Approaches to Implementing Tailored Metaschemas in XML

Martin Bernauer, Gerti Kappel, Gerhard Kramler

Business Informatics Group, Vienna University of Technology, Austria
{lastname}@big.tuwien.ac.at

Abstract. The unique selling point of XML as standard representation of content is its ease of use thus facilitating interoperability between various partners and platforms. To overcome XML Schema's weakness concerning semantic expressiveness, tailored XML schema languages would be favorable to represent domain specific knowledge. The contribution of this paper is to identify various approaches to implementing tailored XML schema languages with XML Schema.

1 Introduction

Tailored schema languages define domain concepts thus semantics once and for all across schemas. In relational databases for example, the schema language defines concepts such as tables and foreign keys, constituting modelling primitives for database schemas. Analogously tailored schema languages exist for data warehousing and others.

As the usage of XML increases, the need for tailored XML schema languages[1], which go beyond the semantic expressiveness of XML Schema, arises. This goes in parallel with the emerging practice to define an XML syntax both for schemas and instances (e.g., as RDF does).

Using XML for schemas and instances instead of using other data formats is beneficial with respect to *interoperability*, *openness*, and *integration*. This means that schemas and instances described with XML syntax are accessible under various platforms and environments, they can be extended by employing XML namespaces, and they can be integrated with other XML standards such as XLink, XSLT, and RDF.

The contribution of this paper is to identify approaches to implementing tailored metaschemas with XML Schema. In particular, four approaches with distinct characteristics are presented in the following section.

To talk about the various levels of tailoring, we adopt a four layer metadata architecture as for example proposed by OMG's Meta Object Facility [5]. The layers comprise the instance layer (short "M0") for data, the schema layer ("M1") for metadata describing data, the metaschema layer ("M2") describing

[1] For the purpose of readability, we use the term metaschema instead of schema language in the rest of the paper. If we talk about a schema expressed in XML Schema (the schema language proposed by the W3C), we concisely call it XML schema.

metadata, and the meta-metaschema layer ("M3") describing meta-metadata. Between elements of the various layers instance-of relationships exist in the sense that an element of M*i* is an instance of an element of M*(i+1)* with the exception that depending on the underlying conceptual model an element of M3 can be seen as an instance of itself.

2 Approaches

The Proprietary Schema approach expresses schemas directly in terms of a tailored metaschema, constituting the "most natural" approach with respect to schema design. As shown in Figure 1, schema s (an XML document) is created by instantiating tailored metaschema m (e.g., an XML schema).

Schemas usually describe both intensional and extensional aspects. For intensional aspects, i.e., aspects that apply for all instances but have no corresponding materialization at the instance level, it is not necessary to consider an XML serialization. On the contrary, for extensional aspects it is necessary to define a so-called *instance transformation function* τ_0*: I→XML*, which transforms a proprietary instance into an XML document. It is important to note that τ_0 is defined independent of a particular schema, i.e., at M2. Therefore it can be reused across applications.

For example, RDF employs the Proprietary Schema approach. The standard defines a serialization of RDF instances as XML by an EBNF grammar, which can be seen as a declarative specification of τ_0. In addition, the RDF Schema (RDFS) standard defines a tailored metaschema for RDF.

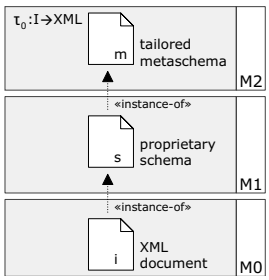

Fig. 1. Proprietary Schema Approach

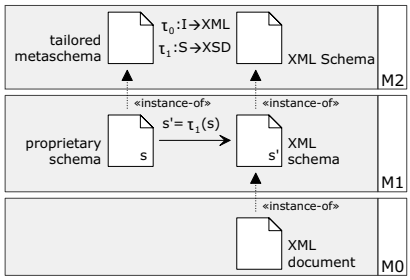

Fig. 2. Side by Side Approach

The Side by Side approach is similar to the Proprietary Schema approach in that it uses an explicit tailored metaschema to define schemas. However, an XML schema is provided in addition to the proprietary schema. The XML schema does

not replace the original, proprietary one, but stands side by side to it. Likewise, instance transformation function τ_0 is still used to serialize instances as XML.

The transformation of a proprietary schema s into an XML schema s' is defined at M2 by *schema transformation function* τ_1*: S→XSD* and applied at M1 as depicted in Figure 2. Because only extensional aspects of the proprietary schema are transformed to an XML schema, τ_1 is *partial*.

For example, [4] employs the Side by Side approach. It describes τ_1 textually, which transforms OIL ontologies to XML schemas. As an example independent of XML, [6] extensively employs the Side by Side approach, yielding an abstract algebra for model mapping [1].

The Framework approach uses only an XML schema that expresses all circumstances formerly modelled by the proprietary schema. Thereby it eliminates the need for proprietary schema s, transformation τ_1, and synchronization of s with s'. Since intensional aspects are orthogonal to XML Schema, they can be expressed easily using XML Schema extension mechanisms (annotations and foreign attributes). Expressing extensional aspects is more complicated as they must be expressed solely with concepts provided by XML Schema.

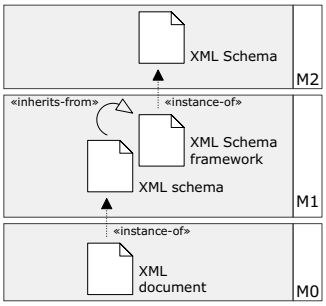

Fig. 3. Framework Approach

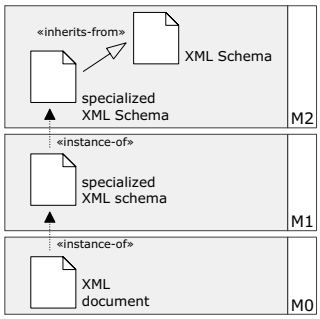

Fig. 4. Specialized XML Schema Approach

The framework concept [2] succors to define extensional aspects. A framework usually provides a base schema common to all applications, along with conventions for adapting and using it. XML Schema provides concepts that can be employed in framework design, such as abstract types or type derivation. Therefore an XML Schema framework comprises a set of reusable and/or specializable elements and types, which form the base schema, and a set of informal conventions describing their reuse and specialization.

The Framework approach has not been employed for implementing tailored metaschemas in XML. Going beyond XML, an example is UML with its extension mechanisms [7]. Thus, instead of extending UML's metaschema at M2, the extension mechanisms provide a means to customize UML at M1.

The Specialized XML Schema approach extends XML Schema with new concepts of a tailored metaschema. To relate new concepts to XML Schema concepts, mechanisms provided by XML Schema itself are used, because XML Schema is defined by an XML schema, which in turn assumes XML Schema at M3. Thus plenty of possibilities exist to relate concepts, such as element composition or type derivation.

An extension with intensional aspects is simply a matter of adding new concepts to XML Schema without the need to relate them to existing concepts. On the contrary, extensional aspects *must* be defined as specializations of existing concepts such as elements and attributes, in order to inherit the extensional semantics of those concepts. An XML Schema validator can interpret new concepts because it is possible to perform a *downcast* according to the principle of type substitutability. Unfortunately, standard XML Schema validators currently do not provide for a plug-able XML Schema, i.e., for a dynamically changeable metaschema, which would be necessary for a downcast.

The Specialized XML Schema approach has not been employed in the XML field yet. Again going beyond XML, so-called open data models have been proposed in the past (e.g., [3]), which consist of a few built-in concepts but which can be extended by additional modeling concepts at M2 for specific application needs.

This paper presented four approaches to implementing tailored metaschemas in XML. Ongoing research focusses on employing each approach to implementing Active XML Schema [8] with XML Schema and on their evaluation to give insight into their respective implications.

References

1. P. A. Bernstein, A. Y. Halevy, and R. Pottinger. A Vision of Management of Complex Models. *SIGMOD Record*, 29(4), 2000.
2. Ralph E. Johnson. Frameworks = (Components + Patterns). *Communications of the ACM (CACM)*, 40(10), 1997.
3. W. Klas and M. Schrefl. *Metaclasses and Their Applications – Data Model Tailoring and Database Integration*, volume 943 of *LNCS*. Springer, 1995.
4. M. Klein et al. The Relation between Ontologies and Schema-Languages: Translating OIL-specifications in XML-Schema. In *Proceedings of the Workshop on Applications of Ontologies and Problem-solving Methods at ECAI*, 2000.
5. OMG. OMG Meta Object Facility (MOF) Specification. `http://www.omg.org/technology/documents/formal/mof.htm`, 2000.
6. E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, 10(4), 2001.
7. J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison Wesley, 1999.
8. M. Schrefl and M. Bernauer. Active XML Schemas. In *Proceedings of the eCOMO Workshop at ER*, volume 2465 of *LNCS*. Springer, 2001.