OCML ontologies to XML schema lowering

Conference or Workshop Item

OCML Ontologies to XML Schema Lowering ⋆,⋆⋆

Vlad Tanasescu[1,2], John Domingue[2], Liliana Cabral[2]

[1] Swiss Federal Institute of Technology (EPFL), 1015 Lausanne, Switzerland
vlad.tanasescu@epfl.ch
[2] Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK
{v.tanasescu, j.b.domingue, l.s.cabral}@open.ac.uk

Abstract. Ontologies are explicit specifications of a conceptualization intended for logical processing. They are used to express meaning and to apply it on otherwise less structured data. Nevertheless other models of organization and structure of knowledge, like database models and XML document definition standards, are widely used and valuable. In order to extend ontology usage to multiple layers of an application, or to integrate ontologies with pre-existing software, the use of these various means to structure data could be necessary. We are therfore using metadata in order to adapt an ontology to these use cases and provide the example of automatically generating user interfaces for goal descriptions in IRS-III [1] by extracting an XML schema from an ontology.

1 Introduction

The semantic web vision foresees the advent of automatic machine operable services through the use of knowledge based technologies. An interesting class of standardized services are *web services*, which are widely accepted if not yet massively used by the industry. Current web service implementations are, however, relatively inflexible, and not powerful enough to support automatic discovery, mediation and composition. Therefore ongoing research is investigating how semantic web technology can alleviate this. The IRS-III (Internet Reasoning Service) framework [1] supports the creation of semantic web services according to the WSMO ontology [2], and extends it. Users of IRS-III directly invoke web services via *goals* i.e. IRS-III supports *capability-driven* service execution. A goal is described as a class in OCML (Operational Conceptual Modelling Language) [3] and related to one or more web services through mediators, which provide

⋆ This research was partially supported by the Advanced Knowledge Technologies (AKT) project. AKT is an Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University.

⋆⋆ This work was also partially supported by the DIP (Data, Information and Process Integration with Semantic Web Services) project. DIP (FP6 - 507483) is an Integrated Project funded under the European Unions IST programme.

the required flexibility by allowing to specify mapping mechanisms between goal input roles, output roles, and matching web services.

Ontologies, like the ones used in a goal description to define a capability, are intended for logical inference, and are often loosely related to applications based on them; therefore one has to provide some *glue* to allow diverse aspects of an application to conform to an ontology; the problem of data consistency, of user input for example, usually occurs and is hard to tackle in a generic way.

Our motivating example is to provide a consistent interface to IRS-III goals. The input to these goals, i.e. the data one has to provide in order to expect achievement of the goal, is defined in an OCML ontology, and uses data types provided by it. For example an *exchange rate* goal will have three input roles: *has-source-currency*, *has-target-currency* and *has-amount*, which represents the amount which has to be converted from source to target currency.

The difficulty is that OCML types are used to describe the roles, i.e. *pound* or *euro* are instances of a *currency* class. The enumeration of available currencies is therefore well defined in the ontology and this constraint has to be reflected in the actual web or Java goal access interfaces, task which is actually achieved by the developer, in a disconnected and hence highly error-prone way.

2 Translating OCML ontologies to XML schema

The relation between ontologies and XML schemas (XSD [5]) has already been described (for example in [4]): ontology languages are a means to specify domain theories while XML schemas provide integrity constraints for information sources, but both provide vocabularies and structure for describing information sources that are intended for exchange. However ontology modelling languages like OCML do not offer rich collections of built-in data types for two main reasons:

1. Providing clear semantics and reasoning support for a large collection of complex data types is difficult.
2. The precise representation of a data type is often superfluous in a knowledge modeling context, i.e. a *date* may be an important aspect of a domain but various representations of it are not.

The XML schema type system has been inspired by language independent data types as well as actual query and object-oriented programming idioms like SQL or JAVA [5]. OCML is a frame-based, i.e. object-centered, knowledge representation system, which also provides a relational view. These systems therefore present similarities that we can exploit and a quite intuitive mapping appears between the two (see table 1).

Unfortunately the XSD type system does not support multiple inheritance, and even simple derived types can be only created *either* by *restriction* or by *extension* but not both at the same time. However this kind of modelling is mostly useful for knowledge representation and have usually little role to play when interacting with other software architectures. Moreover we are only interested

Table 1. OCML to XSD data types

OCML	XSD
class	complexType
slot	element
string, float, ...	simpleType

by the *constraints* allowed by an XSD definition, not by the reuse of the derived XSD types since we are able to generate them on the fly. Therefore we chose to treat these cases by defining a new type for every OCML class, inherited or not, containing all the slots (local and inherited) as elements. Also there is actually no planned support in our approach for classes defined in OCML *intensionally* by logical expressions.

Still the result of this simple mapping is not very helpful from an applicative point of view since we do not have any validation constraints yet, i.e. we do not know, because it is not specified in the ontology, that for our purposes the slot/element called *has-amount* must be smaller than a given value (otherwise the program may crash, or the database will not be able to store the value, etc).

3 Constraint metadata applied to ontologies

Constraint information could simply be added to the ontology, however the applicative environment (Java program, web interface, database) may change while the ontology is already shared in multiple contexts, therefore it is more appropriate to only *attach* this information to the ontology, without any modification, hereby allowing for multiple *views* or *aspects* of it depending of the applicative context. Without doing so constraints on a goal could be superfluous or even conflicting depending on the context, e.g. the maximum value of a number could be greater for a Java application than for a Web interface, while entirely meaningless for the knowledge domain.

During the translation process we therefore use a metadata system to store the constraints we chose to add to the ontology XSD representation for our specific context dependent purpose. Our metadata system is composed of:

1. A *naming convention* mimicking the *has-a* tree-like structure of the ontology and allowing to request information about a particular element of it (class, slot or relation). For example:

   ```
   '(transform classes EXCHANGE-RATE-GOAL HAS-AMOUNT)
   ```

2. A *repository* which can be anything from a DBMS to simple association lists, able to store the information regarding a node according to the naming structure, for example the required length of a string, as well as other relevant information like OCML basic types mappings to XSD:

```
'((transform ((classes ((EXCHANGE-RATE-GOAL
                          ((HAS-AMOUNT
                            ((maxInclusive ((1000000)))))))))
              (types ((float (("xs:decimal")))))))
```

3. An *access interface* to the metadata repository that has to provide necessary access functions as well as commodity ones required by the generation program.

By using the metadata convenience (through the repository called *mddb*) we can now easily map an OCML ontology to a constrained XSD schema:

Algorithm *OCML2XSD*(*ontology*, *mddb*)
1. **for** $c \in classes(ontology)$
2. **do** Create element *complexType* with attribute $name = name(c)$
3. Create *sequence* element
4. **for** $s \in slots(c)$
5. **if** s is a type referenced as *basic* in *mddb*
6. **then** create element *simpleType*
7. add *base* attribute with value specified in *mddb*
8. **if** s has attached constraints in *mddb*
9. **then** add the constraints as *restriction* elements
10. **else** create element with *type* attribute set to $class(s)$

It is straightforward to restrict this algorithm to the slots representing input roles of an IRS-III goal. Then, by applying an XSL transformation to the generated schema, we obtain a web interface with embedded validation.

4 Conclusion

We demonstrated the use of metadata applied to ontologies in order to adapt them to practical application contexts, and used this technique to generate XML schema from an OCML goal description in order to obtain a consistent validating web interface. Metadata can be used to produce many applicative *views* of an ontology, like SQL Data Definition Language statements for persisting ontology instances, or JavaBeans template code to ease ontology driven enterprise application building. In further work we aim to generalise this process by providing a language for building generative mappings.

References

1. J. Domingue, L. Cabral, F. Hakimpour, D. Sell and E. Motta (2004). IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services. (WIW 2004)
2. Web Service Modeling Ontology Standard, http://www.wsmo.org/2004/d2/
3. E. Motta, An Overview of the OCML Modelling Language (KEML '98)
4. M. Klein, D. Fensel, F. van Harmelen, and I. Horrocks, 'The Relation between Ontologies and XML Schemas' (2001)
5. XML Schema - W3C Recommendations 28 October 2004, http://www.w3.org/TR/xmlschema-0/ and http://www.w3.org/TR/xmlschema-2/