XSLT Transformation Generating OWL Ontologies Automatically Based on XML Schemas

Thomas Bosch, Brigitte Mathiak
Monitoring Society and Social Change
GESIS – Leibniz Institute for the Social Sciences
Mannheim, Germany
Thomas.Bosch@gesis.org, Brigitte.Mathiak@gesis.org

Abstract—**Designing domain ontologies from scratch is a time-consuming process. In many cases, both the terminologies and the syntactic structures of domain data models are already described in form of XML Schemas. XSLT transformations are used to lift the syntactic level of XML documents to the semantic level of OWL ontologies by mapping any XML Schemas to generated ontologies automatically. Ontology engineers base domain ontologies on generated ontologies to enrich the information located in the XML schemas with additional domain specific semantic information. The aim of this paper is to show the implementation of the general approach transforming any XML Schemas into generated ontologies automatically using XSLT.**

Keywords-XSLT, OWL, XML Schema

I. Introduction

XML [1] documents are commonly used to store and transfer information in distributed environments. XML documents may be instances of XML Schemas [2] determining their terminology and syntactic structure. XML represents a large set of information within the context of various domains and has reached wide acceptance as standard data exchange format in e-business. This objective fact has driven the development of general-purpose tools for converting XML Schemas to OWL ontologies. Both data and metadata, structured by ontologies, can be published in the increasingly popular and widely adopted LOD cloud to get linked with a huge number of other RDF datasets [3]. 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.

XML Schema and OWL follow differing modeling goals. On the one hand, the XML data model describes the terminology and the syntactic structure of XML documents, a node labeled tree [4]. OWL, otherwise, is based on formal logic and on the subject-predicate-object triples from RDF [5]. OWL specifies semantic information about specific domains of interest, describes relations between domain classes and thus allows the sharing of conceptualizations. An effective and efficient cooperation between e-business partners is only possible if they agree on a common syntax (specified by XML Schemas) and have a common understanding of the domain classes (defined by OWL ontologies). The authors of this paper attempt to bridge the gap between XML Schema and OWL by lifting the syntactic level of XML documents to the semantic level of OWL ontologies.

II. Problem

The process developing domain ontologies is very time-consuming. XML Schemas describing specific domains are often existent in early stages of the ontology design process. The traditional procedure is to transfer the information located in XML Schemas of certain data models to OWL ontologies manually, which requires a lot of time. Saved time could be used more effective to extend the knowledge expressed in the XML Schemas in order to enrich the data models with supplementary domain specific information. In this paper, the authors describe the implementation of a generic multilevel approach accelerating the process designing domain ontologies from scratch based on already available XML Schemas. The intention of this approach is to create generated ontologies completely automatically based on any XML Schemas, expressing domain data models, using XSLT transformations. The direct mapping from XML and XML Schema to RDF and OWL is not sufficient, since it only transports information about the terminology and the syntactic structure of XML document instances. Semantic information has to be added in a further step. Initially generated ontologies are connected to domain ontologies using equivalence relationships. Thus, ontology engineers, collaborating with experts of given domains, enrich domain ontologies with additional semantics not or not satisfyingly covered by the underlying XML Schemas. Domain experts as well as ontology engineers enhance domain ontologies with further semantic information needed for tasks typically performed in particular domains.

Compared with previous general-purpose tools for transforming XML Schemas into OWL ontologies, the novelty of the developed approach is that the translation of XML Schemas into OWL ontologies is based on the XML Schema for XML Schemas, the XML Schema meta-model [6]. As this approach considers each component of the XML Schema abstract data model, unexceptionally any XML Schema can be transformed into an OWL ontology in a totally automatic manner without any manual adaption after the translation process. Most tools try extracting semantics directly out of XML Schemas. The suggested approach, in contrast, only gains information about the terminology and the syntactic structure of XML document instances contained in XML Schemas. Semantic domain specific information is associated to domain ontologies in a next step.

III. Generic Multilevel Approach Designing Domain Ontologies Based on XML Schemas

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

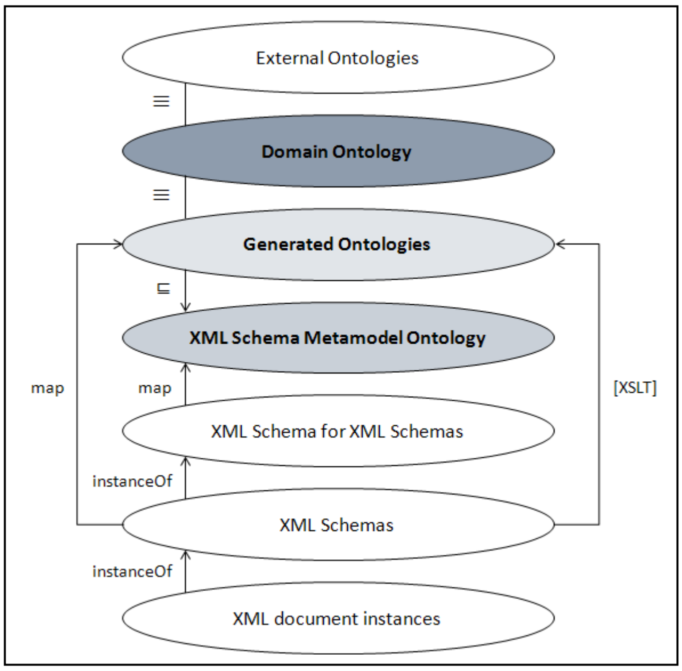

Figure 1. Generic multilevel approach designing domain ontologies based on XML Schemas.

The main goal of this approach is that XSLT transformations map any XML Schemas to generated ontologies automatically without subsequent manual adaptations. The components of the XML Schema abstract data model [6], also called element information items in the XML representation, are mapped directly to classes of the XML Schema Metamodel Ontology. This generic ontology consists of super-classes of the generated ontologies' classes, datatype properties, object properties and universal restrictions on both property types. The generated ontologies' classes correspond to element information items located in the XML Schemas. Generated ontologies and external ontologies are related to domain ontologies via class equivalence relationships. As a consequence, domain ontologies contain all the information located in the XML Schemas of particular domains. After the process translating XML Schemas into generated OWL ontologies, ontology engineers collaborate with domain experts in order to add supplementary semantic information, not expressed in the underlying XML Schemas, to domain ontologies.

IV. Automatic Generation of OWL Ontologies Based on XML Schemas Using XSLT Transformations

A. Definition of RDF Document Header Information

RDF documents consist of a header and an ontology. The RDF document header information, including an XML declaration, an optional document type declaration, an RDF document header, and an OWL document header, is specified by invoicing appropriate templates within the template for the document node representing the input XML Schema. The file extension of the output file is set to '.owl', since OWL documents, which are RDF documents as well, are generated. The XML declaration includes the XML version and the character set, the encoding of the RDF document's content. The document type declaration contains internal entity declarations specifying abbreviations for long namespace URI strings used in the RDF document. The RDF document header defines the default namespace of the RDF document, the base URL, as well as the XML Schema, RDF, RDFS, OWL, and XML Schema Metamodel Ontology's namespace prefixes. The OWL document header includes the statements of the ontology IRI, the ontology version IRI as well as the instruction importing the XML Schema Metamodel Ontology's classes, datatype properties, object properties, and class axioms.

The ontology IRI is set to '<base ontology IRI>/<local ontology IRI>.owl' and is used to identify the ontology in the context of the WWW. The ontology IRI represents the URL where the latest version of the ontology is published. The base ontology IRI is defined as 'http://www.semanticweb.org/ontologies/XMLSchemaOntologies'. The local part of the ontology IRI is determined as follows: The XML Schema's element information items are specified in the target namespace of this XML Schema. If the target namespace of the current XML Schema is stated, this information is used as local ontology IRI. As the target namespace is not a mandatory attribute of XML Schemas' root elements, a user-defined local ontology IRI can also be passed to the translation process as input parameter. Finally, if there is neither a target namespace specified nor a user-defined local ontology IRI delivered, the file name of the input XML Schema serves as local part of the ontology IRI.

The ontology version IRI corresponds to '<base ontology IRI>/<local ontology IRI>:<local ontology version IRI>.owl' and represents the URL where a given version of the ontology is published. Initially, the local part of the ontology version IRI, representing the version of the ontology the input XML Schema is mapped to, is set to '1.0.0'. Minor changes of the generated ontologies and the underlying XML Schemas will cause adjustments of the last two version levels and major ontology changes will cause modifications of the first version stage. After the definition of the RDF document header information and the ontology, the RDF document is closed by calling an appropriate template.

B. Traversing the XML Schema's Document Tree

After the definition of the RDF document header information, the generated ontology representing the input XML Schema is specified by invoicing the template 'ontologyDefinition'. This template serves as starting point to traverse the XML Schema's document tree in order to implement the mappings between the input XML Schema and the generated ontology beginning with the XML Schema's root element 'schema'.

Each element information item located in the input XML Schema is mapped to the generated ontology by calling a template named according to the appropriate XML Schema Metamodel Ontology's super-class representing the corresponding meta-element information item (e.g. 'Schema' to define the class corresponding to the element information item 'schema'). As the element information item 'schema' is always the root element of any XML Schema, the XSLT processor calls the template 'Schema' first.

Element information items may contain other element information items. Part-of-relationships have been converted to object properties ‚contains_<domain meta-element information item>_<range meta-element information item>' in the XML Schema Metamodel Ontology. Containing element information items are in the domain of these object properties, contained element information items in the range. As, for instance, element information items ‚schema' may contain complex type definitions, the object property ‚contains_Schema_ComplexType' has been defined in the XML Schema Metamodel Ontology.

After invoicing the template 'Schema', the XML Schema's root element is located at the actual position of the process traversing the XML Schema's document tree. At this time, the root element stands for the actual domain element information item, which may include multiple range element information items as content. After the mapping of the actual domain element information item to an equivalent class of the generated ontology, the contained range element information items have to be defined by calling appropriate templates representing the meta-element information items they belong to. As the element information item 'schema' may include complex type definitions, the template 'ComplexType' is invoiced for each top-level complex type definition. Then, each active complex type definition stands for the current domain element information item which is situated at the actual position of the traversing process. When a range element information item should be defined, the template named according to the appropriate meta-element information item is called with the two parameters containing the name of the range meta-element information item (e.g. 'ComplexType') and the name of the range element information item. After the template is invoiced, the range meta-element information item is the current domain meta-element information item and the range element information item is the actual domain element information item, since the domain element information item is now located at the current position of the XML Schema's document tree traversing process. If the root element of the input XML Schema includes the complex type definition 'ComplexType1', for instance, this complex type definition is defined by invoking the template 'ComplexType' with 'ComplexType' as the new domain meta-element information item parameter and 'ComplexType1-Type_<local ontology IRI>-Schema' as the new domain element information item parameter.

The identifier of the current domain element information item is built hierarchically containing all the ancestor element information items' names and the name of the actual domain element information item. The ancestor element information items include the actual domain element information item either directly or indirectly. The XML Schema's root element's identifier is the rightmost part of the actual domain element information item's name. As a consequence, the names of the active domain element information items are built recursively. The input XML Schema's identifier is set to <local ontology IRI>-Schema (e.g. inputXMLSchema-Schema for the input XML Schema with the target namespace 'inputXMLSchema'). The containing ancestor element information items, contributing to the overall identifier of the current domain element information item, are separated by the underscore character. Each element information item's associated meta-element information item is also part of the entire identifier, separated from the individual element information item by the negative sign. If the complex type definition called 'ComplexType1', for example, is contained in the input XML Schema's root element with the target namespace 'inputXMLSchema', the class with the identifier 'ComplexType1-Type_inputXMLSchema-Schema' is added to the generated ontology.

C. Definition of domain element information items as sub-classes of XML Schema Metamodel Ontology's super-classes

Classes, standing for domain element information items located at the current position in the process traversing the XML Schema's document tree, are specified by invoking the named template 'classDefinition' with the local ontology IRI and the current hierarchically built domain element information item's name as formal parameters. Fully qualified classes' identifiers are determined according to the pattern ‚<base ontology IRI>/<local ontology IRI>.owl#<local class identifier>'. The local class identifier always refers to the name of the current domain element information item. In this paper, the abbreviated term 'class identifier' is used, which is equivalent to the term 'local class identifier'. To resolve fully qualified class identifiers, ‚<base ontology IRI>/<local ontology IRI>.owl#' has to be added in front of the local class identifiers.

XML Schemas' element information items are mapped to sub-classes of the XML Schema Metamodel Ontology's super-classes (<element information item> $\sqsubseteq$ <meta-element information item>, e.g. Element1-Element_<local ontology IRI>-Schema $\sqsubseteq$ Element) by invoking the template 'superClassDefinition' with the XML Schema Metamodel Ontology's super-class representing the meta-element information item as parameter.

D. *Definition of hasValue restrictions on XML Schema Metamodel Ontology' datatype properties*

Values of element information items' attributes and any well-formed XML content included in the element information items 'Appinfo' and 'Documentation' are transformed into XML Schema Metamodel Ontology's datatype properties' hasValue restrictions <domain element information item> ⊑ ∃ <attribute>|any_<domain meta-element information item>_String.{<String>}, as the domain 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>|any_<domain meta-element information item>_String' to the specified individual of the primitive datatype 'string'.

To define datatype property hasValue restrictions for the actual domain element information item, the named template 'hasValueRestrictionOnDatatypeProperty' is invoked with the name of the datatype property (e.g. 'name_ComplexType_ String' or 'any_Documentation_String') and the hasValue restriction, which is either the attribute value of the actual domain element information item (e.g. './@name') or the current node ('.'), as parameters. The presence of the optional attributes and element information items' XML contents is checked before the datatype property hasValue restriction can be defined.

If hasValue restrictions on the datatype properties 'any_<Appinfo|Documentation>_String' have to be defined, the default template for element nodes with the template mode 'any' is invoked. The sequence, transmitted to this template, encompasses the current node representing the actual domain element information item 'appinfo' or 'documentation'. The element nodes '<XML Schema namespace prefix>: <appinfo|documentation>' can include any well-formed XML content (i.e. text nodes, element nodes, and element nodes' attributes). The XML content is added to the result tree recursively by calling the element nodes' default template with the template mode 'any' for all the child and descendent element nodes of the element information items 'appinfo' and 'documentation'. The element nodes 'appinfo' and 'documentation' themselves are not part of the output tree. In hasValue restrictions on datatype properties not allowed characters (<, >, ") are escaped and the attribute and text nodes for each actual child or descendent element node are also appended to the output tree.

E. *Definition of universal restrictions on XML Schema Metamodel Ontology' object properties*

1) Subsequent definition of range element information items is not necessary

a) Transformation of values of element information item's attributes referring to other element information items or to type definitions

Values of element information items' attributes 'ref', 'substitutionGroup', and 'refer', referring to other element information items, are converted to the XML Schema Metamodel Ontology's object properties' universal restrictions <domain element information item> ⊑ ∀ <ref|substitutionGroup|refer>_<domain meta-element information item>_<range meta-element information item>.<range element information item>. The reference to the element information item 'attribute' called 'a1' (<xs:attribute ref="a1"/>), for example, is transformed into the object property universal restriction a1-Attribute-Reference <position>_<domain element information item> ⊑ ∀ ref_Attribute_Attribute.a1-Attribute_<local ontology IRI>-Schema.

Values of element information items' attributes 'type' and 'base', referring to type definitions, are transferred to XML Schema Metamodel Ontology's object properties' universal restrictions <domain element information item> ⊑ ∀ type|base_<domain meta-element information item>_ Type.<range element information item>. The value of the attribute 'type' of the element information item 'element' named 'element1' (<xs:element name="element1" type="ComplexType1"/>), for instance, is converted to the object property's universal restriction element1-Element_<local ontology IRI>-Schema ⊑ ∀ type_Element_ Type.ComplexType1-Type_<local ontology IRI>-Schema.

b) Definition of universal restrictions on object properties

Universal restrictions on these two object properties are specified by invoking the template 'universalRestrictionOn ObjectPropertyNotContainedRangeElementInformationItems' with the name of the object property (e.g. 'ref_Attribute_Attribute' or 'type_Element_Type'), the domain element information item's identifier, and either the name of the range element information item (e.g. './@substitutionGroup') or of the type definition (e.g. './@base'), which represent the range element information item part of the universal restriction, as parameters. As the attributes, including the range element information items of the object properties' universal restrictions, are not mandatory, the object properties' universal restrictions can only be defined if the attributes are present.

In order to specify the object properties' universal restrictions, the general template 'universalRestrictionOn ObjectProperty' is called with the name of the object property, the range element information item's ontology IRI, and the name of the range element information item as parameters.

The range element information items do not have to be defined after the specification of the object properties' universal restrictions, since classes, representing other element information items or type definitions, are already defined or will still be defined in the process traversing the XML Schema's document tree.

c) Definition of range element information items' class identifiers and ontology IRIs

As the range element information item can be defined in an external XML Schema, the ontology IRI of the range element information item has to be determined first by invoicing the named template 'getNotContainedRange ElementInformationItemOntologyIRI' with the range element information item as formal parameter. If the attribute includes a namespace prefix, the ontology IRI corresponds to '<base

ontology IRI>/<external local ontology IRI>.owl', since the range element information item is defined in an external XML Schema. And if the attribute does not contain a namespace prefix, the range element information item is specified in the input XML Schema and therefore the ontology IRI is set to '<base ontology IRI>/<local ontology IRI>.owl'.

The class identifier of the range element information item is determined by calling the named template 'getNotContained RangeElementInformationItemIdentifier' with the names of the range element information item and the corresponding meta-element information item as parameters. If an attribute includes a namespace prefix, the referenced range element information item is defined in an external XML Schema. In this case, the range element information item's identifier is set to '<range element information item [without namespace prefix]>-<Range meta-element information item>_<external local ontology IRI>-Schema', since it is always referenced to top-level element information items situated in external XML Schemas. If, however, attributes do not contain namespace prefixes, the global range element information items are specified in the input XML Schemas and therefore the identifiers of the range element information items are set to '<range element information item>-<Range meta-element information item>_<local ontology IRI>-Schema'.

It is assumed that references point to top-level element information items. 'key' (referenced by the attribute 'refer' of the element information item 'keyref') is the only referenced element information item which is not located at the global position in an XML Schema. But as element information items 'key' have to be unique in an XML Schema, the identifiers can also be set as if they would be top-level element information items: <range element information item>-Key_<local ontology IRI>-Schema. And because of this, it can be referenced to XML Schema unique element information items 'key' as if they would be top-level element information items. If referenced keys are defined in external XML Schemas, the target namespaces of the input XML Schema and of the external XML Schema have to be identical. Thus, the local ontology IRI can be used to identify the external XML Schema in which the key is defined. As element information items 'key' are named like top-level element information items, external XML Schemas do not have to be traversed to locate the element information items containing the keys.

If type definitions, specified in external XML Schemas, are referenced, the type definitions' class identifiers have to be determined. The meta-element information item's name is one part of the type definitions' class identifiers. If specific meta-element information items like 'SimpleType' or 'ComplexType' serve as the meta-element information item parts of the type definitions' class identifiers, the corresponding external XML Schemas, in which the type definitions are specified, have to be traversed. And if, in contrast, the general meta-element information item 'Type' serves as the meta-element information item part of the type definitions' class identifiers, the corresponding external XML Schemas do not have to be traversed. An obligatory traversing of external XML Schemas' XML document trees would be critical, since in many cases, external XML Schemas cannot be available physically and namespaces can be imported using arbitrary values of 'schemaLocation' attributes. Because of these reasons, the type definitions' class identifiers' meta-element information item parts 'AnySimpleType', 'SimpleType', and 'ComplexType' are set to 'Type'. 'Type' is the super-class representing the general meta-element information item of the sub-classes standing for the more specific meta-element information items of type definitions (i.e. simple ur-type, simple type and complex type definitions).

2) Subsequent definition of range element information items is necessary

a) Transformation of XML Schemas' element information items' part-of relationships

Element information items' part-of relationships are realized by XML Schema Metamodel Ontology's object properties' universal restrictions <domain element information item> $\sqsubseteq$ $\forall$ contains_<domain meta-element information item>_<range meta-element information item>.<union of range element information items>. If the input XML Schema's root element includes only one 'element' element information item (<xs:schema … ><xs:element name="element1"/> </xs:schema>), the range of the object property can only consist of individuals of one class (<local ontology IRI>-Schema $\sqsubseteq$ $\forall$ contains_ Schema_Element.element1-Element_<local ontology IRI>-Schema). If element information items like 'schema' have more than one element information item as content (<xs:schema … ><xs:element name="element1"/> <xs:element name="element2"/> </xs:schema>), 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 (<local ontology IRI>-Schema $\sqsubseteq$ $\forall$ contains_Schema_Element.(element1-Element_ <local ontology IRI>-Schema $\sqcup$ element2-Element_<local ontology IRI>-Schema)).

For each XML Schema Metamodel Ontology's object property 'contains_<domain meta-element information item>_<range meta-element information item>' defined for the actual domain element information item's associated meta-element information item, the template 'universalRestriction OnObjectPropertyContainedRangeElementInformationItems' is called by passing the names of the object property and of the current domain element information item as parameters. As the template is called for each object property 'contains_<domain meta-element information item>_<range meta-element information item>' of the actual domain element information item, it is tested, if the current domain element information item really includes the range element information items of the given range meta-element information item. Only in this case, the universal restriction on the object property is defined.

b) Definition of range element information items' class identifiers and ontology IRIs

In order to store a string sequence of all the range element information items' names of the given range meta-element information item contained in the actual domain element information item in a variable, it is iterated over these range element information items. The template 'getContained RangeElementInformationItemIdentifier' is invoked for each contained range element information item with the names of the range meta-element information item and the current domain element information item as formal parameters.

Range element information items may have an associated name, may be references to top-level, global element information items, or may be contained without an identifier. Range element information items, having the attribute ‚name', are named according to the pattern ‚<range element information item>-<Range meta-element information item>_<domain element information item>' (e.g. <xs:simpleType name="st3"> is converted to st3-Type_<domain element information item>). As element information items 'key' have to be unique in an XML Schema, their identifiers are determined as if they would be top-level element information items: <range element information item>-Key_<local ontology IRI>-Schema.

Range element information items, which do not have an associated name, may have an attribute 'ref'. If the attribute 'ref' contains a namespace prefix, the referenced element information item is defined in an external XML Schema and the input XML Schema's range element information item's identifier is set to '<range element information item [without namespace prefix]>-<Range meta-element information item>-Reference<position()>-<namespace URI>_<domain element information item> (e.g. <xs:attribute ref="xml:lang"/> is transformed to lang-Attribute-Reference<position>-http://www.w3.org/XML/1998/namespace_<domain element information item>). The namespace URI is part of the identifier, as references to top-level element information items defined in different namespaces are possible (e.g. <xs:attribute ref="lang"/> and <xs:attribute ref="xml:lang"/>). Identifiers without namespace URI statements would not be unique. Domain element information items may include multiple range element information items of the same meta-element information item with identical 'ref' attribute values (e.g. <xs:extension…><xs:attributeGroup ref="ag1"/><xs:attribute Group ref="ag1"/></xs:extension>). To ensure the uniqueness of range element information items' identifiers, their positions within the domain element information item have to be part of their identifiers. If the attribute 'ref' does not include a namespace prefix, the referenced element information item is specified in the input XML Schema and the name of the input XML Schema's referencing element information item is determined as '<range element information item>-<Range meta-element information item>-Reference<position()>_<domain element information item> (e.g. <xs:attribute ref="a3"/> is translated into a3-Attribute-Reference<position>_<domain element information item>).

As domain element information items may include multiple range element information items, which have no attributes 'name' or 'ref', of the same meta-element information item, these range element information items will be identified using sequential numbers: <Range meta-element information item><position()>-<Range meta-element information item>_ <domain element information item> (e.g. <xs:schema><xs:annotation/><xs:annotation/></xs:schema> is transformed into Annotation1-Annotation_<local ontology IRI>-Schema and Annotation2-Annotation_<local ontology IRI>-Schema). If simple or complex type definitions are included, the first range meta-element information item part of the identifier is set to the more specific type definition 'SimpleType' or 'ComplexType' and the second range meta-element information item part of the name is set to the super-class representing the more general type definition 'Type' (e.g. SimpleType1-Type_<domain element information item>), in order to distinguish simple and complex type definitions.

A variable stores the string sequence of IRIs of the ontologies in which the individual range element information items of the given range meta-element information item are defined. To realize this, the template 'getContained RangeElementInformationItemOntologyIRI' is called for each range element information item. As contained rage element information items are always specified in the input XML Schema, the ontology IRI is determined as '<base ontology IRI>/<local ontology IRI>.owl'.

c) Definition of universal restrictions on object properties

To specify the universal restriction on the given object property, the XSLT processor invokes the general template ‚universalRestrictionOnObjectProperty' with the name of the object property and the two string sequences consisting of the range element information items' identifiers and the associated ontology IRIs as parameters. As the active domain element information item may contain one or multiple range element information items of the same meta-element information item, the class corresponding to the actual domain element information item can only have ‚contains_<domain meta-element information item>_<range meta-element information item>' relationships with one specific class representing a range element information item or with a union of specific classes standing for contained element information items.

d) Definition of range element information items

As contained element information items are referenced in definitions of universal restrictions on the object properties 'contains_<domain meta-element information item><range meta-element information item>', the range element information items have to be defined as well by invoking the template 'rangeElementInformationItemsDefinition' with the object property's name and the domain element information item's identifier as parameters. For each range element information item of the given range meta-element information item (e.g. for each global complex type definition contained in the input XML Schema's root element 'schema'), the range element information item's identifier is determined as shown before and the template named after the range meta-element information item is invoked recursively in order to define each range element information item of the given meta-element information item. The range meta-element information item and the range element information item serve as parameters. After the template is invoiced, the passed range element information item is then the current domain element information item, which is now at the actual position in the process traversing the XML Schema's document tree.

V. Related Work

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 on the instance, the conceptual, or both the instance and the conceptual level.

On 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 [7]. As extension to this approach, Battle has introduced a bidirectional mapping of XML components to RDF [8]. 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 [9]. O'Connor and Das developed an approach transforming XML documents to individuals of an OWL ontology describing the serialization of the XML document. SWRL [10] is used to map these instances to individuals of a domain ontology [11].

On the conceptual level you can distinguish between approaches converting XML schema languages to RDFS or OWL. Several languages for writing schemas like DTD [1], XML Schema [6], DSD [12] and Relax NG [13] exist. The prototype OntoLiFT [14] 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 [15]. 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 the specific one XML Schema to ontologies. Anicic et al. evolved an approach based on meta-models transforming between the different models of XML Schema and OWL [16].

On 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 [17]. Using the approach of O'Connor and Das [18], 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 [19] and XPath [20] 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 [21]. 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 [22]. Besides that, the approach of Tous et al. is very similar to this method [23]. The authors of [24] 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 representing database columns, are transformed into datatype properties with the surrounding element as domain. Besides, XML cardinality constraints are transformed into equivalent OWL cardinality restrictions.

Compared with former general-purpose tools for translating XML Schemas into OWL ontologies, the approach presented in this paper converts XML Schemas into OWL ontologies on the basis of the XML Schema meta-model [6]. As this approach regards each meta-element information item of the XML Schema for XML Schemas, any XML Schemas can be transformed into OWL ontologies completely automatically. Many approaches try extracting 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 on the conceptual level or XML to RDF on 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 attempts 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, divers existent methods generate RDFS ontologies and not the more expressive OWL ontologies.

VI. Conclusion

The ontology design process is sped up significantly when XML Schemas are transformed automatically into generated ontologies. The authors introduced the implementation of the developed generic multilevel approach designing domain ontologies based on XML Schemas using XSLT transformations. Both the terminology and the syntactic structure of domain data models are mapped to generated ontologies. The generated ontologies are linked to domain ontologies in order to supplement the information about terms and document structures located in the XML Schemas with domain specific semantic information.

VII. Future Work

The authors will develop an XSLT framework to implement a complete approach generating OWL ontologies stylesheet-driven. So far, XSLT transformations build generated ontologies automatically based on arbitrary XML Schemas. Moreover, the authors will write XSLT transformations transforming XML documents without corresponding XML Schemas, determining their syntactic structure, into 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] Extensible Markup Language (XML) 1.0 (fifth edition) - W3C recommendation 26 November 2008, http://www.w3.org/TR/2008/REC-xml-20081126/.

[2] XML Schema part 0: primer second edition - W3C recommendation 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/.

[3] Linked Data, http://linkeddata.org.

[4] The XML data model, http://www.w3.org/XML/Datamodel.html.

[5] Resource Description Framework (RDF): concepts and abstract syntax, http://www.w3.org/TR/2002/WD-rdf-concepts-20021108/.

[6] XML Schema part 1: structures second edition – W3C recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-1/.

[7] M.C.A. Klein, "Interpreting XML documents via an RDF Schema ontology," in *13th International Workshop Database and Expert Systems Applications*, Aix-en-Provence, 2002.

[8] S. Battle, "Gloze: XML to RDF and back again," in *1st Jena User Conf.*, Bristol, 2006.

[9] G. Reif, H. Gall, and M. Jazayeri, "WEESA - web engineering for Semantic Web applications," in *14th World Wide Web Conf.*, Chiba, 2005.

[10] SWRL: a Semantic Web Rule Language combining OWL and RuleML, http://www.w3.org/Submission/SWRL/.

[11] M.J. O'Connor and A.K. Das, "Semantic reasoning with XML-based biomedical information models," in *13th World Congr. Medical Informatics*, Cape Town, 2010.

[12] N. Karlund, A. Moller, and M.I. Schwartzbach, "DSD: a schema language for XML," in *ACM SIGSOFT Workshop Formal Methods in Software Practice*, 2000.

[13] J. Clark, J. Cowan, M. Fitzgerald, J. Kawaguchi, J. Lubell, M. Murata, N. Walsh, and D. Webber, "Information technology – document schema definition language (DSDL) – part 2: regular-grammar-based validation," – RELAX NG. ISO/IEC 19757-2:2003(E), 2003.

[14] R. Volz, D. Oberle, S. Staab, and R. Studer, "OntoLiFT Prototype – WonderWeb: ontology infrastructure for the Semantic Web," Karlsruhe, 2003.

[15] M. Murata, D. Lee, M. Mani, and K. Kawaguchi, "Taxonomy of XML schema languages using formal language theory," in *ACM Transactions Internet Technology*, vol. 5, New York, 2005.

[16] N. Anicic, N. Ivezic, and Z. Marjanovic, "Mapping XML Schema to OWL," in *Enterprise Interoperability*, Part V, Springer, Berlin, 2007, pp. 243-252.

[17] M. Dell'Erba, O. Fodor, F. Ricci, and H. Werthner, "Harmonise: a solution for data interoperability," in *Proc. 2nd IFIP Conf. E-Commerce, E-Business, E-Government I3E*, 2002.

[18] M.J. O'Connor, A.K. Das, "Acquiring OWL ontologies from XML documents," in *Proc. 6th Int. Conf. Knowledge Capture*, New York 2011.

[19] OWL 2 Web Ontology Language Manchester Syntax, http://www.w3.org/TR/owl2-manchester-syntax/.

[20] XML Path Language (XPath) 2.0 (second edition), http://www.w3.org/TR/xpath20/.

[21] M. Ferdinand, C. Zirpins, and D. Trastour, "Lifting XML Schema to OWL," in *Web Engineering - 4th Int. Conf.*, Munich, 2004.

[22] N. Kobeissy, M.G. Genet, and D. Zeghlache, "Mapping XML to OWL for seamless information retrieval in context-aware environments," in *Int. Conf. Pervasive Services*, Istanbul, 2007.

[23] R. Tous, R. Garcia, E. Rodriguez, and J. Delgado, "Architecture of a semantic XPath processor. Application to digital rights management," in *6th E-Commerce and Web Technologies*, Copenhagen, 2005.

[24] H. Bohring, S. Auer, "Mapping XML to OWL Ontologies," in *Leipziger Informatik Tage*, vol. 72, Leipzig, 2005.