Mapping XML to OWL Ontologies

Hannes Bohring* and Sören Auer+

* Benedixstr. 8, 04157 Leipzig, Germany
hannesbohring@gmx.de,

+ University of Leipzig, 04109 Leipzig, Germany,
auer@informatik.uni-leipzig.de,

Abstract: By now, XML has reached a wide acceptance as data exchange format in E-Business. An efficient collaboration between different participants in E-Business thus, is only possible, when business partners agree on a common syntax and have a common understanding of the basic concepts in the domain. XML covers the syntactic level, but lacks support for efficient sharing of conceptualizations. The Web Ontology Language (OWL [Bec04]) in turn supports the representation of domain knowledge using classes, properties and instances for the use in a distributed environment as the World Wide Web. We present in this paper a mapping between the data model elements of XML and OWL. We give account about its implementation within a ready-to-use XSLT framework, as well as its evaluation for common use cases.

1 Introduction

Today XML has reached a wide acceptance as data exchange format in E-Business. An efficient collaboration between different participants in E-Business is only possible, when business partners agree on a common syntax and have a common understanding of the basic concepts in the domain. XML covers the syntactic level, but lacks support for efficient sharing of conceptualizations. The Web Ontology Language (OWL [Bec04]) in turn supports the representation of domain knowledge using classes, properties and instances for the use in a distributed environment as the World Wide Web.

The aim of this paper is to bridge the gap between XML and OWL. In particular we propose a strategy how OWL ontologies may be generated automatically out of existing XML data. This has to be done by establishing suitable mappings between the different data model elements of XML and OWL.

Several strategies for mappings have been proposed. Some targeted either more on a general mapping between XML and RDF others aim at mapping XML Schema to OWL without considering XML instance data. But there is no complete approach, which focuses on a transformation from ”legacy” XML instance documents to OWL ontologies. In [DMvH+00] the authors even assume, that a suitable automatic mapping between XML and RDF is impossible, because XML does not contain any semantic constraints. It is

claimed that XML represents the document structure, but does not contain any information about the meaning of the content.

Contrary, other approaches assume that there is some semantics in the XML documents, which can be discovered out of the document structure. Melnik [Mel99a] for instance tries to detect semantics in XML instance documents and to map them to RDF documents, but with a simplified syntax [Mel99b]. Melnik assumes that every XML document has an RDF model.

In [Vie] the authors propose an automatic mapping from XML contents to RDF meta-data (called WEESA) by using an ontology, which was created from the corresponding XML Schema. This ontology contains the model, but has no instances. The XML data will not be mapped to its OWL equivalents. By now, the mapping from the XML Schema to the OWL ontology is created manually. Our aim is it to be able to create the mapping automatically. The WEESA system can be used furthermore to generate (X)HTML web pages with RDF annotations with regard to the constructs defined in the ontology.

Steve Battle's [Bat04] proposal describes a direct mapping between XML and an RDF model, without passing through a special serialization like RDF/XML. Furthermore he assumes, that an XML Schema is available which guides the mapping process in contrast to Melnik's approach, trying to establish generic mappings. Complementary, we try to find a middle course. If we have an XML Schema available, we will use it to create the respective OWL model. But if we do not have a suitable XML Schema, we generate one out of the XML instance document. So we are even in absence of an XML Schema able to extract conceptual relationships.

The authors of [FZT04] describe mappings from XML to RDF as well as from XML Schema to OWL, but these mapping are independent of each other. That means, that OWL instances have not necessarily to suit to the OWL model, because elements in XML documents may have been mapped to different elements in OWL. Furthermore this approach does not tackle the question how to create the OWL model, if no XML Schema is available.

Another interesting approach is the Piazza system [HSM+03]. Piazza does not transform existing XML data into RDF data respectively OWL ontologies, but mediates between pairs of XML sources through a mediating schema. Piazza can help to create a huge semantically interlinked database, but does not build an integrated ontology. A further relevant difference between our approach and the Piazza system is, that the source and the target must be known to produce a mapping. In our work only the source document have to be known, a target ontology is proposed and an appropriate mapping generated.

In this paper we present a framework, which does the whole translation process completely: from a single XML instance document, over a (possibly) generated XML Schema, finally to an OWL model with OWL instances. We also try to detect relational structures in the XML documents, since we assume XML data to contain relational data, as stored in databases. Our approach thus primarily focuses on data oriented XML. The ready-to-use framework implements the mapping in the standard XML technology XSLT.

2 The Mapping

In this section, we present a proposal for a mapping from XML/XML Schema to OWL which raises the XML source documents to the level of an OWL ontology. We assume the XML documents to contain relational structures (see also Figure 1), try to detect them as good as possible and represent them using OWL classes, properties and instances.

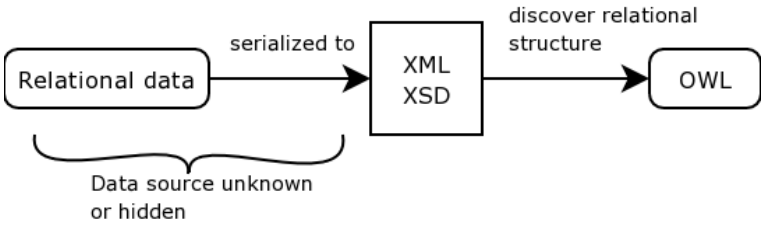

Figure 1: Dataflow diagram

The data model of XML [Bos97] describes a node labeled tree, while OWL's data model is based upon the subject-predicate-object triples from RDF [KCM02]. RDF-Schema [BG02] defines a vocabulary for creating class hierarchies, attaching properties to classes and adding instance data. Hence, we try to exploit the tree structure of XML to create a corresponding class hierarchy. With OWL on the top of RDF and RDFS, restrictions, such as cardinality constraints on properties, can be expressed. This enables the straightforward representation of relational data in OWL: relations/tables correspond to classes, columns to properties and rows to instances. But the detection of relational structures within XML is difficult.

For instance, there is the general question, how to handle nested tags. On the the one hand they can be considered representing a "part-of" relationship or they express a "subtype-of" relationship. Due to focusing on data oriented XML, we can assume a relational structure and use this implicit knowledge about the design of the source documents to improve the transformation process.

For nested elements, we have chosen a middle course: For the case, when one element contains another element, which contains not only a literal, we assume a "part-of" relationship, so that we can assume a 1:N relationship. This is mapped to an `owl:ObjectProperty`, which establishes a relationship between two classes. But we also can create "subtype-of" relations, i.e. we link together named `xsd:complexTypes` and therefrom derived elements. Here multiple inheritance is possible (more than one domain).

Classes (`owl:Class`) will emerge from `xsd:complexTypes` and `xsd:elements` according to the following rules: For the case, that an element in the source XML tree is always a leaf, containing only a literal and no attributes, this element will be mapped to an `owl:DatatypeProperty` having as domain the class representing the surrounding element. XML attributes will be handled equally, i.e. mapped to `owl:DatatypeProperties`, too. Despite there is no real database counterpart for XML attributes and attributes are mostly used in document oriented XML, it makes sense

to assume them representing database columns.

XML Schema also can contain arity constraints like `xsd:minOccurs` or `xsd:maxOccurs`, which we map to the equivalent cardinality constraints in OWL, `owl:minCardinality` and `owl:maxCardinality`. Table 1 summarizes the mapping.

XSD	OWL
xsd:elements, containing other elements or having at least one attribute	owl:Class, coupled with owl:ObjectProperties
xsd:elements, with neither sub-elements nor attributes	owl:DatatypeProperties
named xsd:complexType	owl:Class
named xsd:SimpleType	owl:DatatypeProperties
xsd:minOccurs, xsd:maxOccurs	owl:minCardinality, owl:maxCardinality
xsd:sequence, xsd:all	owl:intersectionOf
xsd:choice	combination of owl:intersectionOf, owl:unionOf and owl:complementOf

Table 1: The mapping is based on these correspondences between XML schema elements and OWL classes and properties

3 Example

In the this section we demonstrate the mapping at example data from the Citeseer Metadata Archive[1]. XML documents similar to the following excerpt represent metadata about scientific publications:

```
<!-- ... -->
<record>
  <header>
    <identifier>oai:CiteSeerPSU:1</identifier>
  </header>
  <metadata>
    <oai_citeseer:oai_citeseer>
      <dc:title>A title</dc:title>
    </oai_citeseer:oai_citeseer>
  </metadata>
</record>
<!-- ... -->
```

[1] http://citeseer.ist.psu.edu/

Starting from an extract of a sample XML file, we generate an XML Schema. A section of the automatically extracted XML Schema to the XML instance shown above could look like:

```
<schema
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:oai="http://copper.ist.psu.edu/oai/oai_citeseer/">
  <!-- ... -->
  <element name="metadata">
    <complexType>
      <sequence>
        <element ref="oai:oai_citeseer"
                        maxOccurs="1" minOccurs="1"/>
      </sequence>
    </complexType>
  </element>
  <element ref="oai:oai_citeseer">
    <complexType>
      <sequence>
        <element ref="dc:title" maxOccurs="1"
                    minOccurs="1" type="xsd:string"/>
      </xsd:sequence>
    </complexType>
  </element>
   <!-- ... -->
</schema>
```

After the transformation the OWL model will contain the classes and properties shown in Table 2. To simplify matters we do not show the OWL syntax.

Naming and namespaces: As you can see in Table 2, there are some names of properties, which have no equivalent in the XML source document. If there are two elements with the same name, but in different levels in the input tree, they would be mapped to a class and property with the same name. This ambiguity is not permitted in OWL, because OWL needs for every resource a unique identifier. Therefore we introduced two prefixes for properties "has" for `owl:ObjectProperties` and "dtp" for `owl:DatatypeProperties`. Instances of classes will get an automatically generated value for `rdf:ID`.

The XML elements in the XML instance documents will be transformed to instances in OWL according to the generated OWL model.

```
<Record rdf:ID="id2248394">
  <hasHeader rdf:resource="#id2248377"/>
</Record>
```

Name/ID	Type	Constraints
Record	Class	hasHeader: minCard = 1 hasMetadata: minCard = 1
Header	Class	hasIdentifier: minCard = 1
Metadata	Class	hasOai_citeseer: minCard = 1
Oai_citeseer	Class	-

Name/ID	Type	Domain	Range
hasHeader	ObjectProperty	Record	Header
hasMetadata	ObjectProperty	Record	Metadata
hasIdentifier	ObjectProperty	Header	Identifier
hasOai_citeseer	ObjectProperty	Metadata	Oai_citeseer
dtpIdentifier	DatatypeProperty	Header	xsd:string
dc:title	DatatypeProperty	Oai_citeseer	xsd:string

Table 2: The generated OWL model for the example Citeseer data

The `owl:DatatypeProperties` are represented in two ways. The first one references locally defined `owl:DatatypeProperties`.

```
<Header rdf:ID="id2251828">
  <dtpIdentifier rdf:datatype="&xsd;string">
    oai:CiteSeerPSU:1
  </dtpIdentifier>
</Header>
```

The second way describes externally defined elements as those from the Dublin Core Metadata Initiative[2].

```
<oai_citeseer:oai_citeseer rdf:ID="id2243767">
  <dc:title rdf:datatype="&xsd;string">A title</dc:title>
</oai_citeseer:oai_citeseer>
```

For a better support of document oriented XML, we also introduced a special datatype property. This is used, if an `xsd:element` contains literal content as well as at least one `xsd:attribute`. The `xsd:element` becomes an OWL class, the `xsd:attribute` is mapped to a datatype property and the literal content will be stored in the additional OWL datatype property.

Information about data types found in the XML Schema will be also integrated in the ontology, as can be seen in the examples with `owl:DatatypeProperties`. For the range value of `owl:DatatypeProperties` we use the built-in data types from XML Schema [BM04].

[2]http://dublincore.org/

4 Implemented Framework

The mapping is implemented in XML stylesheet language transformations (XSLT [Cla99]) and thus interoperable with different programming languages. For XML data, to which no XML schema is attached, a suitable intermediate XML schema is generated. The overall architecture of the framework is shown in Figure 2.

The conversion process requires at most three steps (XML instance data only) and at least one step (XML Schema only). When processing XML Schema only, we only create the model of the ontology with classes and properties.

If we have XML instance data only we have to make an intermediate step. The first step extracts an XML Schema out of the XML instance data so that we could create the model in the second step. We decided to create the OWL model only out of an XML Schema and not from an XML instance file directly for reasons of maintainability. Stefan Mintert states in [Min05] that in every XML instance document an XML Schema is implicit existent, so we can try to extract it. Unfortunately, such an automatically generated XML Schema could be incomplete, because XML instance documents do not contain as much information about constraints as a manual created XML Schema. There are also several XML Schema components, which cannot (yet) be discovered via stylesheet driven extraction (e.g. SimpleTypes, patterns, substitionGroups, facets, the ID/IDREF mechanism, etc.). Furthermore, XML instance documents can contain optional elements or attributes, which were not in the sample document. Thus they are not in the generated XML Schema and OWL ontology. For this reason we need for the XML Schema extraction a preferably representative XML instance document, so that the XML Schema can serve as a good basis. A further advantage of having such a basis is its reusability. This XML Schema extraction is based upon an XSLT stylesheet from Charlie Halpern-Hamu [HH99], which we have extended and adapted to our framework. In future versions of this stylesheet it is planned to use multiple source documents and to add the detection of missing XML Schema components to improve the XML Schema extraction process.

A stylesheet, which converts the XML instance data into the instances part of the ontology, is created simultaneously. This stylesheet will be configured automatically to adjust the transformation process of the instances to the OWL model. It determines whether elements become classes or properties. That is necessary, because the XML instance data can have optional elements or attributes and the created stylesheet will be their common denominator.

To support the separation of model and data, the OWL model will be stored separately from the OWL instances. The OWL instances will be connected to their model using the `owl:import` property. Therefore every OWL instance, which references the OWL model will obtain an adjustable namespace prefix.

By now our implementation consists of 4 XSLT stylesheets. A further stylesheet is generated automatically for the conversation of XML instance data to OWL instances. The framework is designed to be easily extensible, so that the support for the missing XSD components can be included and a better support for document oriented XML can be integrated.

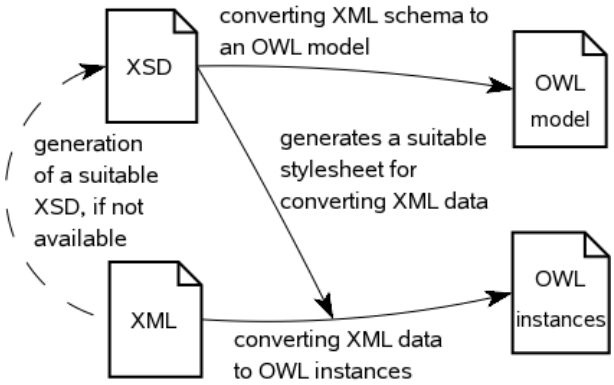

Figure 2: This chart illustrates the operating sequence of the application

5 Use Cases

As use cases for evaluating the presented approach we used publicly available bibliographic data in XML from Citeseer, XML generated from Relational Database Management Systems like MySQL or Firebird and XML data generated with Microsoft Excels XML export. Unfortunately, none of these exported XML data is in pure data oriented XML, so we have to deal with more or less document oriented XML.

The XML exporter from MySQL dumps a whole database and can easily mapped to an ontology. We also want to map relational constraints like foreign keys, but MySQL does not support them yet. So we examined the Firebird RDBMS, which makes use of such relational constraints. Unfortunately, its XML exporter exports only a single database table into an XML file. Because the generated XML Schema is common for all XML files exported from a certain Firebird database, the OWL model is also suitable for the all respective OWL instances. Furthermore, the generated stylesheet, which was created by using the XML Schema, can be used for all of the XML files exported from this database. The resulting OWL instances can now be combined through the `owl:imports` mechanism. This has the advantage, that we obtain a modularized ontology. A problem is, that the foreign key constraints are not easily detectable, because there is no information indicating foreign keys in the XML file. Hence in many cases they might be detected out of column names (e.g. `author_id` referencing the column `id` in the authors table). This functionality is planned for future versions of the framework.

We have chosen Microsoft Excel as another use case, since it is widely used for working with relational data. An Excel spreadsheet can be exported into XML and a spreadsheet in MS Excel has much in common with a table in a relational database. Unfortunately, the XML exported from an Excel file is document oriented XML. Excel's XML does not only describe the structure of the data, but it is also used to encode layout information. It is difficult to distinguish whether an element is semantically important or responsible for layout.

Databases may contain huge amounts of data and the XML files with the exported data might be very large (e.g. the Citeseer archives contain 2 GB of metadata). Therefore it is interesting how the transformation scales. This transformation can be compared with the initial loading of a relational database. The results of the performance evaluation for the Citeseer data are summarized in Table 3.

Number of records	Lines	File size	Time for Processing
10	1122	62KB	0.110s
100	18078	952KB	4.402s
1000	180752	9746KB	4m 52.060s

Table 3: Performance evaluation for transforming Citeseer data.

6 Summary and Future Work

We presented an approach for generating ontologies automatically out of existing XML data with relational origins. This is crucial for referencing and integrating conventional XML and relational data sources into the Semantic Web.

OWL is semantically much more expressive than needed for the results of our mapping. Furthermore (and especially if no XML Schema is available) the transformation is based upon a heuristic method, so that possibly no optimal solution will be reached. For this reason, there has to follow some subsequent manual work after the initial step of converting the source documents, to refine and adapt the ontology until it suits the requirements.

Our subsequent work will be focused on not yet supported XML Schema components, so that more detailed and precise ontologies can be generated. Furthermore, we want to improve the support for document oriented XML (also with mixed content) by letting the user control the transformation process to get more influence on the mapping. We also want to reach an improvement of the performance when processing the OWL instances.

We presented an efficient implementation of our approach within an extensible XSLT framework. The framework is ready-to-use with arbitrary XSLT processors and available for download[3].

References

[Bat04] Steve Battle. Round-tripping between XML and RDF. In *International Semantic Web Conference(ISWC), Hiroshima, Japan, November 2004*. Springer, 2004.

[Bec04] Sean Bechhofer. Web Ontology Language (OWL) Reference version 1.0. W3C. Technical report, W3C, http://w3.org/TR/owl-ref/, 2004.

[3] http://semanticscripting.org/XML2OWL_XSLT

[BG02] Dan Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. Technical report, W3C, http://www.w3.org/TR/2002/WD-rdf-schema-20021112/, 2002.

[BM04] P. V. Biron and A. Malhotra. XML Schema Part 2: Datatypes W3C Recommendation. Technical report, W3C, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/, 2004.

[Bos97] Bert Bos. The XML data model. http://www.w3.org/XML/Datamodel.html, 1997.

[Cla99] James Clark. XSL Transformations (XSLT). Technical report, W3C, http://www.w3.org/TR/xslt, 1999.

[DMvH+00] Stefan Decker, Sergey Melnik, Frank van Harmelen, Dieter Fensel, Michel C. A. Klein, Jeen Broekstra, Michael Erdmann, and Ian Horrocks. The Semantic Web: The Roles of XML and RDF. *IEEE Internet Computing*, 4(5):63–74, 2000.

[FZT04] Matthias Ferdinand, Christian Zirpins, and D. Trastour. Lifting XML Schema to OWL. In Nora Koch, Piero Fraternali, and Martin Wirsing, editors, *Web Engineering - 4th International Conference, ICWE 2004, Munich, Germany, July 26-30, 2004, Proceedings*, pages 354–358. Springer Heidelberg, 2004.

[HH99] Charlie Halpern-Hamu. Transform a sample instance to a schema. http://incrementaldevelopment.com/papers/xsltrick/, 1999.

[HSM+03] Alon Halevy, Dan Suciu, Gerome Miklau, Igor Tatarinov, Jayant Madhavan, Nilesh Dalvi, Peter Mork, Xin luna Dong, Yana Kadiyska, and Zachary Ives. The Piazza Peer Data Management Project, May 24 2003.

[KCM02] Graham Klyne, Jeremy J. Caroll, and Brian McBride. Resource Description Framework (RDF): Concepts and Abstract Syntax. Technical report, W3C, http://www.w3.org/TR/2002/WD-rdf-concepts-20021108/, 2002.

[Mel99a] Sergej Melnik. Bridging the gap between XML and RDF. http://www-db.stanford.edu/ melnik/rdf/fusion.html, 1999.

[Mel99b] Sergej Melnik. Simplified Syntax for RDF. http://www-db.stanford.edu/ melnik/rdf/syntax.html, 1999.

[Min05] Stefan Mintert. Schluesselqualifikation; XML jenseits des Mainstreams. *iX*, 8:48–51, 2005.

[Vie] Gerald Reif Vienna. WEESA - Web Engineering for Semantic Web Applications. http://citeseer.ist.psu.edu/725648.html.