Reusing XML Schemas' Information as a Foundation for Designing Domain Ontologies

Thomas Bosch

GESIS - Leibniz Institute for the Social Sciences, Mannheim, Germany
thomas.bosch@gesis.org

Abstract. Designing domain ontologies from scratch is a time-consuming endeavor requiring a lot of close collaboration with domain experts. However, domain descriptions such as XML Schemas are often available in early stages of the ontology development process. For my dissertation, I propose a method to convert XML Schemas to OWL ontologies in an automatic way. The approach addresses the transformation of any XML Schema documents by using the XML Schema metamodel, which is completely represented by the XML Schema Metamodel Ontology. Automatically, all Schema declarations and definitions are converted to class axioms, which are intended to be enriched with additional domain-specific semantic information in form of domain ontologies.

Keywords: Semantic Web, Ontology Design, OWL, XML, XSD, XSLT.

1 Problem and Research Question

XML represents a large set of information within the context of various domains and has reached wide acceptance as standard data exchange format. Traditionally, ontology engineers work in close collaboration with domain experts to design domain ontologies manually, which requires a lot of time and manpower. Domain ontologies as well as XSDs describe domain data models. In many cases, XSDs are already existent and can therefore be reused in the process designing domain ontologies from scratch. As a consequence, saved time and effort could be used more effectively to enrich data models with supplementary domain-specific semantic information, not or not satisfyingly covered by the underlying XSDs. The main research question, how the time-consuming process designing domain ontologies based on already available XSDs could be accelerated, results from the stated problem.

2 Proposed Approach

Concept. Bosch and Mathiak have developed the concept of the generic multilevel approach to design domain ontologies based on already available XSDs [1]. XSDs determine the terminology, the vocabulary and the syntactic structure of XML document instances. XSDs are instances of the XSD metamodel. The components of the

P. Cudré-Mauroux et al. (Eds.): ISWC 2012, Part II, LNCS 7650, pp. 437–440, 2012.

XSD abstract data model, also called element information items (EIIs) in the XML representation, are mapped to classes, universal restrictions on datatype and object properties of a generic ontology, the XML Schema Metamodel Ontology (XSDMO). The idea of the developed approach is to convert any XSDs automatically to classes, hasValue restrictions on XSDMO's datatype properties and universal restrictions on XSDMO's object properties using XSLT (Bosch and Mathiak explain implementation details in [2]). Any XSD can be transformed into a generated ontology, since each component of the XSD abstract data model is covered by this approach. On the instance level, XML document instances are translated into an RDF representation of the generated ontologies by means of a Java program as XSLT is less powerful for this purpose. After these two transformation processes, which take only seconds, all the information located in the XSDs is re-used in generated ontologies and their RDF representations can now be published in the LOD cloud and be linked to other RDF datasets. Generated ontologies are not directly as useful as manually created domain ontologies, as XSD and OWL follow different modeling goals, since generated ontologies' structures are rather complex, and as generated ontologies are not conform to the highest quality requirements of domain ontologies. Therefore, the generated ontologies' class axioms are intended to be further supplemented with additional domain-specific semantic information, not specified in underlying XSDs, in form of domain ontologies. These domain ontologies can be derived automatically out of the generated ontologies using SWLR rules on the schema as well as on the instance level. Consequentially, all XML data conforming to XSDs can be imported automatically as domain ontologies' instances. The effort and the time, however, delivering high quality domain ontologies subsequently is much less than creating domain ontologies completely manual and could be used more effectively to expand the XSDs' domain knowledge.

Related Work. The XSDMO corresponds to the general database ontology designed by Kupfer et al. [3]. They have defined a schema-to-ontology mapping: database ontologies are generated automatically from database schemas. Semantic domain-specific information is added supplementary to database ontologies in form of domain ontologies.

Motivation and Use Case. Bosch et al. delineate the DDI ontology [4], whose derivation serves as complete, intuitive, and representative use case to motivate the approach's application. The Data Documentation Initiative (DDI) is an acknowledged international standard for the documentation and management of data from the social, behavioral, and economic sciences. Excerpts of the DDI ontology are derived out of the underlying XSDs describing the statistics domain. DDI XML documents include XML elements 'Question' containing 'QuestionText' elements, which may comprise plain text such as 'How old are you?'. The element 'Question' is an instance of the XSD EII 'element' whose 'name' and 'type' attributes have the values 'Question' and 'QuestionType'. The complex type 'QuestionType' includes the EII 'complexContent' containing the EII 'extension' which comprises a sequence. This sequence contains a reference to the global element 'QuestionText', the type of the XML element 'QuestionText'. 'QuestionText' includes the text 'How old are you?' which if of the XSD's primitive datatype string.

XSD's EIIs are converted to generated ontology's classes which are defined as sub-classes of XSDMO' super-classes: <EII> ⊑ <meta-EII>. The global element 'QuestionText' (<xs:element name="QuestionText"/>), for example, is translated into the class 'QuestionText-Element…' which is specified as sub-class of the super-class 'Element' (<QuestionText-Element…> ⊑ <Element>), as each particular EII 'element' is also part of the 'Element' class extension. EIIs' attributes' values are converted to XSDMO's datatype properties '<attribute>_<domain meta-EII>_String' and to hasValue restrictions on these datatype properties: <domain EII> ⊑ ∃ <attribute>_<domain meta-EII>_String.{<String>}. The value 'Question' of the 'element' EII's attribute 'name' (<xs:element name="Question"/>) is translated into the datatype property 'name_Element_String' and into the datatype property's universal restriction Question-Element… ⊑ ∃ name_Element_String.{'Question'}, since 'Question-Element…' resources must have at least one relationship along the datatype property 'name_Element_String' to the string individual 'Question'. EIIs' attributes' values referring to other EIIs are transformed into XSDMO's object properties' universal restrictions <domain EII> ⊑ ∀ <ref|substitutionGroup|refer>_<domain meta-EII>_<range meta-EII>.<range EII>. The value 'QuestionText' of the 'element' EII's attribute 'ref' (<xs:element ref="QuestionText"/>) referring to the EII 'element' with the name 'QuestionText' is translated into the object property's universal restriction QuestionText-Element-Reference… ⊑ ∀ ref_Element_Element.Question Text-Element…, as 'QuestionText-Element-Reference…' instances can only have 'ref_Element_Element' relationships to 'QuestionText-Element…' resources or have no such relations. Values of EIIs' attributes referring to type definitions are translated into universal restrictions on XSDMO's object properties <domain EII> ⊑ ∀ type|base_<domain meta-EII>_Type.<range EII>. The value 'QuestionType' of the attribute 'type' of the 'Question' EII 'element' (<xs:element name="Question" type = "QuestionType"/>) is converted to the object property's universal restriction Question-Element… ⊑ ∀ type_Element_Type.QuestionType-Type…. The part-of relationship of the EII 'sequence' (<sequence><element ref="QuestionText"/></sequence>) is translated into the object property's universal restriction Sequence… ⊑ ∀ contains_Sequence_Element.QuestionText-Element-Reference…. The strict order of the in the sequence contained EIIs is expressed by the object property's universal restriction Sequence… ⊑ ∀ sequence.QuestionText-Element-Reference…. As resources of the class 'QuestionText-Element…' may have text as content, the datatype property 'value_Element_String' is introduced and the datatype property's universal restriction QuestionText-Element… ⊑ ∀ value_Element_String.String is defined.

We want to derive that the 'Question-Element…' resource 'age' is also of the type 'Question' with the question text 'How old are you?'. The following program fragment demonstrates the antecedent and the consequent of the SWRL rule, executed by a rule engine to derive the two statements: `(?a type_Element_Type ?b) ∧ (?b contains_ComplexType_ComplexContent ?c) ∧ … ∧ (?g rdf:type QuestionText-Element…) ∧ (?g value_Element_String ?h) -> (?a rdf:type Question) ∧ (?a questionText ?h)` The two statements can be derived since the individual 'age', substituting the SWRL variable '?a', has a relationship along 'type_Element_Type' to

an individual replacing the variable ‘b’. This resource is linked to an instance ‘?c’ via ‘contains_ComplexType_ComplexContent’. Further, there’s a navigation path from the ‘?c’ individual to the ‘?g’ instance along the stated properties. As XML elements ‘QuestionText’ may contain text nodes like ‘How old are you?’, the ‘?g’ instance is assigned to the class ‘QuestionText-Element…’ ensuring that derived question texts are only strings contained in ‘QuestionText-Element…’ resources. The ‘?g’ resource must have a ‘value_Element_String’ relation to a ‘?h’ individual. As the instances ‘age’ and ‘How old are you?’ correspond to the SWRL rule’s antecedent, it can be inferred that the resource ‘age’ is a question with the question text ‘How old are you?’.

3 Results and Future Work

The approach’s concept has been finalized and the mapping of the XSD metamodel to the XSDMO has been defined and implemented. The mapping between XSDs and generated ontologies has been specified and programmatically realized. Also the generality of the approach has been verified, since the generic test cases have shown that all meta-EIIs of the XSD metamodel are covered and thus each XSD can be transformed into a generated ontology using the same transformation rules.

Currently, I’m writing a Java program translating XML documents into RDF representations of the generated ontologies. So far, the most relevant subsets of the DDI domain ontology are derived and appropriate SWRL rules are defined. To verify the hypothesis that the effort and the time delivering high quality domain ontologies using the developed approach is much less than creating domain ontologies manually, the traditional manual and the proposed semi-automatic approach will be compared by means of a user study. For an extensive evaluation of the work, it is absolutely essential to create generated ontologies and to deduce domain ontologies out of XSDs of multiple and differing domains.

References

1. Bosch, T., Mathiak, B.: Generic Multilevel Approach Designing Domain Ontologies Based on XSDs. In: Proceedings of the Workshop Ontologies Come of Age in the Semantic Web, 10th International Semantic Web Conference, pp. 1-12. CEUR Workshop Proceedings, Aachen (2011)
2. Bosch, T., Mathiak, B.: XSLT Transformation Generating OWL Ontologies Automatically Based on XSDs. In: IEEE Xplore Digital Library, 6th International Conference for Internet Technology and Secured Transactions, pp. 660–667. IEEE Xplore Digital Library (2012)
3. 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)
4. Bosch, T., Cyganiak, R., Wackerow, J., Zapilko, B.: Leveraging the DDI Model for Linked Statistical Data in the Social, Behavioural, and Economic Sciences. In: Proceedings of the International Conference on Dublin Core and Metadata Applications, pp. 46–55 (2012)