Mapping XML-Documents into Object-Model

Tanja Sieber, László Kovács

Department of Information Technology, University of Miskolc
Miskolc-Egyetemváros, H-3515 Miskolc, Hungary
{sieber, kovacs}@iit.uni-miskolc.hu

Abstract: In the field of technical documentation there are currently a lot of XML-based documentations that exist or are under development. Some of them are based on DTDs or XMLSchemas that define the structure of XML documents of a special document type, the majority of them don't. Whereas in the context of webpublishing ontologies have increased as a very useful and powerful method towards a more effective machine processing of existing files, in the field of technical documentation ontologies respectively formal representation languages like F-logic are not used at the time. Our approach uses a simple mapping method to derive object-models from XML-files to access sets of distributed XML documents on a conceptual level, that allows further-on processing in a formal represented language like they are used for ontologies.

Keywords: XML-processing, object-model, ontologies, technical documentation, XMLSchema, knowledge representation, ODL

1 Introduction and Motivation

Technical documentation is a summarized term for all information, that a user of a product can be provided with. The parts of the documentation give information about the product itsself, the usage of the product and the intended behaviour of the user. Internal documentation contains all kinds of construction and productization documents, duty documents and documentation of the quality ensurance measures. It is focussed on the employees of the manufacturer. External documentation is directed towards the user, who has to be informed about the correct usage of the product with the help of manuals, operating instructions and safety instructions. [3]

Technical writers are in the situation of being on the one hand the "recipient" and "collector" of data, information and knowledge from all over distributed enterprise processes and on the other hand the data-, information- and knowledge- "sender" for the users of technical documentation [4]. Problems and challenges of data, information and knowledge management occur exactly in the same way considering the process of the development of technical documentation.

In general the development processes are embedded in complex structures [4, 6]. The concerned knowledge is allocated towards involved persons of different departments in different production places and in different applications. As a consequence of that fact, users have difficulties in searching and finding relevant information. Most interesting of all is that in such a process nearly everyone has both roles: the one of having knowledge/information/data and the person searching and needing it. The main task is to collect knowledge/information/data and to place it, where it is needed.

Based on these occurring problems and challenges in the field of technical documentation our research attempts to analyze and extend existing methods of data, information and knowledge representation methods according to their possible applications in the field of technical documentation management. Our research focusses on the development of gaps between typical technical documentation's format like XML-based data and the formats or languages, which are commonly used in representation methods of data, information and knowledge.

With respect to that intention that this document gives in the next section shortly an overview of the term ontology, the different types of ontologies, their application in semantic web and introduces XML as storing format of technical documentations. In the third section of this we will summarise XML, XML Schema, ODL and conversions of XML into object-oriented models. Then, in section 4, we present our idea and algorithm of a simple mapping of XML into Schema. Finally, the paper concludes and gives a short overview of our future research work.

2 Ontologies

An ontology [7] in the informational sense is a formal defined system of terms and/or concepts and relations between these terms. In addition ontologies can contain rules. With the idea of semantic web ontologies increased in the last years. Ontologies are commonly used in artificial intelligence and knowledge representation.

2.1 Types of Ontologies

According to [7] the representation of ontologies lead to different types of ontologies:

- Taxonomy: Objects are strong hierarchically classified, e.g. A is child of B. Taxonomies often are visualized in trees.

- Thesaurus: Objects are related (e.g. A is a B; A is related with B).
- Logic-mathematical representation: Objectrelations are presented in formal notations (e.g. synonym(a,b):=synonym(b,a);).

The possibility of relations over relations (in RDF so-called reification) and rules are highly complex and are therefore very rarely used, though exactly these characteristics distinguish ontologies from other systems of concepts. Analogously to a database, wherein structure and data form the whole, an ontology consist of rules and concepts. Languages for the description of ontologies are RDF-S, DAML+OIL, F-Logic, OWL, WSML or XTM. Using rule-based representations in so-called deductive databases further facts can be deduced from stored relations.

2.2 Ontologies and Technical Documentation

According to an article of Berners-Lee et. al. in Scientific American [9] the semantic web is "an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation". Additional to for-humans-readable form information should be formally represented, so that machines can process it.

This annotation of HTML/XML-webpages happens through knowledge representing languages like RDF or OWL. Background is, that on the one hand better categorising possibilities are provided and on the other hand conclusions can be made using the based ontologies. With an according quality/granularity of annotation a high level of automatic processing can be achieved.

Whereas in the context of webpublishing ontologies increased as a very useful and powerful method to a more effective machine processing of existing files, in the field of technical documentation ontologies respectively formal representation languages like F-logic are not used at the moment. That’s in fact very estonishing, because in comparison with the World Wide Web the field of technical documentation seems to be a perfect candidate for ontologies, because it is a special domain, for which ontology-based attributes should be more easy, faster and effective creatable than for ‘semanticWeb’ with many different domains. Our idea therefore is to analyze in our research work, if and how ‘semantic Doc’ can work. With the development of XML in the middle of the 90s the awareness arose, that a standardized storage format should be used in order to have a better basic for further machine-based processing of data.

2.3 XML as Storing Format of Technical Documentations Data

XML [1] was developped by an XML Working Group formed under the auspices of the World Wide Web Consortium (W3C) in 1996.

Four of the design goals for XML have been:

- XML shall be straightforwardly usable over the Internet.
- XML documents should be human-legible and reasonably clear.
- It shall be easy to write programs which process XML documents.
- XML documents shall be easy to create.

In fact XML influenced the technical writers world: meanwhile a lot of redactional tools offer the technical writers an easy-accessible interface and store the published data in XML. The technical writers became more and more semantic designers as they have to think about more in a semantic way than on the presentation way, when developing their documentation. Interesting that we can see here, that the above created term semanticDoc is already reality in some companies. The further-going questions are, how these semanticDoc files can be further-processed, how a gap between semanticDoc and ontologies can be realized and if the structure of the according documentations can be prepared in a better way for such a process? And naturally what the benefit will be from such a further processing? Developing XML-related applications a major issue is the extraction of data, the processing of that data and their computational appliance depending on the desired application or business logic [2].

3 XML and Object-Oriented Model

3.1 XML

XML documents [1] are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. The XML format is usually called semi-structured data format as it contains both the

- data structure and the
- data values.

The structure of a XML document is given with the tag formalism:

```
<tagname attribute=value ..>
        value
</tagname>
```

The value belongs to a structure element denoted by tagname. As a sructure may contain another structure elements, the elements constitute a hierarchy. The XML document must contain a unique root element. Thus the structure of the XML document is described with a tree. The main characteristics of an simplified XML tree are:

- one root element
- the node types are: element, text, attribute
- there is no child at nodes of type text
- exactly one text child node at attribute nodes
- arbitrary number of element children nodes at element nodes
- one or zero text child node at element nodes
- the tree is ordered
- each no-text node has a label
- different nodes may have the same label
- the text values belong to data types

```
<CARS>
        <CAR   id=”112”>
                <TYPE>Fiat</TYPE>
        </CAR>
</CARS>
```

Figure 1a
Sample XML document in text format

The common approach for implementation to process and compute on XML documents is characterized by the transformation of XML documents into the respective representation of the application’s programming language. For example, by using Java XML documents are basically transformed into Java objects.

As we are interested to process XML documents using ontologies, it is quite interesting what kind of representation offers a powerful further-processing possibility for that.

3.2 XML Schemas

Using the XML formalism arbitrary tree can be generated from a given set of elements, attributes and text values. Although these trees are formally correct they do not represent semantically valid information. The data structure of the investigated problem area can be mapped only to a limited number of tree structures. To restrict the generated XML structures to the valid ones, an XML schema mechanism should be used. The XML schema rules can be used to define the valid tree structures. It can be used to define the

- type of child element nodes
- type of attribute nodes
- data type of text nodes
- order and cardinality of child nodes
- check conditions on the text values

There are two main kinds of schema rule systems: the DTD and XMLSchema standards. The DTD is the simplier variant. It does not contain among others the complex data type and data value checking, the complex structure definition. The DTD schema is given within the DOCTYPE tag. It uses the ELEMENT tag for the elements, ATTLIST tag for the attributes.

The XMLSchema standard is better integrated into the XML world as it itself uses the XML formalism. It provides more elements to describe the

- ordering and cardinality of child nodes
- checking of data type
- grouping of elements
- checking of references and other integrity rules

The following example shows an XMLSchema example:

```
<ElementType name=”employee” content=”eltOnly” model=”closed”>
            <element type=”department” minOccurs=”0” maxOccurs=”1”/>
            <AttributeType  name=”Id”  dt:type=”int”/>
             <attribute type=”id”/>
</ElementType>
```

3.3 Object-Oriented Model

Ontologies are formalized, see also section 2.1 of this abstract, in object-oriented and logic-based languages from deductive database community, e.g. F-Logic [10] or Datalog.

In order to build a bridge between these kinds of programming languages and the XML-based data we decided to develop an object-oriented model using ODL, “a specification language to define the specifications of object types based on the ODMG ODL” [11].

ODL can be understood as a definition language for object specifications, that defines the characteristics of types, including their properties and operations. ODL defines only the signatures of operations and does not address definitions of the methods that implements those operations. ODL is intented to define object types that can be implemented in a variety of programming languages.

In the ODL model, user-defined types can be specified by means of the *class* and *interface* mechanisms. The definition of a class is achieved through the specification of its *properties* and *operations*. The set of properties defines the state of an object and consist of *attributes* and *relationships*. An attribute has a state and its value can be a literal or an object identifier, but it is not a “first class” object. A relationship, too, has a state and is not a “first class” object. Only binary relationships can be represented and the representation consists of a pair of inverse references (from each object to the other). If “behaviour” is used to refer to operations and “state” to refer to properties, then a class is a specification of both the abstract behaviour and abstract state of an object type. Classes are *instantiable*. Besides classes, types can also be specified by means of *interfaces*. An interface is a concept similar to a class, but it represents the specification of only the abstract behaviour of an object type, i.e. the object’s operation *signatures*. Interfaces are non-instantiable. The following definition shows a sample ODL schema:

```
class Course {
keys name,number;

attribute string name;
attribute string number;
relationship List<Section> has_sections
inverse Section::is_section_of;
relationship Set<Course> has_prerequisites
inverse Course::is_prerequisite_for;
relationship Set<Course> is_prerequisite_for
inverse Course::has_prerequisites;

offer(in Semester) raises(already_offered);
drop(in Semester) raises(not_offered);
                    }
```

3.4 Conversion

Erdmann and Studer [10] describe in their article of the year 2000 a functionality called DTD-maker, that they implemented in the ONTOBroker-Software and that allows the creation of DTDs out of ontologies. For the reverse direction they write: „Since, it cannot be expected that application development always starts with modeling an ontology we must take care of existing XML document structures or XML Schemas, how they can be related to an ontology, or how they can be used to derive an ontology.” And further on “This reverse direction allows (i) to keep and use the existing XML documents and structures, (ii) to use all existing applications that create, access, manipulate, filter, render, and query these documents, and (iii) at the same time to benefit from the domain knowledge modeled in the ontology by utilizing smarter applications that can complement (or even replace) the existing applications in some areas esp. query answering.” As far as we know that mentioned reverse direction is still an open question and our research work aims in the development of algorithms for generating an ODL model for XML documents.

4 Simple Mapping Algorithm

The mapping from XML into object model is separated in two steps: first a Schema has to be generated from the XML document and in a next step outgoing from this Schema the object model will be developed.

4.1 Generating Schema from the Document

In this project, a simplified schema functionality was used to focus on the core elements of the transformation. The schema includes the containment, the labeling and the cardinality rules but excludes the ordering rule. Thus in this investigation, the XML documents are treated as unordered trees. This restriction is less important than the others restrictions in viewpoint of the semantic evaluation.

The applied schema description can be given with the following elements:

$E = \{e\}$ the set of elements

$N = \{n\}$ the set of labels

$L : E \rightarrow N$: the labeling function

$V \subseteq E \times E$: the parent–child relationship, the first tag is the parent and the second is the child.

$C : V \rightarrow \{(0:1), (1:1), (0:*), (1:*)\}$ the cardinality function. The meaning of the parameters:

0: the child is an optional tag

1: compulsory tag

:1 single occurence

:*   multiple occurences

The input for the schema integration is a set of XML documents and the output is a schema of the given form. To perform the merge step, a separate schema is generated first for every XML documents. The generated schema is the most resctricted one from the possible schema instances. In the next step, the local schemas are merged into a common schema. In favour of simplicity, the schema graph is decomposed into a set of schema units. A schema unit contains only one parent element and the other elements are the children of the parent. Thus a schema unit describes the structure of one element. The list presentation of the schema unit is

(p; (e1,C(e1)), (e2,C(e2)),…, (ei,C(ei)),..)

where p and e1,e2,..,ei are the elements in the documents. The detailed algorithm for schema extraction is the following:

1 For every element collecting the occurences with the set of children into one group

2 Ordering the child elements by the label value

3 Generate list presentation of the schema unit for every element occurence in the following form:

   (p; (e1,(1:m)), (e2, (1:m)),…, (ei, (1:m)),..)

   where m is either 1 or *.

4 Merging the schema descriptons for every elements. The merging rule is the following:

   - if one element occurs in both schema the first part of the output cardinality value is 1 if both input values are 1, otherwise 0
   - if one element occurs in both schema the second part of the output cardinality value is 1 if both input values are 1, otherwise *
   - if one element occurs in only one schema the first part of the output cardinality value is 0
   - if one element occurs in only one schema the second part of the output cardinality value is 1 if the input values are 1, otherwise *

Based on this algorithm, the general schema for every element can be generated using iterative merging steps.

4.2 Generation of the OO Model from Schema

The second main phase of the document processing is the generation of the OO model. For description of the OO model a standard, the ODL model was selected. The mapping is performed on the following rules:

- single element $\rightarrow$ attribute
- compound element $\rightarrow$ class
- containement of compound element $\rightarrow$ relationship

The following example shows a simple conversion.

```
<students>

 <student id="S44098">

     <name>Kelly Griftman</name>

      <year>Senior</year>
      <status>full-time</status>
     <program>HIST6010</program>
 </student>


 <student id="S44091">

     <name>Tom Martin</name>

      <year>Senior</year>
     <program>HIST6010</program>
    <program>MATH4301</program>
 </student>

</students>
```

Figure 2a
The sample XML

```
S1:   ( student; (id,(1:1)),(name,(1:1)),(year,(1:1)),(status,(1:1)),(program(1:1)) )

S1:   ( student; (id,(1:1)),(name,(1:1)),(year,(1:1)), (program(1:*)) )
```

Figure 2b
The schema instances for the student element

```
S :   ( student; (id,(1:1)),(name,(1:1)),(year,(1:1)),(status,(0:1)),(program(1:*)) )
```

Figure 2c
The generalized schema

```
Class Student {
   keys id;
   attribute string id;
   attribute string name not null;
   attribute string year  not null
   attribute string status;
   relationship Set<subject> program;
}
```

Figure 2d
The result ODL model

Conclusions

In this paper we presented the motivation and the approach of one of our research work's goal – the generation of an ODL model for XML documents – in order to have a basic for the further processing of XML data using ontologies and also the base for creating ontologies out of XML documents, which will be – as far as we know – the first steps from this direction.

The presented algorithm deals with a simple mapping method, considering strong-label identities and level-wise proceeding. For our future research work we will extend that algorithm and develop new algoithms for strong label identity but and tree-wise proceeding. Our idea for that is a comparison of the edit distance with the created schema distance. In a last step weak label identity will be examined from us considering probabilistic label mapping based on structure and domain.

The so-developped Schemas will be transformed into ODL and the so-created ODL models mapped into Datalog or another ontology-language.

As this research work is only at the beginning, a very important next step will be to find or develop a framework, that is suitable for implementing and testing our algorithms. The focus of this tests will not only be to find out, if the algorithms are working, but also to define quality measure metrics for the XML-data concerning special requirements for their further processing in object-oriented surroundings.

References

[1] T. Bray. J. Paoli, C. Sperberg-McQueen, E. Maler, and F. Yergeau: *Extensible Markup Language (XML) 1.0 (Third Edition)*, World Wide Web Consortium (W3C), http://www.w3.org/TR/REC-xml/, Feb. 2004

[2] B. D. Heumesser, R.-D. Schimkat: *Deduction on XML documents: A case study*. Proceedings of the 14th International Conference of Applications of Prolog, Tokyo, Japan, October 2001

[3] T. Sieber, L. Kovács: *Technical documentation: functional design and information modelling as methods to structure published content*.

Proceedings of 5th International Conference of PhD students, University of Miskolc, Hungary, August 2005

[4] T. Sieber, L. Kovács: *Technical Documentation: Terms, problems and challenges in managing data, information and knowledge*. Proceedings of 5th International Conference of PhD students, University of Miskolc, Hungary, August 2005

[5] T. Sieber: *Funktion von Wissen im Kommunikationsprozess*. Seminararbeit, FH Karlsruhe, 1999

[6] Ontoprise GmbH: *Accelerate Development*. http://www.ontoprise.de, Sept. 2005

[7] Wikipedia: *Ontologie (Informatik)* http://de.wikipedia.org/wiki/Ontologie_%28Informatik%29, Sept. 2005

[8] A. B. Cremers, U. Griefahn, R. Hinze: *Deduktive Datenbanken*. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 1994

[9] Wikipedia: *Semantic Web*. http://de.wikipedia.org/wiki/Semantic_Web, Sept. 2005

[10] M. Erdmann, R. Studer: *How to Structure and Access XML Documents With Ontologies*. Data and Knowledge Engineering. 2000

[11] Centre de Ressources INFOBIOGEN, *The Object Definition Language.* http://www.infobiogen.fr/services/eyedb/pub/manual/node5.html, Oct. 2005