Representing XML Schema in UML – A Comparison of Approaches

Martin Bernauer, Gerti Kappel, and Gerhard Kramler

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

Abstract. There is a need to integrate XML schemas, i.e., schemas written in XML Schema, into UML-based software development processes. Not only the production of XML schemas out of UML models is required, but even more the integration of given XML schemas as input into the development process. In the model driven architecture, a two step integration is assumed, comprising a platform specific model and a platform independent model. Several approaches already exist addressing the problem of automatically creating a platform specific model for XML schemas. This paper contributes a comparison of these approaches, based on a comprehensive set of transformation patterns supporting creation of a platform specific UML model that is as concise and semantically expressive as possible without loosing XML Schema information.

1 Introduction

UML is being used as de-facto standard for software development, including web applications that exchange XML documents. Therefore a need arises to integrate XML schemas, i.e., schemas written in XML Schema, into UML-based software development processes. Not only the production of XML schemas out of UML models is required, but even more the integration of XML schemas as input into the development process, because standard data structures and document types are part of the requirements.

In the model driven architecture [9], a two step integration is assumed, comprising a platform specific model which abstracts from implementation language details, and a platform independent model which abstracts from technology details. For the platform independent model, plain UML is applied, whereas for the platform specific model, UML tailored to the target technology is employed. An evaluation of existing UML profiles for XML Schema as possible target technology is the main contribution of this paper.

A UML profile for XML Schema must fulfill several requirements. In particular, we are looking for a semantically equivalent representation of an XML schema in UML supporting a bijective mapping between both representations. A solution to this problem has to address the whole range of XML Schema concepts, such that any XML schema can be expressed in UML. Another requirement is to support round-trip engineering, i.e., transformation from XML

© Springer-Verlag Berlin Heidelberg 2004

Schema to UML and back again without loss of schema information. Furthermore, a solution should maximize understandability of semantic concepts by users knowledgeable of UML but not XML Schema. Semantic equivalence is even more important when the UML models are to be used for application code generation, as it will happen in a model-driven development process.

This paper compares five main approaches for representing XML Schema in UML. The features of the approaches are compared based on a comprehensive set of transformation patterns fulfilling the above identified requirements. The patterns have been extracted from a previous effort to define a UML profile ([1]). In the next section, an overview of existing approaches is given, followed by the comparison results and a description of the transformation patterns in the final section.

2 Overview of Approaches

Existing work on representing XML Schema in UML has emerged from approaches to platform specific modeling in UML and transforming these models to XML Schema, with the recognized need for UML extensions to specify XML Schema peculiarities. [2] is the first approach of this kind to modeling XML schemas using UML. Although based on a predecessor to XML Schema, it introduces UML extensions addressing modelling of elements and attributes, model groups, and enumerations that can also be found in following approaches.

The approach by Carlson ([3]) describes an approach based on XMI rules for transforming UML to XML Schema. [3] also defines a UML profile which addresses most XML Schema concepts, except of simple content complex types, global elements and attributes, and identity constraints. Regarding semantic equivalence, the profile has some weaknesses in its representation of model groups, i.e., sequence, choice, and all. Based on the profile defined in [3], a two-way transformation between XML Schema and UML has been implemented in the commercially available tool "hypermodel"[1].

In [10], Provost has addressed some of the weaknesses of [3], addressing representation of enumerations and other restriction constraints, and of list and union type constructors, although the latter doesn't conform to UML.

Eckstein's approach ([5], in german, based on [4]) also defines a profile similar to that in [3], with some enhancements regarding simple types and notations.

Goodchild et al (in [8]) point out the importance of separating the conceptual schema, i.e., the platform independent model, from the logical schema, i.e., the platform specific model, a separation that is not considered in the other approaches. In [8], the logical schema is a direct, one-to-one representation of the XML schema in terms of a UML profile. The profile[2] covers almost all concepts of XML Schema, but several of its representations are not UML conform.

[1] `http://xmlmodeling.com/hyperModel/`

[2] A complete description can be found at
`http://titanium.dstc.edu.au/papers/xml-schema-profile.pdf`

Our approach (in [1]) follows [8] in that it also aim at a one-to-one representation of XML schemas in an UML profile. The approach builds on the existing UML profiles for XML Schema, with some improvements and extensions.

Related work on mapping conceptual models expressed in UML or EER to XML Schema or DTD, has also identified various options for transforming conceptual-level concepts to XML Schema concepts [3,4,6,7,10]. Most of these transformations are, however, not unambiguously applicable in the reverse direction and would thus only be useful in an interactive transformation process, requiring a user's knowledge of the XML schema to be transformed to UML. Therefore, these approaches are not evaluated in this paper, although some of their results have influenced the design of the transformation patterns.

3 Comparison

A comparison of the features of each approach is provided in Table 1, organized along the various transformation patterns as described below. As can be seen, most of the approaches fail to fulfill the requirement of supporting all XML Schema concepts. Furthermore, some approaches represent XML Schema concepts in UML in a way supporting syntactic transformation but failing to provide semantic equivalence. [1] provides a solution satisfying the requirements, with the main improvements being solutions to represent model groups (MG) as well as global elements (EG, EW) and global attributes (AG, AW) in a way more compliant to UML semantics, to represent identity constraints (KY), and to represent simple types in a more concise, UML like way (ST3-4). For technical details on the differences of the approaches it is referred to [1].

Table 1. Comparison of UML profiles by transformation patterns

		CT		ST				EL							MG		KY			GE			
	SC	1	2	1	2	3	4	1	2	EG	EW	AL	AG	AW	1	2	1	2	GA	1	2	AN	NO
[3]	+	+		/	−			−	+		−	+			+								
[10]		+	+	+	−			+	+			+			−								
[5]			−	−	−			+	+		−	+		/	+								−
[8]	+	+	−	/	−			+	+	/	−	+			+			−	−	+		−	
[1]	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Legend: + ... good support / ... violation of UML semantics
− ... incomplete support *space* ... not supported

3.1 Transformation Patterns

Three design goals have guided the design of transformation patterns. First, it must be possible to represent any XML schema in UML, i.e., there must be a representation for each relevant XML Schema concept, in order to *facilitate round-trip engineering* without loss of schema information. Second, a representation of an XML schema has to be such that if the profile specific stereotypes are omitted, the result should - to the extent possible - convey the same meaning, in order to *facilitate understanding by non-XML Schema experts* and to support interoperability with tools not aware of the profile. This goal is also in line with the capability of UML stereotypes, which can only extend but not modify the semantics of UML concepts. Finally, the number of *UML constructs* necessary to represent a certain XML schema should be *minimal*, to improve readability. This goal can be achieved in some situations where UML concepts are more expressive than XML Schema concepts, allowing to represent certain patterns of XML Schema concepts using only one UML concept.

The transformation patterns are organized along the major XML Schema concepts, i.e., schema, complex types, simple types, elements, attributes, model groups (i.e., complex content), identity constraints, group definitions, annotations, and notations. A more detailed description of the transformation patterns can be found in [1].

SC	Represent every schema document as a stereotyped package.
CT1	Represent every global complex type as a stereotyped class.
CT2	Represent every local complex type as a stereotyped class nested into its containing class.
ST1	Represent every simple type that includes an enumeration constraint as a stereotyped enumeration.
ST2	Represent every simple type that does not include an enumeration constraint as a stereotyped primitive datatype.
ST3	Simplification of ST1/2: Merge the representation of a local simple type that is the type of an UML attribute with that attribute.
ST4	Simplification of ST1/2: Merge the representation of a list or union type defined local to a restriction type with the representation of the latter.
EL1	Represent a local element as a stereotyped association role of an association connecting the element's containing model group with the element's type.
EL2	Represent a local element as a stereotyped attribute of the class representing the containing model group if the element's type is a simple type.
EG	Represent every global element like a local element declaration with an additional stereotyped class. Represent its usage by generalizations to that class.
EW	Represent every element wildcard as a multiple classification constraint, indicating that occurrences can be instances of global elements as well.
AL	Represent every local attribute as a stereotyped attribute of the class representing the containing complex type or group.
AG	Represent every global attribute like a local attribute with an additional stereotyped class. Represent its usage by generalizations to that class.
AW	Represent every attribute wildcard as a multiple classification constraint, indicating that occurrences can be instances of global attributes as well.

MG1 Represent a model group as a stereotyped class where the stereotype depends on the group's compositor. Represent its usage by a composition.

MG2 Represent the grammar expressed by a model group tree as a constraint, using a textual notation which covers hierarchical structuring and ordering.

KY1 Represent a key constraint as a constraint attached to the class containing the representation of the element that is the key's scope.

KY2 Represent a key constraint whose selector does not contain union and wildcard steps as a constraint attached to the class selected by the selector.

GA Represent every attribute group as an abstract stereotyped class with the attributes represented by AL. Represent its usage by generalizations.

GE1 Represent an element group as a stereotyped class with the elements represented by EL1 and/or EL2. Represent its model group and usage by MG1.

GE2 Represent an element group as an abstract stereotyped class with the elements represented by EL1 and/or EL2. Represent its model group by MG2 and its usage by generalizations.

AN Represent every annotation as a set of stereotyped comments.

NO Represent every notation declaration as a stereotyped literal in a stereotyped enumeration.

References

1. M. Bernauer, G. Kappel, and G. Kramler. A UML Profile for XML Schema. Technical Report, `http://www.big.tuwien.ac.at/research/publications/2003/1303.pdf`, 2003.
2. G. Booch, M. Christerson, M. Fuchs, and J. Koistinen. UML for XML Schema Mapping Specification. Rational White Paper, December 1999.
3. D. Carlson. *Modeling XML Applications with UML.* Addison-Wesley, 2001.
4. R. Conrad, D. Scheffner, and J. C. Freytag. XML Conceptual Modeling Using UML. In *19th International Conference on Conceptual Modeling (ER), Salt Lake City, Utah, USA*, volume 1920 of *Springer LNCS*, pages 558–571, 2000.
5. R. Eckstein and S. Eckstein. *XML und Datenmodellierung.* dpunkt.verlag, 2004.
6. R. Elmasri, Y. Wu, B. Hojabri, C. Li, and J. Fu. Conceptual Modeling for Customized XML Schemas. In *21st International Conference on Conceptual Modeling (ER), Tampere, Finland*, volume 2503 of *Springer LNCS*, pages 429–443. Springer, 2002.
7. T. Krumbein and T. Kudrass. Rule-Based Generation of XML Schemas from UML Class Diagrams. In *In Proceedings of the XML Days at Berlin, Workshop on Web Databases (WebDB)*, pages 213–227, 2003.
8. A. Goodchild N. Routledge, L. Bird. UML and XML Schema. In *13th Australian Database Conference (ADC2002)*, pages 157–166. ACS, 2002.
9. OMG. MDA Guide Version 1.0.1. OMG Document omg/2003-06-01, `http://www.omg.org/docs/omg/03-06-01.pdf`, 2003.
10. W. Provost. UML For W3C XML Schema Design. `http://www.xml.com/lpt/a/2002/08/07/wxs_uml.html`, August 2002.