat}}(v_1) \subseteq S^{c_{cat}}(v_2)$, and for every $v \in V$, $S^{c_{cat}}(v)$ is the smallest set such that the conditions (a) and (b) hold. Obviously there is such a $S^{c_{cat}}$.

Step 4: Generating the ontology

Start with an empty set G.

Step 4.1: Generating ontology classes

For each category $v \in V$ and relevant context $c \in C$, add the triple
$\langle genURI(l^C(c), l^V(v)), \text{rdf:type}, \text{rdfs:Class} \rangle$ (for an RDF-S ontology) or
$\langle genURI(l^C(c), l^V(v)), \text{rdf:type}, \text{owl:Class} \rangle$ (for an OWL DLP ontology)
to *G*.

Step 4.2: Generating subclassOf relations

Since many categorization schemas that are so large that it is infeasible to determine individually whether $S^{c_1}(v_a) \subseteq S^{c_2}(v_b)$ holds, we use the following approximations for creating subclassOf relations:

If $c_{cat} \in C$, then for any $v_a, v_b \in V$, $\langle v_{a,} v_b \rangle \in E$, add the triple
$\langle genURI(l^C(c_{cat}), l^V(v_a)), \text{rdfs:subClassOf}, genURI(l^C(c_{cat}), l^V(v_b)) \rangle$
to G, and for any $c \in C, c \neq c_{cat}$, $v \in V$, add the triple
$\langle genURI(l^C(c), l^V(v)), \text{rdfs:subClassOf}, genURI(l^C(c_{cat}), l^V(v)) \rangle$

If for **all** $v_{a,} v_b \in V$ such that $\langle v_{a,} v_b \rangle \in E$, $c \in C, c \neq c_{cat}$ holds
$S^c(v_a) \subseteq S^c(v_b)$, add the triple
$\langle genURI(l^C(c), l^V(v_a)), \text{rdfs:subClassOf}, genURI(l^C(c), l^V(v_b)) \rangle$
for any $v_{a,} v_b \in V$ such that $\langle v_{a,} v_b \rangle \in E$.

As a simplification, we may use a representative sample and statistic inferencing to determine whether the hierarchy would be a valid subsumption hierarchy for the categories in this particular context. In other words, instead of manually checking this property for the whole input schema, we draw a representative sample from the categories in the original schema and determine manually whether for this set of categories, the above mentionend conditions hold.

The output of step 4 is the ontology G (apart from the ontology header etc.). Figure 1 illustrates this for an ontology that contains classes for one context c and the category context c_{cat}. In this example, the original hierarchy would not be a valid subsumption hierarchy in the context c. If this was the case, there would be an additional `rdfs:subClassOf` relation from $S^c(v_2)$ to $S^c(v_1)$.

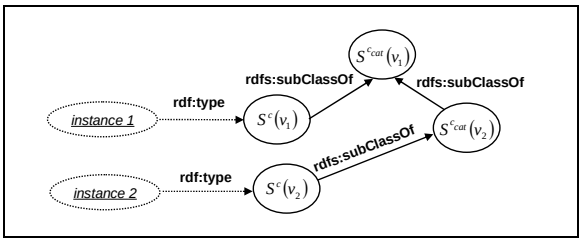

Figure 1. Example of the representation of two categories v_1, v_2 as four ontology classes

3.2 Implementation

Our algorithm depends, as said, on a number of external functions, which we explain in this section.

3.2.1. Function genS

This function takes as input a categorization hierarchy and returns a formal specification, as defined in Section 2.2. In particular, it handles all pre-processing and for disambiguating local labels. Local labels are such that are unique only in their particular position in the hierarchy, e.g. “Portable” in the following example.

```
Computer Equipment
     !- Laser Printers
                  !- Portable
```

One approach to handle such cases is by representing each node by a logical formula that takes into account the label of the node and its position in the hierarchy, as proposed by Giunchiglia, Marchese, and Zaihrayeu [9]. The simplest approach (and often sufficient for our purpose) is to disambiguate local names by concatenating the local name with the label of the path of parent nodes (with a suitable way of escaping colons). This would turn the label “Portable” in our example into:

```
Computer Equipment: Laser Printers: Portable
```

Since most classifications that we found were very limited with regard to the depth of branching, the growth in length created no problems.

A related anomaly is that of a varying semantics of the hierarchical relation by depth of branching. In UNSPSC, for example, the last level of the hierarchy reflects so called “Business Functions” for the next higher level:

```
Computer Equipment
     !- Laser Printers
            !- Sales
            !- Lease
```

We can handle this in the same way as local labels; however, this will usually make it impossible to use the hierarchy as a subsumption hierarchy in this context, since the lease of laser printers is not a subclass of laser printers etc.

3.2.2. Function getContextInfo

This function takes as input a formal categorization and returns a formal categorization which includes an interpretation for every category at every context, except for possibly c_{cat}. Basically, this function returns what will be relevant instances of the classes to be subsumed under the given label in the relevant contexts.

3.2.3. Function genURI

This function takes as arguments a context label and a category label and returns a URI based on this information. It will usually use a given base URI for the resulting ontology and concatenate the category and context labels, possibly separated by a slash. If any of the labels contains extra characters, the function will also rewrite them so that the result is a valid URI. Since we have disambiguated local labels, we can assume that the category labels are unique. In practice, we can often also assume the category labels to be unique.

As an example, the category "TV Set" in the two contexts "Product or Service" and "Category" could be transformed into e.g.

```
http://www.foo.org/myontology/TV_Set_ProductOrService
http://www.foo.org/myontology/TV_Set_Category
```

3.3. Statistical Diagnosis of Conceptual Properties and Relevant Anomalies

Depending on the size of the schema and our knowledge of its properties, we may not know a priori whether the categories in our selected context build a proper subsumption hierarchy. Also, we might need to check for the anomalies outlined in section 3.2, since they will require additional preprocessing.

We advocate the use of representative random samples and deductive statistics for these diagnosis tasks.

When taking a random sample, we should include only such categories v that are not top-level nodes. A nice property of this approach is that we can calibrate the test depending on our needs and thus deal with the unavoidable trade-off decisions between the value of an additional subsumption hierarchy vs. the risk of an undetected inconsistency, which is naturally domain-dependent.

It should be stressed that it can be attractive to make such decisions for one context as a whole, since only this allows for quick and cheap script-based creation of derived ontologies without substantial human intervention and engineering effort.

3.4 Example

We want to built a products and services ontology based on a fictive hierarchical schema for electronic-related categories, as shown in Fig. 2. In this case, the relevant target context is "Products or Services". We create two ontology classes for each category, one reflecting the category concept (e.g. "Radio and TV (Category)"), and

one reflecting respective types of electronic equipment (e.g. “Radio and TV (Product or Service)”). We see that the original hierarchy is not a consistent subsumption hierarchy in the context of products or services, since “TV Maintenance”, read as the actual type of services, is not a subclass of “TV Set”, and “Radio Antenna” is not a subclass of “Radio”. Thus, we arrange the category concepts in a subsumption hierarchy that represents the original edges, but do not arrange the products and services classes in such a hierarchy. All products and services classes are just subclasses of the respective category concepts. Fig. 3 shows the resulting ontology. Elipses represent ontology classes (`rdfs:Class` or `owl:Class`) and arrows represent `rdfs:subclassOf` relations.

If our target context was “cost accounting branch”, then we could additionally arrange the context-specific ontology classes in a subsumption hierarchy, since invoices accounting for radio antennas are usually also regarded as invoices accounting for radios.

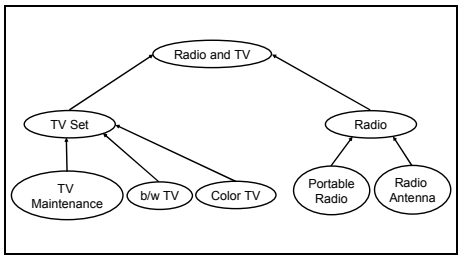

Figure 2. Example of a hierarchical categorization for electronics

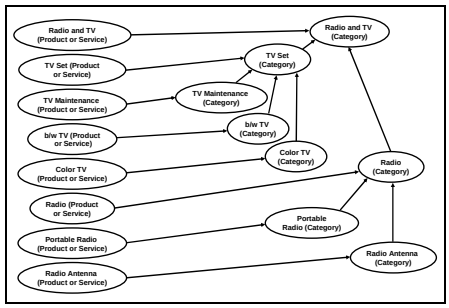

Figure 3. The resulting ontology for the context "Product or Service"

4. Evaluation: eClassOWL and unspscOWL

In order so evaluate our approach, we tried to derive useful e-business ontologies in OWL-DLP from eCl@ss 5.1de [2] and UNSPSC [1]. Our goal was to generate, with minimal human intervention, one eCl@ss ontology that can be used to annotate products and services that are available on the Web, and another UNSPSC ontology to be used for cost accounting purposes, i.e. aggregating incoming invoices by spend categories. This exercise is also practically relevant, since the existing prototypes of UNSPSC and eCl@ss ontologies are not very useful in these application domains as has been detailed in [6]. Both categorization schemas contain more than 20,000 categories, which renders manual steps in the transformation infeasible.

We have implemented preliminary tooling support for our methodology. Our prototype consists of a Java program that expects the informal categorization schema to be stored in a RDBMS. The program accesses the categories via an ODBC link. The reason why we use an RDBMS is that we needed nested queries. Also, it proved to be handy to import the various source formats into the RDBMS using standard tooling instead of developing proprietary import interfaces.

4.1. eCl@ss as a Products and Services Ontology

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. For evaluating our methodology, it was sufficient to import the categories.

The application of our methodology to eCl@ss creates only minor problems. First, the original hierarchy does not constitute a correct subsumption hierarchy if the categories are interpreted as products and services categories. Fig. 4 gives an example of how services of repairing assembly and maintenance technology are subnodes of machine. Thus, the structure of the resulting ontology is as in the example in Fig. 3.

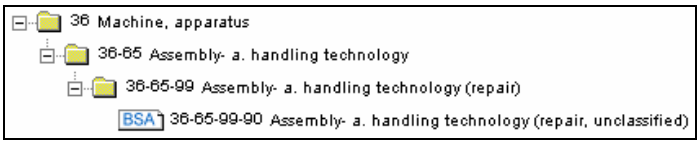

Figure 4. The eCl@ss hierarchy is no subsumption hierarchy in the context of products and services

Second, the resulting ontology is very big: About 25,000 categories in the source taxonomy result in more than 50,000 OWL classes. 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, we were able to load the full ontology into an OWLIM [10] configuration.

As compared to our early approaches described in [6] and [7], our new approach requires only two ontology classes instead of three per category, while the old evaluation results still hold.

While we started our experiments 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.

Generating ontologies in other ontology languages than OWL (e.g. WSML) was also successful and just required expressing the OWL ontology patterns used in the respective target ontology language.

4.2. UNSPSC as a Cost Accounting Ontology

UNSPSC [1] is similar to eCl@ss in its structure, but has more top-level categories and is limited to the hierarchy of labels, while eCl@ss also includes properties and other elements.

Same as eCl@ss, UNSPSC contains hierarchical relations that do not constitute a correct subsumption hierarchy in some contexts, in particular when reading the labels

in the literal sense. For example, we can find the following two candidate inconsistencies:

a) Non-dairy creamers are neither coffee nor tea, and not even a true beverage.

```
-family-[50.20.00.00] Beverages
     -class-[50.20.17.00] Coffee and tea
             -commodity-[50.20.17.14] Non-dairy creamers
```

b) Ice is not a beverage.

```
-family-[50.20.00.00] Beverages
     -class-[50.20.23.00] Non-alcoholic beverages
             -commodity-[50.20.23.02] Ice
```

However, in this second example, the target context of our ontology is "Cost Accounting Categories". If we interpret the labels in this sense, then it is acceptable for "Beverages" to subsume "Ice", since anything spend on ice may be correctly regarded as a beverage-related expenditure. Thus, other than in the example in section 4.1, the ontology classes in the target context "Cost Accounting Categories" can also be arranged in a subsumption hierarchy, which reflects the original order. If we wanted to create a products and services ontology from UNSPSC, the situation would be the same as with eCl@ss, i.e. the classes in this context cannot be arranged in a subsumption hierarchy.

We expect that running our script on other hierarchical categorization schemas, e.g. eOTD or XBRL standard reporting taxonomies should require only slight modifications in the embedded SQL.

5. Discussion

There exists a substantial amount of publications on the analysis of the meaning of taxonomic relationships, especially the fundamental work of [11]. This yielded the insight that there are multiple types of taxonomic relationships, which should be represented separately. In this paper, we have presented a generic methodology for deriving consistent ontologies in a script-based fashion from hierarchical categorization schemas, and successfully applied it to eCl@ss and UNSPSC. While the resulting ontologies are rather lightweight, the cost/benefit ratio of our ontologies seems very convincing, since the amount of human intervention is limited to importing source data into an RDBMS and determining some parameters in a script.

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

- Methodologies for and experiences with the reuse of consensus in classifications, thesauri, and taxonomies for the creation of ontologies. This is the most related field of work. [12] discusses the transformation of tangled hierarchies, as e.g. such derived from ambiguous "broader than / narrower than" taxonomies in library science, into formal ontologies. [13] presents the experiences gained while transforming the constructs of an existing semantic net in the medical domain into an OWL ontology. [14] describe how machine learning approaches can be used to integrate objects from taxonomies available on the Web into a consolidated master taxonomy. [6] is a detailed description of creating products and services ontologies based on UNSPSC and eCl@ss, but requires three classes per category and is also not generically applicable. [15] shows the reuse and semantic enrichment of an

existing hierarchical standard, and demonstrates this for the Art and Architecture Thesaurus (AAT). [16] and [8] are consequent works of this stream of research. An important characteristic of [16] and [8] is that the authors leave the limits of OWL DL in order to capture semantics contained in the original thesaurus, namely to be able to treat classes as instances and vice versa. [9] presents a formal theory of classifications; [17] is an extension of this work and proposes how lightweight ontologies can be derived from such specifications.

- 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 [5] and the DAML+OIL variant from the Knowledge Systems Laboratory at Stanford University [4]. For eCl@ss, there exists one early prototype by Bizer and Wolk [18] and the official release of eCl@ssOWL [19], which is based on our previous work [6].
- Ontology engineering methodologies, implicitly or explicitly focusing on the manual creation of 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. [20] and [21]. The main difference between our work and traditional ontology engineering is that we advocate the script-based transformation without involving an ontology engineer for revising the modeling in every single case.

Our approach is different from previous work in that it allows for the script-based creation of meaningful ontology classes (1) for a particular context while (2) preserving the original hierarchy, even if the latter is not a real subsumption hierarchy in this particular context. The resulting ontologies can be either RDF-S or OWL DLP; in fact, they require no reasoning support beyond `rdfs:subClassOf`, which allows for the use of lightweight, scalable reasoners, while still being able to merge an OWL DLP variant with OWL DL data without leaving the boundaries of OWL DL (which would be the case if e.g. RDF-S meta-modeling would be used).

Our proposal comes not without cost. First, the resulting ontology provides quite limited reasoning support. Second, we create at least two ontology classes per each category, which increases the size of the ontology. However, the unwanted ontology growth has to be set in relation to the low costs of reasoning and to fact that the ontology building process requires almost no human labor.

In general, we agree that a greater amount of e.g. axioms would be desirable. On the other hand, we see no lightweight way of automatically adding more semantics because it cannot be easily derived from the input schemas. Also, we will have to set the resources necessary for the respective enrichment in relation to the gain in automation and the resulting economies.

Acknowledgements: Parts of the work presented in this paper have been supported by the European Commission under the projects DIP (FP6-507483), SUPER (FP6-026850), and MUSING (FP6-027097), and by the Austrian BMVIT/FFG under the FIT-IT project myOntology (grant no. 812515/9284).

References

[1] United Nations Development Programme, "United Nations Standard Products and Services Code (UNSPSC)," available at http://www.unspsc.org/, retrieved March 15, 2007.

[2] eClass e.V., "eCl@ss: Standardized Material and Service Classification," available at http://www.eclass-online.com/, retrieved March 15, 2007.

[3] M. Hepp, J. Leukel, and V. Schmitz, "A Quantitative Analysis of Product Categorization Standards: Content, Coverage, and Maintenance of eCl@ss, UNSPSC, eOTD, and the RosettaNet Technical Dictionary," *Knowledge and Information Systems*, (forthcoming).

[4] D. L. McGuinness, "UNSPSC Ontology in DAML+OIL," available at http://www.ksl.stanford.edu/projects/DAML/UNSPSC.daml, retrieved March 15, 2007.

[5] M. Klein, "DAML+OIL and RDF Schema representation of UNSPSC," available at http://www.cs.vu.nl/~mcaklein/unspsc/, retrieved March 15, 2007.

[6] M. 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, pp. 72-99, 2006.

[7] M. Hepp, "Representing the Hierarchy of Industrial Taxonomies in OWL: The gen/tax Approach," Proceedings of the ISWC Workshop Semantic Web Case Studies and Best Practices for eBusiness (SWCASE05), Galway, Irland, 2005.

[8] M. van Assem, M. R. Menken, G. Schreiber, J. Wielemaker, and B. J. Wielinga, "A Method for Converting Thesauri to RDF/OWL," Proceedings of the ISWC'04, Hiroshima, Japan, 2004.

[9] F. Giunchiglia, M. Marchese, and I. Zaihrayeu, "Towards a Theory of Formal Classification," Proceedings of the AAAI-05 Workshop on Contexts and Ontologies: Theory, Practice and Applications (C&O-2005), Pittsburgh, Pennsylvania, USA, 2005.

[10] A. Kiryakov, D. Ognyanov, and D. Manov, "OWLIM – a Pragmatic Semantic Repository for OWL," Proceedings of the International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005), New York City, USA, 2005.

[11] R. J. Brachman, "What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks," *IEEE Computer*, vol. 16, pp. 30-36, 1983.

[12] A. L. Rector, C. Wroe, J. Rogers, and A. Roberts, "Untangling Taxonomies and Relationships: Personal and Practical Problems in Loosely Coupled Development of Large Ontologies," Proceedings of the K-CAP'01, Victoria, British Columbia, Canada, 2001.

[13] V. Kashyap and A. Borgida, "Representing the UMLS Semantic Network using OWL," Proceedings of the 2nd International Semantic Web Conference 2003 (ISWC 2003), Sanibel Island, Florida, USA, 2003.

[14] D. Zhang and W. S. Lee, "Learning to integrate web taxonomies," *Journal of Web Semantics*, vol. 2, pp. 131-151, 2004.

[15] B. J. Wielinga, A. T. Schreiber, and J. A. C. Sandberg, "From Thesaurus to Ontology," Proceedings of the First International Conference on Knowledge Capture (K-CAP 2001), Victoria, British Columbia, Canada, 2001.

[16] B. J. Wielinga, J. Wielemaker, G. Schreiber, and M. van Assem, "Methods for Porting Resources to the Semantic Web," Proceedings of the First European Semantic Web Symposium (ESWS'04), Heraklion, Greece, 2004.

[17] F. Giunchiglia, M. Marchese, and I. Zaihrayeu, "Encoding Classifications into Lightweight Ontologies," Proceedings of the 3rd European Semantic Web Conference (ESWC 2006), Budva, Montenegro, 2006.

[18] C. Bizer and J. Wolk, "RDF Version of the eClass 4.1 Product Classification Schema," available at http:////www.wiwiss.fu-berlin.de/suhl/bizer/ecommerce/eClass-4.1.rdf, retrieved March 15, 2007.

[19] M. Hepp, "eCl@ssOWL. The Products and Services Ontology," available at http://www.heppnetz.de/eclassowl/, retrieved March 15, 2007.

[20] M. Fernández-López and A. Gómez-Pérez, "Overview and analysis of methodologies for building ontologies," *The Knowledge Engineering Review*, vol. 17, pp. 129-156, 2002.

[21] J. de Bruijn, "Using Ontologies. Enabling Knowledge Sharing and Reuse on the Semantic Web," *DERI Technical Report DERI-2003-10-29, October 2003*, pp. 1-49, 2003.