Generic Multilevel Approach Designing Domain Ontologies based on XML Schemas

Thomas Bosch[1] and Brigitte Mathiak[2],

[1] GESIS - Leibniz Institute for the Social Sciences, Square B2, 1,
68159 Mannheim, Germany
[2] GESIS - Leibniz Institute for the Social Sciences, Lennéstr. 30,
53113 Bonn, Germany
{Thomas.Bosch, Brigitte.Mathiak}@gesis.org

Abstract. Designing an ontology for a specific domain is a time-consuming process. In many cases, information sources like XML Schemas serve as a basis for ontology engineers to conceptualize the intended ontologies. The ontology design process is sped up significantly when XML Schemas are transformed automatically into generated ontologies. An XML Schema Metamodel Ontology has been designed to represent the components of the XML Schema abstract data model. The generated ontologies' classes are defined as sub classes of this ontology. The classes specified for the generated ontologies are intended to be further supplemented with additional semantic and domain specific information defined in domain ontologies. The resulting ontologies are as usable as ontologies that were constructed completely manual, but with a fraction of necessary effort.

Keywords: Semantic Web, Ontology Design, XML Schema

1 Introduction

XML has reached wide acceptance as a data exchange format in e-business. Data and metadata structured by ontologies can be published in the increasingly popular LOD cloud to get linked with a huge number of other RDF datasets [1]. As RDF is an established standard there is a plethora of tools which can be used to interoperate with data and metadata represented in RDF. An effective and efficient cooperation between e-business partners is only possible if they agree on a common syntax and have a common understanding of the domain classes. XML Schema and OWL support differing modeling goals. The data model of XML describes a node labeled tree [2], the syntactic structure of XML document instances. OWL, however, is based on the subject-predicate-object triples from RDF [3], based upon formal logic, and describes semantic information about domain classes as well as their relations and therefore allows the sharing of conceptualizations. XML represents a large set of information in many domains. This fact has driven the development of general-purpose tools for converting XML Schemas to OWL ontologies. The direct mapping from XML and XML Schema to RDF and OWL is not sufficient, since it only transports information about the syntactic structure of XML document instances. Semantic information has to be added in a further step. The aim of this paper is to bridge the gap between XML and OWL by lifting the syntactic level of XML documents to the semantic level of OWL ontologies. The process of designing domain ontologies is extremely time-consuming. XML Schemas describing specific domains are often existent in early stages of the ontology design process. In this paper, the authors describe a generic multilevel approach which accelerates the process of designing domain ontologies from scratch based on already available XML Schemas. The intention is to create generated ontologies automatically based on any possible XML Schemas of an underlying domain data model using XSLT transformations. Initially defined generated ontologies are linked to an ontology of the appropriate domain used to specify supplementary semantic information not covered in the XML Schemas. Domain experts enrich the domain ontology with additional semantics needed for tasks typically performed in the particular domain.

2 Designing Domain Ontologies based on XML Schemas

Figure 1 sketches the devised underlying concept of the generic multilevel approach for designing domain ontologies based on XML Schemas.

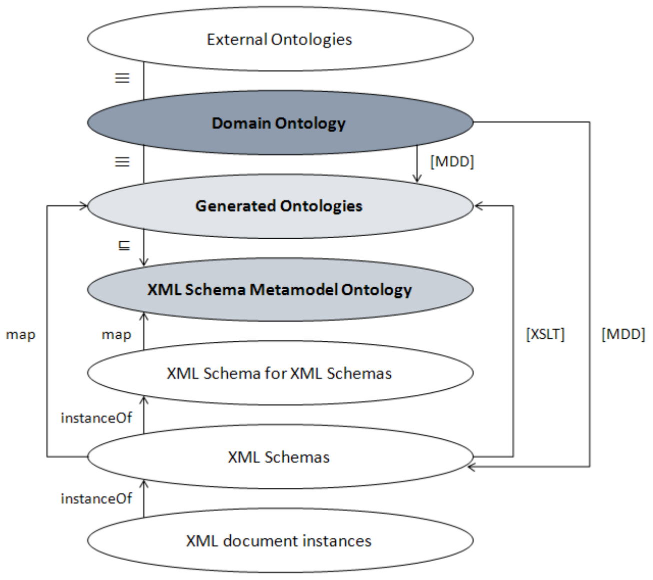

Fig. 1. Generic multilevel approach for designing domain ontologies based on XML Schemas

XSLT transformations map any XML Schemas to generated ontologies automatically. The XML Schema Metamodel Ontology serves as a basis for this process. Domain ontologies are related to generated ontologies in order to append semantic information not expressed in the XML Schemas. Further, you may integrate external ontologies' semantics. The relationships between the separate levels of XML and between the distinct ontologies are delineated. You can derive generated ontologies and corresponding XML Schemas simultaneously, automatically and model-driven from the data model of the domain ontologies. The ensuing paragraphs present the different levels of XML, the individual ontologies and their relationships in more detail.

2.1 XML Schema and the XML Schema Metamodel Ontology

XML [4] documents are commonly used to store and transfer information in distributed environments. XML documents may be instances of XML Schemas [5] determining their terminology and syntactic structure. The W3C has defined XML Schema, the class of XML documents, recursively using the XML Schema language to describe the XML Schema language [6], just like XML Schema documents are XML documents describing XML documents. Generated ontologies are based on the components of the XML Schema abstract data model, the meta-model of XML Schema. Table 1 outlines the mappings between the XML Schema meta-model and the XML Schema Metamodel Ontology. In order to visualize OWL language constructs, Description Logic syntax is used.

Table 1. Mapping of the XML Schema meta-model to the XML Schema Metamodel Ontology

XML Schema for XML Schemas	XML Schema Metamodel Ontology
meta-element information items	classes: <meta-element information item>
attributes of meta-element information items	datatype properties and associated universal restrictions: <domain meta-element information item> ⊑ ∀<attribute>_<domain meta-element information item>_String.String
texts contained in meta-element information items	datatype property and associated universal restriction: <domain meta-element information item> ⊑ ∀valueXSD_<domain meta-element information item>_String.String
texts contained in XML document instances' components	datatype property and associated universal restriction: <domain meta-element information item> ⊑ ∀valueXML_<domain meta-element information item>_String.String
attributes of meta-element information items referring to meta-element information items	object properties and associated universal restrictions: <domain meta-element information item> ⊑ ∀<attribute>_<domain meta-element information item>_<range meta-element information item>.<range meta-element information item>
attributes 'type' and 'base'	object properties and associated universal

	restrictions: <domain meta-element information item> ⊑ ∀type|base_<domain meta-element information item>_Type.Type
meta-element information items' part-of relationships	object properties and associated universal restrictions: <domain meta-element information item> ⊑ ∀contains_<domain meta-element information item>_<range meta-element information item>.<range meta-element information item>

The authors have mapped the meta-element information items corresponding to the XML Schema abstract data model components (e.g. 'element') directly to classes of the XML Schema Metamodel Ontology (e.g. 'Element'). Attributes of meta-element information items have been mapped to datatype properties '<attribute>_<domain meta-element information item>_String' (e.g. 'name_Element_String') with the classes representing the meta-element information items as domains and the built-in primitive datatype 'string' as range. Universal restrictions on datatype properties have been defined, since all range individuals of these datatype properties have to be of the primitive datatype 'string': <domain meta-element information item> ⊑ ∀ <attribute>_<domain meta-element information item>_String.String (e.g. Element ⊑ ∀ name_Element_String.String). As XML Schemas' components can not only have child elements as content, but also plain text, the datatype property 'valueXSD_<domain meta-element information item>_String' (e.g. 'valueXSD_Documentation_String') and the associated universal restriction <domain meta-element information item> ⊑ ∀ valueXSD_<domain meta-element information item>_String.String (e.g. Documentation ⊑ ∀ valueXSD_Documentation_String. String) have been added, since the class representing the meta-element information item is a sub-class of the anonymous super-class of all the individuals which have only relationships along this datatype property to individuals of the class 'String' corresponding to the built-in datatype 'string' or have no relationships along this datatype property. The XML Schema Metamodel Ontology includes the datatype property 'valueXML_<domain meta-element information item>_String' and the corresponding universal restriction <domain meta-element information item> ⊑ ∀ valueXML_<domain meta-element information item>_String.String, since XML document instances' components may contain text. Considering the OWL assertional knowledge, the XML document fragment <VariableName ... lang="en">EF1 </VariableName> is mapped to the property assertions valueXML_Attribute_String (Lang-Individual, 'en') and valueXML_Element_String (VariableName-Individual, 'EF1'). Attributes of meta-element information items such as 'ref' referring to meta-element information items, have been transferred to object properties '<attribute>_<domain meta-element information item>_<range meta-element information item>' with corresponding universal restrictions <domain meta-element information item> ⊑ ∀ <attribute>_<domain meta-element information item>_<range meta-element information item>.<range meta-element information item> (e.g. Attribute ⊑ ∀ ref_Attribute_Attribute.Attribute). Diverse meta-element information items include the attributes ‘type’ or ‘base’. These attributes may have simple ur-type, simple type or complex type definitions as possible attribute values. According to the specified naming conventions, each attribute would be transformed into three different object properties with the ranges ‘AnySimpleType’, ‘SimpleType’ and ‘ComplexType’. XSLT transformations, building generated ontologies automatically based on XML Schemas, would have to determine the range of the object properties belonging to specific ‘type’ and ‘base’ attributes at runtime. It is complicated and error-prone to determine if type references either point to simple or complex type definitions which are part of external XML Schemas’ namespaces. During the transformation process, XML Schemas with appropriate target namespaces have to be available and it has to be iterated over each simple and complex type. Due to these reasons, the authors have decided to map the attributes ‘type’ and ‘base’ to the object properties ‘type|base_<domain meta-element information item>_Type’ with the class ‘Type’ as range. ‘Type’ represents the super-class of all three possible type definitions. As a consequence, each specific type definition can be in the range of the ‘type’ and ‘base’ object properties. Universal restrictions on these object properties have been specified as well: <domain meta-element information item> ⊑ ∀type|base_<domain meta-element information item>_Type.Type (e.g. Element ⊑ ∀ type_Element_Type.Type). Part-of relationships to child meta-element information items as content of meta-element information items have been transferred to object properties ‘contains_<domain meta-element information item>_<range meta-element information item>’. Universal restrictions on each object property have been defined, because the range of relationships along these object properties is assumed as fixed: <domain meta-element information item> ⊑ ∀ contains_<domain meta-element information item>_<range meta-element information item>.<range meta-element information item> (e.g. ComplexType ⊑ ∀ contains_ComplexType_SimpleContent. SimpleContent).

2.2 Generated Ontologies

Executing an XSLT script, the declarations and definitions of any XML Schemas are transformed into classes of generated ontologies directly and automatically. As all components of the normative XML Schema for XML Schemas are included in the XML Schema Metamodel Ontology, this works with all valid XML Schemas. The mapping process takes seconds and requires no human interaction. The generated ontologies’ classes are defined as sub-classes of the XML Schema Metamodel Ontology. Hence, all generated ontologies are based on the same reusable classes. Like heavyweight ontologies [7], the generated ontologies consist of a hierarchy of classes as well as relations with domains and ranges. Moreover, the generated ontologies include universal restrictions on object properties, hasValue restrictions on datatype properties and complex classes consisting of the union of multiple classes’ individuals, if universal restrictions on object properties have more than one class in the range. Table 2 depicts the mappings between XML Schemas and the generated ontologies.

Table 2. Mapping of XML Schemas to generated ontologies

XML Schemas	Generated Ontologies
element information items	sub-classes of XML Schema Metamodel Ontology's classes: <element information item> ⊑ <meta-element information item>
values of element information items' attributes	hasValue restrictions on XML Schema Metamodel Ontology's datatype properties: <element information item> ⊑ ∃<attribute>_<domain meta-element information item>_String.{<String>}
texts contained in element information items	hasValue restrictions on XML Schema Metamodel Ontology's datatype properties: <element information item> ⊑ ∃valueXSD_<domain meta-element information item>_String.{<String>}
values of element information items' attributes referring to other element information items	universal restrictions on XML Schema Metamodel Ontology's object properties: <domain element information item> ⊑ ∀<attribute>_<domain meta-element information item>_<range meta-element information item>.<range element information item>
values of attributes 'type' and 'base'	universal restrictions on XML Schema Metamodel Ontology's object properties: <domain element information item> ⊑ ∀type|base_<domain meta-element information item>_Type.<range element information item>
element information items' part-of relationships	universal restrictions on XML Schema Metamodel Ontology's object properties: <domain element information item> ⊑ ∀contains_<domain meta-element information item>_<range meta-element information item>.<union of range element information items>

XML Schemas' element information items are mapped to sub-classes of the XML Schema Metamodel Ontology's classes: <element information item> ⊑ <meta element information item>. The element information item 'element' with the assigned name 'VariableName' (<xs:element name="VariableName" ... />), for example, is tranferred to the class 'VariableName' with 'Element' as super-class (VariableName ⊑ Element), since all 'VariableName' individuals are also part of the 'Element' class extension. Values of element information items' attributes are transformed into hasValue restrictions on the XML Schema Metamodel Ontology's datatype properties

<element information item> ⊑ ∃ <attribute>_<domain meta-element information item>_String.{<String>}, as the element information item is the sub-class of the anonymous super-class of all the individuals which have at least one relationship along the datatype property '<attribute>_<domain meta-element information item>_String' to the specified individual of the primitive datatype 'string'. For instance, the value of the attribute 'name' of the element information item 'element' (<xs:element name="VariableName" ... />) is converted to the datatype property hasValue restriction VariableName ⊑ ∃ name_Element_String.{'VariableName'}, since each element 'VariableName' has at least one associated name, namely 'VariableName'. Texts contained in element information items are mapped to hasValue restrictions on the XML Schema Metamodel Ontology's datatype property <element information item> ⊑ ∃ valueXSD_<domain meta-element information item>_String.{<String>}. For example, the text included in the element information item 'documentation' (<xs:documentation>Indicates the language of content.</xs:documentation>) is translated into the datatype property hasValue restriction Documentation1 ⊑ ∃ valueXSD_Documentation_String. {'Indicates the language of content.'}. As element information items may contain more than one element information item of the same meta-element information item, the contained element information items' identifiers are sequential (e.g. Documentation1). Values of element information items' attributes referring to other element information items are converted to universal restrictions on the XML Schema Metamodel Ontology's object properties <domain element information item> ⊑ ∀ <attribute>_<domain meta-element information item>_<range meta-element information item>.<range element information item>. The reference to the element information item 'attribute' called 'lang' (<xs:attribute ref="lang"/>) is transformed into the object property universal restriction Lang-Reference ⊑ ∀ ref_Attribute_Attribute.Lang. The values of the attributes 'type' and 'base' are transferred to universal restrictions on XML Schema Metamodel Ontology's object properties: <domain element information item> ⊑ ∀ type|base_<domain meta-element information item>_Type.<range element information item>. The attribute 'type' of the element information item 'element' named 'VariableName' (<xs:element name="VariableName" type="NameType"/>), for example, is converted to the object property's universal restriction VariableName ⊑ ∀ type_Element_Type.NameType. Element information items' part-of relationships are realized by universal restrictions on XML Schema Metamodel Ontology's object properties <domain element information item> ⊑ ∀ contains_<domain meta-element information item>_<range meta-element information item>.<union of range element information items>. The complex type definition 'InternationalStringType' includes only one 'simpleContent' element information item (<xs:complexType name="InternationalStringType"> ...<xs:simpleContent>...</xs:simpleContent></xs:complexType>). As a consequence, the range of the object property can only consist of individuals of one class (InternationalStringType ⊑ ∀ contains_ComplexType_SimpleContent.Simple Content1). If element information items like 'extension' have more than one element information item as content (e.g. <xs:extension...><xs:attribute name="translated"...> ...</xs:attribute><xs:attribute name="translatable"..>..</xs:attribute></xs:extension>), the domain element information items can only have relationships along the object

property to individuals of the complex class consisting of the union of individuals of multiple classes representing the contained range element information items (Extension1 $\sqsubseteq \forall$ contains_Extension_Attribute.(Translated $\sqcup$ Translatable)).

2.3 Domain Ontologies and Integration of Other Ontologies

In domain ontologies, the semantics of classes are specified as exactly as needed using formal logic [7]. Each data model of a specific domain can be expressed in the form of a domain ontology. Classes of any number of generated ontologies of a given domain can be annotated as equivalent to classes of the domain ontology (<domain ontology class> ≡ <generated ontology class>). Thus, the information of a particular domain stored in generated ontologies and in corresponding XML Schemas can be reused during early stages of the domain ontology design process. Ontology engineers can add further domain specific semantic information to the domain ontology subsequently in a continuous way. You can perform queries on domain ontologies using the semantics of the particular domain without knowledge of complex XML Schemas' structures. Requests on domain ontologies are propagated to the underlying generated ontology or ontologies (if the domain data model consists of more than one XML Schema) via equivalence relationships. Hence, there is no need to query each associated generated ontology individually using different classes, object and datatype properties. Classes of domain ontologies can be annotated as being equivalent to existing similar and widely adopted classes of external ontologies (<domain ontology class> ≡ <external ontology class>; other types of relationships are also possible). Due to this, reasoners may use additional semantic information defined in external ontologies for deductions [8].

3 Related Work

The XML Schema Metamodel Ontology, although much more complex, corresponds to the general database ontology designed by Kupfer et al. [8]. These ontology engineers have defined a database schema-to-ontology mapping, which means that specific database ontologies are generated automatically from any database schemas. Kupfer et al. have specified the conceptual model of the general database ontology as follows: databases can consist of multiple tables and tables can comprise diverse attributes. The authors used the three classes 'Database', 'Table', and 'Attribute' as well as the object property 'consistsOf' to describe database schemas. The classes' identifiers serve as links to all tables and attributes of the underlying database schemas. Kupfer et al. depicted domain ontologies in the context of the developed general database ontology. Using domain ontologies, semantic information about specific domains is annotated and added supplementary to database ontologies. The relation between database ontologies' classes and classes of domain ontologies has been conceptualized using the object property 'containsDataAbout'.
Several strategies lifting the syntactic level of XML documents to the semantic level of OWL ontologies can be distinguished. The authors have clustered appropriate tools

implementing these transformations into three classes depending on the kind of conversion either at the instance, the conceptual, or both the instance and the conceptual level. At the instance level, Klein has developed the so-called RDF Schema mapping ontology enabling a one-way mapping of XML documents to RDF. Relevant XML documents' content can be identified [9]. Extending this approach, Battle has introduced a bidirectional mapping of XML components to RDF [10]. The WEESA system implements an automatic transformation from XML to RDF using an OWL ontology, manually created from corresponding XML Schemas and manually defined rules. XML document instances are not mapped to OWL equivalents [11]. O'Connor and Das developed an approach transforming XML documents to individuals of an OWL ontology describing the serialization of the XML document. SWRL [12] is used to map these instances to individuals of a domain ontology [13].
At the conceptual level you can distinguish between approaches converting XML schema languages to RDFS or OWL. Several languages for writing schemas like DTD [4], XML Schema [5], DSD [14] and Relax NG [15] exist. The prototype OntoLiFT [16] offers a generic means for converting arbitrary XML schema languages to RDFS ontologies semi-automatically. In a first step, XML schema languages are transformed into regular tree grammars consisting of non-terminals, terminals, start symbols and production rules [17]. In a second step, non-terminals as well as terminals are converted to RDFS classes and production rules are mapped to RDF properties. In comparison with our approach, OntoLiFt converts any XML schema language and not just XML Schema to ontologies. Anicic et al. evolved an approach based on meta-models transforming between the different models of XML Schema and OWL [18].
At the instance and the conceptual level, there are methods transforming XML to RDF and XML Schema to either RDFS or OWL. Within the EU-funded project called 'Harmonise' the interoperability of existing standards for the exchange of tourism data has been achieved by the transformation of XML documents and XML Schemas into RDF and RDFS ontologies which have been mapped to each other [19]. Using the approach of O'Connor and Das [20], XML document instances are transformed to OWL ontologies even though associated XML Schemas not exist. As a consequence, unstructured contents can be mapped to OWL ontologies as well. XML Schemas can also be mapped to OWL ontologies, as XML Schema documents are represented in XML, too. New OWL ontologies can be generated from scratch and existing ones can be extended. O'Connor and Das evolved XML Master, a language describing OWL ontologies declaratively. XML Master combines the Manchester OWL Syntax [21] and XPath [22] to refer to XML content. O'Connor and Das criticize the limited and unsatisfactory number of OWL constructs supported by current tools converting XML Schemas to OWL ontologies. Thus, all OWL constructs are covered. One shortcoming associated with this method is that you have to write mapping language expressions manually and therefore you cannot transform XML documents and XML Schemas to OWL ontologies automatically. Another drawback is that ontology engineers have to be familiar with the Manchester OWL Syntax and XPath in order to express the mappings. Ferdinand et al. propose both mappings from XML to RDF and XML Schema to OWL which are independent of each other. This means, OWL individuals do not necessarily correspond to the OWL conceptual model, since XML documents' declarations and definitions may be transferred to differing OWL

constructs [23]. In addition, another system can be stated transferring XML Schema components to OWL language constructs at the terminological level and XML document instances to OWL individuals at the assertional level. XPath expressions are applied selecting XML documents' content [24]. Besides that, the approach of Tous et al. is very similar to this method [25]. The authors of [26] devised a mapping between XML and RDF and between XML Schema and OWL .The authors assume that XML documents are structured like relational databases. Thus, XML documents' relational structures are discovered and represented in OWL. Relations correspond to classes, columns to properties, and rows to instances. XML data model elements are mapped automatically to components of the OWL data model. Named simple and complex types, for instance, are transferred to classes. Elements, containing other elements or having at least one attribute, are converted to classes and object properties between these classes. Both elements, including neither attributes nor sub-elements, and attributes, assumed to represent database columns, are transformed into datatype properties with the surrounding element as their domain. Also, XML cardinality contraints are transformed into equivalent OWL cardinality restrictions.
Many approaches try to extrac semantics from XML Schemas. The suggested approach, in contrast, only gains information about the syntactic structure of XML document instances contained in XML Schemas. Generated ontologies are connected with domain ontologies which are enriched with semantic domain specific information in a further step. The majority of the tools attempt to convert either schemas to ontologies at the conceptual level or XML to RDF at the instance level. The method, presented in this paper, follows a complete approach transforming XML document instances' content to OWL individuals as well as XML Schemas to OWL. In comparison with our approach, many others transform XML to RDF and/or XML schema languages to ontologies in a manual or at most in a semi-automatic and not automatic manner. Furthermore, diverse existent methods generate RDFS ontologies and not the more expressive OWL ontologies.

4 Conclusion

The aim of this paper is to bridge the gap between XML and OWL. XML Schema and OWL ontologies follow differing modeling goals. While XML Schema describes the syntactic structure of XML document instances, OWL is based on formal logic and describes the semantics of data models. In this paper, the authors demonstrated a generic multilevel approach designing domain ontologies when XML Schemas are provided as input sources for the ontology design process. This process of designing ontologies from scratch requires considerable effort. The normal procedure of ontology engineers is specifying the domain ontologies' semantics in collaboration with domain experts already at the beginning of the process. Applying our approach, ontology engineers are allowed to pursue a different path. They can rely on existent information located in XML Schemas of a given domain data model. To realize this, generated ontologies are built in an automatic way based on already available XML Schemas. Therefore, time-consuming work is already done and can be reused by the ontology engineers who do not have to define the domain data model anew. The generated ontologies' classes are based on super-classes of the XML Schema Metamodel Ontology. This ontology consists of classes representing the components of the XML Schema abstract domain model and the corresponding element information items. The components of the XML Schema abstract data model are used to describe XML Schemas recursively using XML Schema language constructs. Based on interviews with domain experts, the information stored in the generated ontologies can be extended in a continuous manner. This supplemental semantic domain specific information is defined in domain ontologies whose classes are linked to the generated ontologies' classes via equivalence relationships. Domain ontologies' classes can be annotated as equivalent to classes of widely adopted external ontologies. As a consequence, reasoners may use additional semantics for deductions.

5 Future Work

A complete use case designing a specific domain ontology using the devised multilevel approach based on already existing XML Schemas will be described in detail. The underlying data model of the application domain is called the Data Documentation Initiative (DDI) [27]. DDI in its current version 3 is an international standard for describing data from the social, behavioral and economic sciences. Furthermore, more use cases from different domains will be shown to prove the generality of the developed approach. The main benefit associated with this approach, saving time for ontology engineers in the process of designing domain ontologies from scratch, will be evaluated as well.
The authors will develop an XSLT framework to implement a complete stylesheet-driven approach to generate OWL ontologies. So far, XSLT transformations build generated ontologies automatically based on arbitrary XML Schemas. Moreover, the authors will write XSLT transformations, converting XML documents without corresponding XML Schemas determining their syntactic structure to generated ontologies. The first step is creating suitable XML Schemas out of XML document instances automatically. These XML Schemas will then be converted to generated ontologies in a second step. Another XSLT stylesheet will convert XML document instances' data to OWL instances according to the generated ontologies. Generated ontologies and corresponding XML Schemas will be derived automaticly from designed domain ontologies using XSLT transformations. These scripts will be evolved realizing the model-driven development of generated ontologies and underlying XML Schemas associated with the domain ontologies.

References

1. Linked Data, http://linkeddata.org
2. The XML data model, http://www.w3.org/XML/Datamodel.html
3. Resource Description Framework (RDF): concepts and abstract syntax, http://www.w3.org/TR/2002/WD-rdf-concepts-20021108/

4. Extensible Markup Language (XML) 1.0 (fifth edition) - W3C recommendation 26 November 2008, http://www.w3.org/TR/2008/REC-xml-20081126/
5. XML Schema part 0: primer second edition - W3C recommendation 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/
6. XML Schema part 1: structures second edition - W3C recommendation 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/
7. Stuckenschmidt, H.: Ontologien: Konzepte, Technologien und Anwendungen. Springer-Verlag, Berlin Heidelberg (2009)
8. Kupfer, A., Eckstein, S., Störmann B., Neumann K., Mathiak B.: Methods for a synchronised evolution of databases and associated ontologies. In: Proceeding of the 2007 Conference on Databases and Information Systems IV. (2007)
9. Klein, M.C.A.: Interpreting XML documents via an RDF Schema ontology. In: 13th International Workshop on Database and Expert Systems Applications, Aix-en-Provence (2002)
10. Battle, S.: Gloze: XML to RDF and back again. In: 1st Jena User Conference, Bristol (2006)
11. Reif, G., Gall, H, Jazayeri, M.: WEESA - web engineering for Semantic Web applications. In: 14th World Wide Web Conference, Chiba (2005)
12. SWRL: a Semantic Web Rule Language combining OWL and RuleML, http://www.w3.org/Submission/SWRL/
13. O'Connor, M.J., Das, A.K.: Semantic reasoning with XML-based biomedical information models. In: 13th World Congress on Medical Informatics, Cape Town (2010)
14. Karlund, N., Moller, A., Schwartzbach, M.I.: DSD: a schema language for XML. In: ACM SIGSOFT Workshop on Formal Methods in Software Practice. (2000)
15. Clark, J., Cowan, J., Fitzgerald, M., Kawaguchi, J., Lubell, J., Murata, M., Walsh, N., Webber, D.: Information technology – document schema definition language (DSDL) – part 2: regular-grammar-based validation – RELAX NG. ISO/IEC 19757-2:2003(E). (2003)
16. Volz, R., Oberle, D., Staab, S., Studer R.: OntoLiFT Prototype – WonderWeb: ontology infrastructure for the Semantic Web. Karlsruhe (2003)
17. Murata, M., Lee, D., Mani, M., Kawaguchi, K.: Taxonomy of XML schema languages using formal language theory. In: ACM Transactions on Internet Technology. vol. 5, New York (2005)
18. Anicic, N., Ivezic, N., Marjanovic, Z.: Mapping XML Schema to OWL. In: Enterprise Interoperability, Part V, pp. 243--252, Springer, Berlin (2007)
19. Dell'Erba, M., Fodor, O., Ricci, F., Werthner, H.: Harmonise: a solution for data interoperability. In: Proceedings of the 2nd IFIP Conference on E-Commerce, E-Business, E-Government I3E. (2002)
20. O'Connor, M. J., Das, A. K.: Acquiring OWL ontologies from XML documents. In: Proceedings of the Sixth International Conference on Knowledge Capture, New York (2011)
21. OWL 2 Web Ontology Language Manchester Syntax, http://www.w3.org/TR/owl2-manchester-syntax/
22. XML Path Language (XPath) 2.0 (second edition), http://www.w3.org/TR/xpath20/
23. Ferdinand, M., Zirpins, C., Trastour, D.: Lifting XML Schema to OWL. In: Web Engineering - 4th International Conference, Munich (2004)
24. Kobeissy, N., Genet, M.G., Zeghlache, D.: Mapping XML to OWL for seamless information retrieval in context-aware environments. In: International Conference on Pervasive Services, Istanbul (2007)
25. Tous, R., Garcia, R., Rodriguez, E., Delgado, J.: Architecture of a semantic XPath processor. Application to digital rights management. In: 6th E-Commerce and Web Technologies, Copenhagen (2005)
26. Bohring, H., Auer, S.: Mapping XML to OWL Ontologies. In: Leipziger Informatik Tage, vol. 72, Leipzig (2005)
27. Data Documentation Initiative, http://www.ddialliance.org