# Lifting XML Schema to OWL

Matthias Ferdinand[1], Christian Zirpins[1], and David Trastour[2]

[1] VSIS Group, University of Hamburg, Germany
{6ferdina,zirpins}@informatik.uni-hamburg.de
[2] Hewlett-Packard Laboratories Bristol, UK
david.trastour@hp.com

**Abstract.** The Semantic Web will allow software agents to understand and reason about data provided by Web applications. Unfortunately, formal ontologies, needed to express data semantics, are often not readily available. However, common data schemas can help to create ontologies. We propose mappings from XML Schema to OWL as well as XML to RDF and show how web engineering can benefit from the gained expressiveness as well as the use of inference services.

## 1 Introduction

The Semantic Web will allow software agents to understand, share and reason about data that is provided by Web application systems. Formal conceptual models, or ontologies, are necessary to express the semantics of the data. Unfortunately, semantic information is not usually available in such a form, but scattered across documentation and various software components. Because developing ontologies from scratch is costly and difficult, one should try to reuse this semantic information as much as possible. Thanks to their formal nature, document schemas like XML Schema provide a good basis for developing or re-engineering ontologies (see e.g. [1,2] for a comparison of ontology languages, web standards and markup languages).

In this work, we focus on extracting semantic information out of document schemas and propose a mechanism to lift XML Schema to the Web Ontology Language (OWL). While, for example, concrete translation procedures from OIL or XOL to XML Schema have been developed by [3,4], we specify and implement a mapping in the reverse direction producing an OWL ontology. In order to apply this semantic meta-information for reasoning on instance data, XML documents have to be mapped to RDF, bridging the gap between those models. General solutions typically require changes to XML or RDF: Melnik [5] developed an RDF interpretation for XML documents. In [6,7] a formal model and an architecture have been developed that allow uniform access to both types of documents. We propose another such mapping that does not require changes to the standards and incorporates XML Schema type information. Subsequently, based on our two mappings, we show that the engineering of XML based web applications can benefit from the high expressive power OWL has to offer and from inference services such as classification or satisfiability checking.

In the following, sect. 2 proposes two mapping concepts from XML to RDF and from XML Schema to OWL that allow lifting data from syntax to representation. Sect. 3 shows how reasoning techniques on such representations can be applied in web engineering. Finally, sect. 4 concludes.

## 2 From Syntax to Representation: Mapping Concepts

In this section, we propose a general binding of XML structured data to Semantic Web languages. The approach is twofold: XML documents are translated into RDF graphs and XML Schemas are lifted to OWL ontologies. This applies to all XML documents that conform to an XML Schema.

The first part of the concept concerns the *XML to RDF mapping.* XML is a language that defines a generic syntax to store and exchange documents by means of a tree-based structure. Although RDF has an XML-based syntax, XML and RDF serve different purposes and have been developed separately within the W3C, which lead to different modelling foundations.

XML is based on a tree model where only nodes are labeled and the outgoing edges are ordered. This model originates from semi-structured data and databases. In contrast to this, RDF is based on a directed graph model where edges have labels but are unordered. It distinguishes between resources (e.g. car) and properties (e.g. car color) while XML does not (e.g. both would be elements). This model originates from knowledge representation languages such as Frames [8] and description logics (DL).

To bridge the gap between both forms of data representation, we developed a procedure that transforms XML documents to RDF data models. In order to keep compatibility with existing documents and applications, this mapping does not require any change on either XML or RDF specifications (however, so-called mixed content models of XML are not fully supported). Structural differences of the data models represent no obstacle, as trees are a specialisation of graphs. We make a distinction between (1) elements that have sub-elements and/or attributes, and (2) attributes and elements that carry only a data type value. These two categories of components correspond respectively to XML Schema declarations associated with a `complexType` or a `simpleType`. The mapping is performed by the following procedure:

Initially, an RDF resource `Document` is created – representing the XML document itself. Then, for each sub-element and attribute of the element that is currently processed (starting with the root element), an RDF property on the RDF resource created before in the previous step is created. If we encounter a data type component (2nd category above), its data value is represented as an RDF literal on the respective property. If we encounter an object component (1st category above), an anonymous RDF resource is created and assigned as the value of the respective property. Then, this component is processed recursively.

As we also want to map XML Schema, it is desirable to transparently incorporate the type information specified in the corresponding schema. To facilitate this, we presume that an XML Schema-aware processor has validated the XML document, which results in type information represented in a Post-Schema Validation Infoset (PSVI). We will see that each XML Schema `complexType` is mapped into an OWL class. Hence each mapped RDF resource is of `rdf:type` the OWL class corresponding to the PSVI retrieved `complexType`.

The second part of the concept concerns the *XML Schema to OWL mapping.* XML Schema and OWL solve different problems: XML Schema provides means to express and constrain the syntax and structure of XML documents. OWL, in contrast, is intended for modelling the semantic relationships of a domain. However, there is an interesting overlap between the two, as both of them have an object-oriented foundation. XML Schema has the notion of class hierarchy and specialisation, and OWL is based on the notion of Frames. Although they accomplish it at two different levels of abstraction, the languages share the goal of defining common vocabularies and structures to support electronic exchange of information. Our mapping approach that complements the one seen before, capitalizes on these similarities. In the following, we give an overview of the fundamental choices. The mapping procedure of a complete XML Schema is composed of the mapping of its different components.

**Main Concepts:** Each XML Schema `complexType` is mapped to an `owl:Class`. Each `element` and `attribute` declaration is mapped to an OWL property. More precisely, elements of `simpleType` and all attributes are mapped to an `owl:Data typeProperty`; elements of `complexType` are mapped to an `owl:ObjectProperty`. Finally, the `schema` root element of a schema is mapped to an OWL `Class` of name 'targetNamespace + `#Schema`'.

**Model Groups:** Model group definitions and attribute group definitions are specialisations of complex types since they only contain element respectively attribute declarations. Hence, they are also mapped to OWL classes.

**Specialisation:** In object-orientation, inheritance mechanisms represent a central modelling tool which is used to express "is-a" relationships between classes. The literature differentiates between various types of inheritance, two of the most important ones are inheritance by restriction and inheritance by extension. XML Schema supports both of these ways by corresponding type derivation constructs and we both map them to `rdfs:subClassOf` in OWL, its only inheritance mechanism. XML Schema offers the `substitutionGroup` construct which specifies that an element can be replaced by a set of other elements in the instance document. Analog to the type derivation mechanisms, this construct can be interpreted as a way to express a specialisation of elements and thus is mapped to an OWL `subPropertyOf`.

**Type and Cardinality:** In XML Schema, "Particles" (resp. "AttributeUses") are used to associate a type and cardinality to a local element (resp. a local attribute).Because these definitions have a local scope, we map them to the intersection of two property restrictions: one restricting the type with `owl:allValuesFrom`, the other restricting the cardinality with either `owl:minCardinality`, `owl:maxCardinality`, or `cardinality`. The two restrictions apply to the same property (i.e. the one corresponding to the element or attribute).

**Compositors:** XML Schema offers three compositors to combine elements, `sequence`, `all` and `choice`. They are mapped to appropriate OWL boolean expressions. The difference between `sequence` and `all` is purely syntactic; semantically they are both conjunctions and are both mapped to an `owl:intersectionOf` constructor. The mapping of the `choice` compositor is more verbose since there is no direct equivalent in OWL to an exclusive-OR. Hence, it needs to be constructed with a boolean expression (with `owl:intersectionOf`, `owl:unionOf` and `owl:complementOf`).

**Global Elements:** Global element and attribute declarations are mapped similarly to local ones. Associated restrictions are added to the `Schema` class.

Identifiers are mapped from XML Schema to URIs by concatenating the `targetNamespace` URI, the `#` character and the component's local name. Problems can occur due to the fact that XML Schema partitions the `targetNamespace` into distinct so-called symbol spaces, one for each kind of definition or declaration. To prevent naming conflicts in OWL, the mapping process applies an appropriate renaming pattern to the affected components.

As a detailed discussion is out of scope here, we can only briefly note that we also found mappings for other language constructs of less common interest. However, because of a limited expressiveness in OWL or because the construct would not be appropriate, we also had to skip some non-essential language components like `abstract`, `final`, `block`, `default`, `form`, wildcards, identity-constraint definitions and `complexType`s derived by restriction from `simpleType`s.

## 3 Reasoning Support for Web Engineering

There are a number of promising applications for the mapping concept in Web engineering. In particular, it allows enhancing traditional XML languages and tools by the capabilities of OWL reasoners. Here, we distinguish support capabilities at design time and runtime of web applications. At *design time*, we see two principal usages for the mapping. On the one hand, ontologies can be extracted out of existing XML Schemas. This skeleton ontology can then be extended using OWL expressions. On the other hand, the mapping can support schema design. Analogous to the use of reasoners to design ontologies [9], they are useful to design XML Schemas. By using `owl:equivalentClass` instead of `rdfs:subClassOf` for the mapping of `complexType`, an OWL reasoner can infer implicit subsumption relationships, thus identifying super-types of some `complexType`s. This fosters reuse and limits the class proliferation when large number of classes are encountered. An OWL reasoner could also help to check the compatibility of two independently developed schemas. DL based reasoners are the most suited for this type of operation as they can do efficient inference on classes. At *runtime*, the XML mapping into RDF can be used to do inference and semantic validation of XML data. Once translated into RDF, the data can be classified with an OWL reasoner. The classification could lead to discover implicit class membership or implicit relationships between objects. Finally, semantic validation can be performed by looking for unsatisfiable concepts.

## 4 Conclusion

We have proposed a general solution for automated binding of XML structured data to Semantic Web languages. General procedures have been shown to map XML documents to RDF graphs and XML Schemas to OWL ontologies. Subsequently, supporting techniques for the engineering of web applications have been presented that get possible by integrating mapping results with OWL reasoners.

To underpin the concepts, we offer a Java software toolkit that implements the mapping process [1]. In terms of engineering concepts, we note that we incorporated the RACER DL reasoner [10] and used its inference services to realise a real-world e-business Web application in the context of RosettaNet [11].

By automatically generating formal conceptual models from semi-structured data, our approach supports the automated bootstrapping of ontology development from existing XML Schemas, speeding up the adoption of Semantic Web technologies. It opens up to a wide range of XML based web applications the expressive power of OWL as well as the potentials of inferencing services. Unlike most traditional techniques (e.g. hard coded validation), semantic constraints can be written in a formal, well-documented and reusable fashion that can be applied to various tasks such as semantic validation of XML instances.

## References

1. Fensel, D.: Relating Ontology Languages and Web Standards. In: Modelle und Modellierungssprachen in Inf. und WiInf., St. Goar, Fölbach-Verlag (2000)
2. Gil, Y., Ratnakar, V.: A comparison of (semantic) markup languages. In: Proc. 15th Intl. Florida Artificial Intelligence Research Society Conf., May 14-16, 2002, Pensacola Beach, AAAI Press (2002) 413–418
3. Klein, M., Fensel, D., van Harmelen, F., Horrocks, I.: The Relation between Ontologies and XML Schemas. Linköping Electr. Art. in Comp. and Inf. Sci. **6** (2001)
4. Rami, R., Nabila, B.: Translation Procedure to Clarify the Relationship Between Ontology and XML Schema. In: Proc. Intl. Conf. on Internet Computing (IC'2001), Las Vegas, CSREA Press (2001) 164–170
5. Melnik, S.: Bridging the Gap between RDF and XML (Accessed 1 Feb 2004) `http://www-db.stanford.edu/~melnik/rdf/fusion.html`.
6. Patel-Schneider, P., Sim on, J.: The Yin/Yang Web: XML Syntax and RDF Semantics. In: Proc. 11th Intl. WWW Conf. (WWW11), ACM (2002)
7. Patel-Schneider, P.F., Sim on, J.: Building the Semantic Web on XML. In: Proc. 1st Intl. Semantic Web Conf. 2002 (ISWC'02). (2002)
8. Minsky, M.: A Framework for Representing Knowledge. Technical report, Massachusetts Institute of Technology (1974) MIT-AI Laboratory Memo 306.
9. Bechhofer, S., Horrocks, I., Goble, C., Stevens, R.: OilEd: a reason-able ontology editor for the semantic web. In: Proc. DL-2001, CEUR Elct. Proc. vol. 49 (2001)
10. Haarslev, V., Möller, R.: Description of the RACER system and its applications. In: Proc. DL-2001, CEUR Elct. Proc. vol. 49 (2001)
11. Trastour, D., Preist, C., Coleman, D.: Using Semantic Web Technology to Enhance Current Business-to-Business Integration Approaches. In: Proc. EDOC 2003, IEEE (2003) 222–231

[1] http://www.servicecompostion.org/owlmap.php