Ontology-based Product Catalogues: An Example Implementation

Philipp Nowakowski, Heiner Stuckenschmidt

KR&KM Research Group
University of Mannheim

1 Introduction

As more and more business is performed online, the importance of electronic marketplaces where products and services are traded in a semi-automatic way is increasing steadily. One of the basic problems connected to an efficient use of electronic marketplaces is the problem of matchmaking between offers for products or services and requests by potential customers (Veit 2003). Electronic product catalogues provide the basis for offering products on such a market place. Matchmaking algorithms try to find products in a catalogue that meet the requirements of a requester as closely as possible. This task is non-trivial as in many cases, it is not possible to find perfect matches. In this situation, the matchmaking algorithm has to find 'good enough' offers. Determining which offers are 'good enough' often does not only depend on offer and request itself, but also on the preferences of the customer and the intended use of the product or service. This means that matching has to understand the nature of a product to be able to decide if it is similar to what the customer wants and it has to be customizable towards the specific needs of a customer. Another challenge for successful matchmaking in electronic marketplaces is the need to integrate heterogeneous product and service descriptions from different product catalogues to enable the customer to chose from different providers, as the representations used by different participants for their internal purposes are often geared towards the specific needs of the company and show little standardization across different enterprises. In order to be able to use this information in matchmaking, open standards for classifying and describing products have been developed that provide a common framework for describing products and services by assigning products to product classes and by defining a number of properties for describing the concrete offer in more details. Examples of such standards are eCl@ss, UNSPSC and RosettaNet (Hepp et al 2007).

In previous work, we have argued for the need of providing flexible matchmaking services for finding complex products and services in product catalogues and proposed ontology-based representation and matchmaking as a basis for intelligent access to product information (Stuckenschmidt and Kolb 2008). In this paper, we present a demonstration system implementing the idea of ontology-based representation and matchmaking in product catalogues. In particular, we have implemented the prototype of a web shop that uses the eClassOWL ontology (Hepp 2006) as a basis for representing the product catalogue of Pepperl&Fuchs, one of the worldwide leading providers of components for process automation. The representation uses concept IDs and background knowledge from eClassOWL and describes products in terms of class descriptions in the web ontology language OWL. User requests for certain products are also represented as OWL concepts and matched against the available product categories.

The paper is structured as follows: In Section 2 we briefly review the idea of ontology-based matchmaking of product and service descriptions and explain possibilities for guiding the matching process based on user preferences. We then discuss our approach for formalizing the Pepperl&Fuchs product catalogue based on eClassOWL that supports the application of the matchmaking approach in Section 3. Section 4 presents the actual demonstration system and its use. We close with a discussion of lessons learned when applying the idea of ontology-based product catalogues to the Pepperl&Fuchs case.

2 Ontology-based Matchmaking

Li and Horrocks (Li and Horrocks 2004) propose to model offers/sales and requests as description logic expressions such as the following example taken from their paper:

We assume a request asking for a Sales service that offers PCs or laptops with at least 512 MB main memory, at least 256 MB cache memory and a price of at most 500 Dollars. The corresponding request can be formulated using the following concept expression:

Request:

```
restriction(item allValuesFrom(
        intersectionOf(unionOf(PC Laptop)
                restriction(has-main-memory minCardinality(512))
                restriction(has-cache-memory minCardinality(256))
                restriction(price maxCardinality(500)))))
```

We further assume a sales service offering PCs with 256 MB main memory and 256 MB cache memory at a price of 450 Dollars and Laptops with 512 MB Main memory and 256 MB cache memory at a price of 650 Dollar. This service can be described using the following concept expression[1]:

Advert 1:

```
restriction(item allValuesFrom(
    intersectionOf(PC
            restriction(has-main-memory minCardinality(256))
            restriction(has-cache-memory minCardinality(256))
            restriction(price maxCardinality(450)))))
```

Advert 2:

```
restriction(item allValuesFrom(
    intersectionOf(Laptop
            restriction(has-main-memory minCardinality(512))
            restriction(has-cache-memory minCardinality(256))
            restriction(price maxCardinality(650)))))
```

The example describes a sales service and its properties. Here sales identifies the type of service that can be based on an existing classification and the specifics of the service are represented using constraints on the relevant properties. As an important aspect of a sales service, the offered product is represented as well by assigning it to a product class (in this case the class of personal computers) and restrictions on the properties of the individual product such as the memory size and properties of the whole offer such as the amount available and the unit price. A major problem of existing matchmaking approaches that rely on standard description logic inference for computing matches as explained above is the lack of differentiation provided by the four degrees of matching. If an offer only disagrees with the request on a single property it drops out of the class of subsuming matches and ends up in the least attractive class of intersection matches that will contain almost any offer with only the slightest relation to the request.

Refer back to the example from above: It is easy to see that neither Advert 1 nor Advert 2 satisfies the specified request because they both fail to satisfy one of the requirements. While the PCs do not have enough main memory, the laptops are too expensive. The goal of our work is to provide a more fine-grained approach to matchmaking in the context of description logics that differentiates between different degrees of matching in a meaningful way and is also able to provide feedback to the requester on the specific aspects of an offer that did not

[1] We use the OWL abstract syntax to describe concepts. Details of this syntax can be found in http://www.w3.org/TR/owl-semantics/

match the request. Our approach to avoid these problems is based on the idea of a stepwise relaxation the matching conditions by ignoring certain aspects of the descriptions to be matched. The challenge of this approach is to define the relaxation in such a way that the properties mentioned above are met and the relaxation has a meaning in the context of the underlying logic. The approach we have developed is based on the signature - the names of concepts and relations - of the expressions to be matched. The idea is to determine a subset of this signature that can be ignored in the process of computing subsumption relations. With regards to the previous matching example we could perform a relaxation on the memory constraint by rewriting the 'has-memory' property appropriately. This would lead to a reformulation of the matching problem where as a consequence Advert 1 would now match the request. More details of this approach are described in (Stuckenschmidt 2007) and (Stuckenschmidt and Kolb 2008).

The creation of the subset that can be ignored during matching has also been a target of our research resulting in the development of several strategies which expand the subset of relaxed concepts and relations in a way that helps the iterative partial matching algorithm find the most relevant matches as early as possible. For example we have defined strategies that exploit the background ontology (MORE-, LESS-Strategy). Moreover, the fact that the relaxation of the match is guided by concept- and property names makes it easy for the customer to formulate preferences that are to be taken into account. In particular, the user can provide an ordering on the properties of a product, say 'has-memory > price' that can be used to decide that offers that do not meet the price criterion are preferred over offers that do not meet the 'has-memory' criterion.

3 Knowledge Modelling

The main challenge in building the demo system was to create an ontology-based representation of the product catalogue that supports the application of the partial matchmaking method described above and that can be easily combined with offers from other catalogues to meet the needs of the electronic marketplace. The first requirement is met by translating XML-based product descriptions from the catalogue into ontological classes represented in OWL. The second criterion is met by using eClassOWL as a basis for describing the vocabulary and for providing background knowledge in terms of a global hierarchy of product classes. Fortunately, the available catalogue already used eCl@ss IDs for most of the definitions, therefore the main task was to create OWL definitions and link them with the eClassOWL model.

3.1 The eClassOWL Ontology

eClassOWL is an OWL DLP ontology for products and services and is based on the comprehensive categorization standard eCl@ss 5.1 that includes the following main building blocks:

1. products and services concepts (e.g. 'TV Set')
2. properties for these products and services (e.g. 'screen size')
3. values for enumerated data types (e.g. 15", 17", 19", …)
4. a hierarchy of the product concepts reflecting the perspective of a buying organization (!)
5. recommendations which properties should be used for which type of product
6. recommendations which values are allowed for which (object) property.

The eClassOWL ontology is a transformation of the eCl@ss 5.1 standard to the OWL (Antoniou and van Harmelen 2003) format. The transformation process, structure of the ontology and naming conventions are described in full detail in (Hepp 2005).

3.2 Modelling Catalogues in eClassOWL

Our system implementation uses the eClassOWL ontology as the common vocabulary which is used to represent the catalogue data internally. Each catalogue item is basically a complex OWL class description composed of concepts and properties defined in the eClassOWL ontology.

```
restriction(item someValuesFrom
        intersectionOf(C_AGZ377003-tax
                restriction(P_BAA469001 allValuesFrom(V_BAB325001))
                restriction(P_BAD840001 allValuesFrom(V_BAB331001))
                restriction(P_BAD866001 allValuesFrom(V_BAB344001))
                restriction(P_BAD947001 allValuesFrom(V_WAA113001))
                restriction(P_BAA038001 value(-25))
                restriction(P_BAA039001 value(70))
                restriction(P_BAD856001 value(40))
                restriction(P_BAD900001 value(10))))
```

The displayed item describes a special type of switch used in factory automation. It is modeled as a complex OWL description that basically is a conjunction of one taxonomic category (e.g. C_AGZ377003-tax = "Capacitive proximity switch"), and a number of restrictions on properties (e.g. P_BAA469001 = "Voltage type")

defined using value ranges (e.g. V_BAB325001 = "DC, direct current") or concrete values from the eClassOWL ontology.

Expressing our catalogue items using OWL descriptions allows us to leverage partial matching in order to recommend similar products in the catalogue to the user. By using eClassOWL as background ontology, we take advantage of having a product taxonomy present, which we can use for fine tuning our matching tasks. Our system is aware of the taxonomic position of product items, and uses this knowledge during the selection for suitable matching candidates (e.g. first match against all items at the same taxonomic level, then extend the search to all direct children, finally extend to all descendants).

Beyond being able to use our partial matching algorithms, an OWL based data model offers further advantages, especially if compared to a more traditional relational database backend:

- *Easy validation* – Using an OWL Reasoner we can search for inconsistencies in the product catalogue. We can state OWL based assertions about the instances of certain product categories, this would the allow us to easily detect a wrongly classified product using a standard OWL Reasoner (e.g. Pellet2).
- *Data integration* – Merging our catalogue with external data is straightforward because of the inherent graph structure of our ontology based data model. External data can be added into the graph and associated with existing concepts without the expensive schema adaptations that would have been necessary in a relational database backed data model.

4 Implementation of a WebShop based on eClassOWL

The Pepperl+Fuchs web-shop is an example catalogue designed to showcase the possibilities of description logic based data representation (Section 3) and partial matching algorithms (Section 2) in the domain of e-commerce. The web-shop serves as a user accessible test bed for the Partial Matcher Framework (Section 4.2). The web-shop has been implemented using the Grails web-framework which enables rapid development of database backed websites.

4.1 Data Import and Conversion

In order to implement the ontology-based catalogue, the knowledge structures described in Section 3 had to be created from existing data. In particular, large parts of the product catalogue of the Pepperl&Fuchs GmbH catalogue have been extracted from an XML file and stored in a relational database (MySQL) as well as

integrated with the original eClassOWL ontology (Hepp 2005). In addition all of the eCl@ss product descriptions have been converted from CSV files and also stored into the relational database.

4.2 Ontology-Based Matching Conmponent

The Partial Matcher Framework is the software implementation of the partial matcher algorithm as described in Section 2. It is based on previous work done by Stuckenschmidt and Kolb (2008) who created an early prototype software implementation of the partial matching algorithm. In order to meet the requirements of actual product catalogues, the matching algorithm had to be extended with the ability to match concrete values, in particular numbers describing certain aspects of a product.

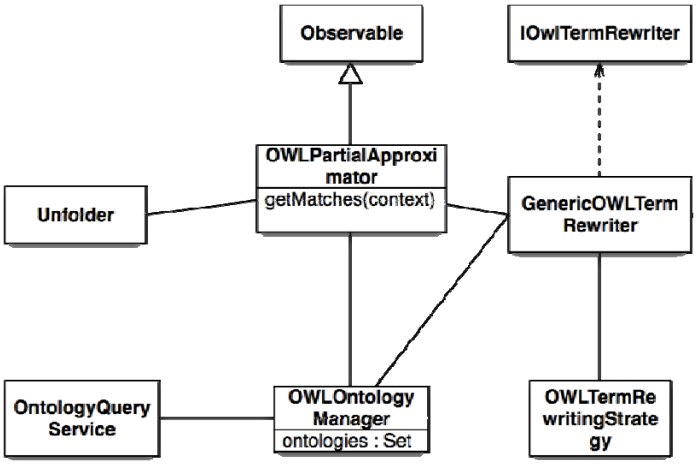


Figure 1 - UML Diagram of the main system components

The framework incorporates several external projects, most prominently the Pellet2 reasoner (Sirin et al 2007), controlled through the OWLAPI library (Bechofer et al 2003). For an in depth system description and accompanied downloads refer to the project website[2].

4.3 Using the System

The main interface of the system is a linear listing of Pepper+Fuchs GmbH content. Selection of a product link leads to a detail screen providing an overview of

[2] http://ki.informatik.uni-mannheim.de/projects/partialmatcher.html

the selected products properties, as well as the ability to start a search for related products. The user starts the search for related products by clicking on one of the buttons denoting the strategy which should be used for matchmaking. The matchmaking process can be influenced by modifying three sets of properties namely the white-, gray- and blacklist. The meaning of those sets is defined as follows:

1. greylist - properties that should stepwise get included into the subset of relaxed properties. The order of inclusion will be determined by the selected approximation strategy.

2. blacklist - properties that should never be included into the approximation. These properties are important for the user and therefore must not get relaxed.

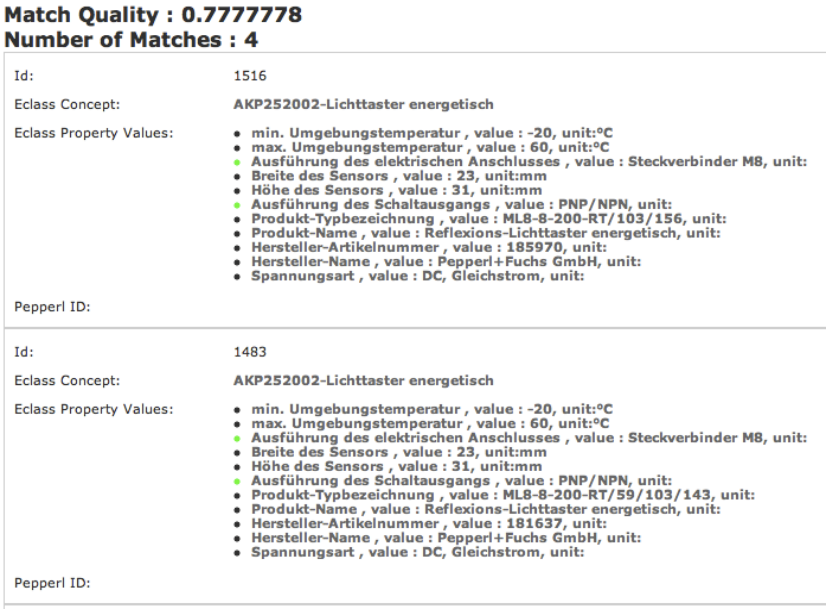
3. whitelist - elements that are unimportant to the user and should get relaxed upfront.

Let's take a closer look at the actual matching steps involved by considering an example matching scenario. Assume the user has selected a product as depicted in Screenshot 1.

Screenshot 1 – Initial product view

The item describes a special type of sensor used in factory automation and the properties that are defined for this item. In our example the user is now interested in similar items from the catalogue. The user moves the properties "Länge des

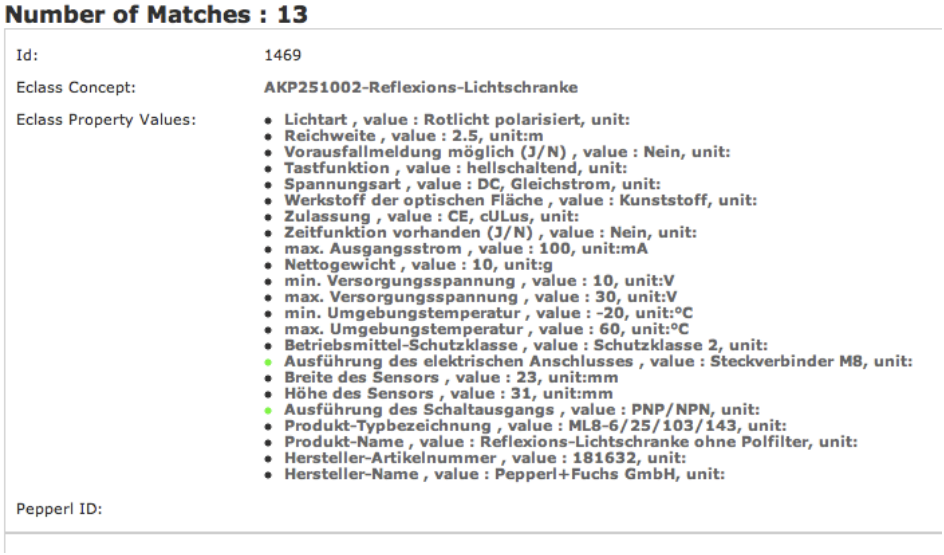
Sensors (Length of sensor)" and "Höhe des Sensors (Height of Sensor)" into the blacklist, because he wants those two properties to serve as hard constraints that should not be relaxed. As a matching strategy he selects the LESS-Strategy (relax the least occurring properties first). The matching process starts and matched items from the catalogue are listed in descending order of their match quality. Relaxed properties are colored green; properties that have not yet been relaxed are colored black.



Screenshot 2 – Matching products with relaxed properties

Screenshot 2 shows the results after the second iteration of the matching process, which so far has relaxed properties "Design of control output" and "Design of electrical connection". We can easily spot that they differ from the original product on the property "Ausführung des elektrischen Anschlusses (Design of electrical connection)" only ("Steckverbinder M8" vs. "Steckverbinder M12"). The matching process continues to relax properties until all properties that were not in the blacklist have been processed. In our example this leads to no additional matches. After all properties have been considered by the matching strategy, the search for matches is restarted but this time the taxonomic position of the current item is included into the approximation. The strategy now extends the set of items that should be matched against. Instead of matching only against all items with the same taxonomic class in the eClassOWL ontology, the strategy includes all items that are at the same taxonomic level (= taxonomic class is a sibling in the taxon-

omy tree of the background ontology). For our concrete example this means that we no longer search for items in the category "AKP252002 – Lichttaster energetisch (Light scanner, energetic)" exclusively, but extend the search over all subcategories of the parent category "AKP249002 – Optoelektronischer Sensor (Optoelectronic Sensor)". Screenshot 3 shows some of the results after the search scope has been widened leading to the inclusion of several matches from the sibling category "AKP251002 – Reflexions-Lichtschranke (Reflection light barrier)".



Screenshot 3 – A matching product from a silbing category

5 Conclusions and Future Work

In this paper, we have presented the prototypical implementation of an ontology based product catalogue system that supports semantic matchmaking of user requests against the product descriptions in the catalogue. We have argued that catalogues benefit from an encoding in the web ontology language OWL, because this provides a formal foundation for comparing products on a semantic level. We further argued that the description of products should be linked to a standardized representation of product categories such as eCl@ss which provides a common terminological basis for taking about product types and their properties. In our work, we have found that the eClassOWL is a very good starting point for ontology-based product catalogues as it combines the fixed terminology of eCl@ss with the formal language OWL that helps us to link product definitions with background information.

From our experience, we conclude that ontology-based product catalogues are feasible from a technical point of view. Development in the field of semantic web

technologies has led to stable ontology language and mature query and reasoning tools. Further, models like eClassOWL are emerging that can provide the terminological basis for catalogues. The biggest obstacle so far seems to be the link between proprietary catalogue solutions and standard classifications like eCl@ss. Standardization efforts are progressing, but legacy data is still a problem. On the other hand, companies that want to compete on the major markets will have to adhere to the leading standards anyways, making the most critical step towards ontology-based product catalogues.

Literatur

Antoniou G, van Harmelen H (2003) Web ontology language: OWL. In Handbook on Ontologies in Information Systems, Pages : 67–92.

Bechhofer S, Volz R, Lord P (2003) Cooking the semantic web with the OWL API. Lecture Notes in Computer Science, Pages : 659–675.

Hepp M, Leukel J, Schmitz V (2007) A Quantitative Analysis of Product Categorization Standards: Content, Coverage, and Maintenance of eCl@ss, UNSPSC, eOTD, and the RosettaNet.Knowledge and Information Systems 13(1):77–114.

Hepp M (2006) Products and Services Ontologies: A Methodology for Deriving OWL Ontologies from Industrial Categorization Standards. International Journal on Semantic Web and Information Systems (IJSWIS) 2(1):72–99.

Hepp M (2005) Representing the hierarchy of industrial taxonomies in owl: The gen/tax approach. ISWC Workshop Semantic Web Case Studies and Best Practices for eBusiness (SWCASE05), Galway, Irland.

Li L, Horrocks I (2004) A Software Framework for Matchmaking Based on Semantic Web Technology. International Journal of Electronic Commerce, 8(4):39–60.

Sirin E, Parsia B, Cuenca-Grau B, Kalyanpur A, Katz Y (2007) Pellet: A practical OWL-DL reasoner. Journal of Web Semantics 5(2):51–53.

Stuckenschmidt H, Kolb M (2008) Partial Matchmaking for Complex Productand Service Descriptions. Proceedings of Multikonferenz Wirtschaftsinformatik (MKWI 2008), München.

Stuckenschmidt H (2007) Partial Matching Using Approximate Subsumption. In Proceedings of the 22nd Conference on Artificial Intelligence (AAAI-07).

Veit D (2003) Matchmaking in Electronic Markets, Jgg. 2882 of Lecture Notes in Computer, Science. Springer Verlag, Berlin.