A Survey of UML Models to XML Schemas Transformations [⋆]

Eladio Domínguez[1], Jorge Lloret[1], Beatriz Pérez[1],
Áurea Rodríguez[1], Ángel L. Rubio[2], and María A. Zapata[1]

[1] Dpto. de Informática e Ingeniería de Sistemas.
Facultad de Ciencias. Edificio de Matemáticas.
Universidad de Zaragoza. 50009 Zaragoza. Spain.
noesis,jlloret,beaperez,arv852,mazapata@unizar.es
[2] Dpto. de Matemáticas y Computación. Edificio Vives.
Universidad de La Rioja. 26004 Logroño. Spain.
arubio@unirioja.es

Abstract. UML is being increasing used for the analysis and design of Web Information Systems. At the same time, many XML–based languages are cornerstones in the development of this kind of system. As a consequence of the predominance of these languages, there are many works in the literature devoted to exploring the relationships between UML and XML. In this paper we present a survey of current approaches to the transformation of UML models into XML schemas. The study is focused on the case of transformation of UML class diagrams to XML schemas, since we have not found any proposal regarding other kinds of UML diagrams.
Key words: UML model, XML Schema, Model Transformation

1 Introduction

Several recent papers (see [5, 24, 25, 35]) address the increasing interest in using UML in order to analyze and design Web Information Systems. The use of UML within this context has some limitations [24] that have been partially resolved by means of the definition of some UML extensions for web modeling [10]. Nevertheless, the advantages of using a well–established, tool–supported and standard language are beyond doubt. At the same time, more and more XML–based technologies are being used in the implementation and deployment of Web Applications. Therefore, it is not surprising that there are many works in the literature devoted to studying the different relationships between UML and XML. This number will probably grow in the foreseeable future due to the proliferation of Model–Driven Development approaches, which will be applied to the development of Web Information Systems in a natural way.

[⋆] This work has been partially supported by DGI (project TIN2005-05534 and FPU grant AP2003-2713), by the Government of La Rioja (project ANGI 2005/19), by the Government of Aragon and by the European Social Fund

In the present paper we investigate numerous approaches in the literature for the transformation of UML models to XML schemas. One result of our research is that, to our knowledge, there are no approaches devoted to the transformation of UML models to XML schemas other than UML class diagrams. On the other hand, there are many papers that deal with the transformation of UML class diagrams to XML schemas. Because of this, the main contribution of the paper is to provide an exhaustive revision and comparison of such transformations.

The comparison of the different approaches has been realized in a systematic way. On the one hand, and taking as a base the framework proposed in [11], we have compared the different features of transformations proposed by each approach. One conclusion regarding this comparison is that only one approach [15] considers traceability properties, which are considered as desirable features in transformation frameworks such as MDA [1]. On the other hand, we have analyzed which UML class diagram elements are considered by each proposal. A conclusion of this analysis is that the proposal of [26, 27] is the most complete but it does not provide general transformation rules for UML models including the application of some profiles. However, we would stress the necessity for such rules, since the use of profiles is the most common method for tailoring UML to the requirements of specific domains, as for instance Web Information Systems Development. For this reason, we have made a proposal including profiles, with a practical application in a medical context [13].

The paper is structured as follows: in the following section we present an overview of the different UML to XML Schema transformation proposals. A statement about the method of comparison that has been used is presented in Section 3. Sections 4 and 5 are devoted, respectively, to the presentation of the feature–based comparison and the element–based comparison. Finally, Section 6 includes some conclusions and future work.

2 UML Models to XML Schemas Proposals

Due to the widespread use of UML and XML Schema languages, it is no wonder that there are many works in the literature whose goal is to propose a transformation of UML diagrams into XML schemas.

These transformations are far from being trivial, and one reason for this complexity is that UML and XML Schema have several important differences [42]. For instance, UML is primarily used in the early stages (analysis, design) of the development process. However, XML Schema is more implementation oriented and includes programmatic and syntactic details (such as ordering or scoping) that are not generally captured in UML models. Another (almost obvious) difference is that UML is mainly a graphical, visual language whereas XML Schema has essentially a textual representation. Moreover, the most important difference is that while XML Schema is concerned with the *structure* of XML data representing entities and their components and relations, UML allows us to represent not only the *structure* but also the *behavior* of a system, by means of different kinds of diagrams. Because of this, it is not surprising that all the papers we have found regarding the transformation of UML to XML Schema only deal with UML class diagrams (the most important part of UML devoted to the representation of structural aspects). In the rest of the paper we focus our attention on this kind of transformation, carrying out an exhaustive revision and comparison. Let us make now a brief comment about the transformations of UML diagrams different from class diagrams to XML.

The main result in this respect is that all the proposals translate the other kinds of UML diagrams to *XML documents* not to XML schemas. On the one hand, the OMG's XML Metadata Interchange (XMI) [38] defines rules to map MOF [37] metamodels and metadata to XML documents. In particular, it can be used to transform any type of UML diagram into an XMI document. Although XMI is widely used as an interchange format, some works claimed its verbosity, poor readability and compactness [11]. On the other hand, several works are concerned with the transformation of particular kinds of diagrams to XML documents. With respect to *UML sequence* and *collaboration diagrams*, the paper [43] defines a previous XML schema for UML sequence and collaboration diagrams in order to be used as a base for the definition of XML documents. Regarding *UML activity diagrams*, several papers [18, 34] use the XML Process Definition Language (XPDL) to transform activity diagrams to XML documents. As for *UML state machines*, StateChart XML (SCXML) [9] has been proposed by the W3C in order to provide a generic state–machine based execution environment, and XTND (XML Transition Network Definition) [33] is a notation for simple finite state machines.

3 Comparison Method Statement

As we have mentioned previously, henceforth we only deal with the transformation of UML class diagrams to XML schemas. In particular, in this section we explain the method followed to select the papers to be compared, as well as the decisions we have taken to determine the type of comparison to be performed.

With regard to the selected papers, firstly we made a bibliographic search of papers proposing UML to XML Schema transformations, without restricting the search to the area of web information systems. Sometimes the proposals of different authors were very similar, in which case we have selected only one proposal representing all of them. In these cases, we have considered that the papers not included in the comparison do not provide any additional information. Finally, we have analyzed 22 papers in all, grouped into 13 different groups, which provide a general vision of the different published proposals.

It should be noted that the goal of some papers we have analyzed [2, 3, 21, 26–28, 32, 41, 44] is the definition of a UML to XML Schema transformation, whereas other papers [4, 6–8, 12–17, 19, 20, 30, 36, 39, 40] form part of more general projects that do not have this transformation as the only goal. Also, it is worth mentioning that, although several of the analyzed papers explicitly indicate the application domain (database design [27], construction industry [44], e-learning [4, 30], e-business [6–8, 12, 19, 36], evolution architecture [14–17], life sciences [13, 20]), most of the works present their transformation approach in a general way without specifying a specific domain where it could be applied. We have not included a summary of individual works for space reasons.

As for the type of comparison, we have decided to compare the selected papers according to two different dimensions. On the one hand, we have analyzed the *features* of the approach followed to define the UML to XML Schema transformation. In order to do this, we have considered several proposals of model transformation classifications [11, 22, 31]. Of these, we have decided that the taxonomy proposed in [11] is the most rigorous and complete approach. For this reason, we have chosen it to carry out our feature–based comparison.

On the other hand, we have considered it appropriate to compare the papers in terms of the specific UML elements that are taken into account by each proposal. Besides, the strategy followed in order to translate each UML element into XML Schema elements is also analyzed.

We also want to note that, beyond the specific papers we compare in this work, the comparison method we have followed can be used as a general framework to compare other UML to XML Schema transformation proposals.

4 Feature–Based Comparison

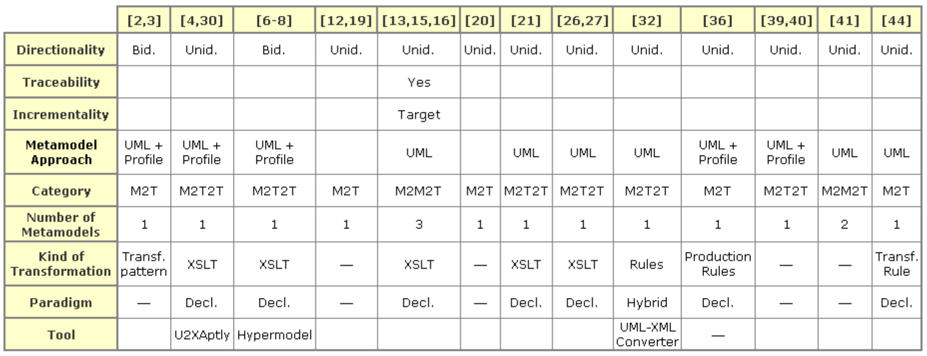
Due to the wide range of varied model transformations proposed in the literature, several recent papers [11, 22, 31] have tried to determine taxonomies or frameworks that help to analyze and compare different approaches. We have considered it appropriate to use these papers in order to perform our comparison. In particular, we have performed the comparison mainly on the basis of the feature model proposed in [11] which makes explicit several possible design choices for a model transformation approach.

Since the type of transformation proposals we want to analyze is not general, some of the features proposed in [11] are not really possible choices. To be precise, in all the analyzed papers:

(1) the proposed transformation is *exogenous*, since the source and target models are not the same,
(2) the proposed transformation involves two *different technological spaces* [31],
(3) regarding the resulting XML schema, although it has a tree–based structure, it is a textual representation of the UML class model, so that we have considered that the type of transformation to be defined is a *model-to-text* transformation.

Aside from these features shared by all the analyzed works, other features that we have used to compare the papers are proposed in [11]. Results of the comparison are summarized in Table 1 which shows the characteristics of each work according to nine different features. In the remaining part of this section, we explain each feature and we comment on the conclusions that can be deduced from each of them.

Directionality. The majority of papers only propose transformations from UML class diagrams to XML schemas (unidirectional), only two approaches, [2,



	[2,3]	[4,30]	[6-8]	[12,19]	[13,15,16]	[20]	[21]	[26,27]	[32]	[36]	[39,40]	[41]	[44]
Directionality	Bid.	Unid.	Bid.	Unid.	Unid.	Unid.	Unid.	Unid.	Unid.	Unid.	Unid.	Unid.	Unid.
Traceability					Yes								
Incrementality					Target								
Metamodel Approach	UML + Profile	UML + Profile	UML + Profile		UML		UML	UML	UML	UML + Profile	UML + Profile	UML	UML
Category	M2T	M2T2T	M2T2T	M2T	M2M2T	M2T	M2T2T	M2T2T	M2T2T	M2T	M2T2T	M2M2T	M2T
Number of Metamodels	1	1	1	1	3	1	1	1	1	1	1	2	1
Kind of Transformation	Transf. pattern	XSLT	XSLT	—	XSLT	—	XSLT	XSLT	Rules	Production Rules	—	—	Transf. Rule
Paradigm	—	Decl.	Decl.	—	Decl.	—	Decl.	Decl.	Hybrid	Decl.	—	—	Decl.
Tool		U2XAptly	Hypermodel						UML-XML Converter	—			

Table 1. Feature–based comparison.

3] and [6–8], consider the possibility of executing the transformations in both directions (bidirectional), so that the XML to UML case is also considered.

Traceability and Incrementality. Traceability is concerned with maintaining an explicit link between the source and the target models of a model transformation [11, 31]. On the other hand, incrementality refers to the ability to update the target model based on changes in the source model [11]. To our knowledge, the proposal developed within our research group [15] is the only work that deals with traceability, maintaining explicit links between the UML and XML models. The aim of this proposal is to use the automatic generated trace for keeping the consistency between the models when evolution tasks are carried out. In this way, target incrementality is achieved since the target model is updated according to the evolution changes performed in the source model.

Metamodel Approach. A metamodel approach is followed in most cases. In these cases, all the papers define UML metamodels and in five cases the metamodel is defined together with a UML profile.

Category. Among the analyzed papers a distinction can be defined on the basis of the fact whether they perform directly a *Model-to-Text* (M2T) transformation from UML Class Diagram to XML Schema or they accomplish some intermediate transformation. In this second case, the analysis of the works has led us to consider two different categories. *Model-to-Text-to-Text* (M2T2T): the UML class diagram is translated first into another representation using some intermediate textual language (XMI in the majority of cases) and then another transformation is performed to obtain the XML schema. *Model-to-Model-to-Text* (M2M2T): an intermediate metamodel is determined so that the UML class diagram is first translated into another model which is an instance of the intermediate metamodel. This second model is then transformed into an XML schema. Two works ([15] and [41]) propose this type of transformation. In both cases, the intermediate metamodel represents a metamodel of the XML Schema language. Let us note that [29] also proposes to use an M2M2T approach but we have not included it in the table because the paper is very short and we do not have enough information to describe it according to the different criteria.

Number of Metamodels. This feature depends on the three previous ones. The M2T and M2T2T papers with a metamodel approach use only one metamodel for the UML class models. The paper [41] with an M2M2T approach defines two metamodels, which are used, respectively, at the conceptual and logical levels. Finally, the M2M2T proposal with traceability, developed within our research group [15], defines three metamodels: two of them for the conceptual and logical levels and a third metamodel representing modeling knowledge with regard to the way in which UML elements are translated into XML Schema elements.

Kind of Transformation and Paradigm. These features are difficult to determine because some authors explain their proposals by means of examples but they do not explain the way in which the transformation is implemented. In these cases, this information cannot be extracted from the papers (the symbol '–' is used to represent this fact). When this feature is mentioned, the great majority of authors implement the transformation by means of XSLT. We have considered, as it is stated in [23], that XSLT is a declarative language, in the sense that the transformation you require is described, rather than providing a sequence of procedural instructions to achieve it. The rest of the papers use different approaches: (1) transformation patterns [2], (2) declarative approach by means of mapping rules [36, 44] and (3) a hybrid approach (declarative and imperative) by means of rules [32].

Tool. According to the information we have gathered from the papers, few works develop a tool based on their proposals. Only three of them indicate the name and the characteristics of the tool, and one work explains that they have a tool but they do not give its name (which is shown by means of the symbol '–').

5 Element–Based Comparison

The goal of this comparison is to analyze which UML Class Diagram elements are considered by each proposal and the strategy followed in order to translate them into XML Schema elements.

The results are summarized in Table 2 indicating in each cell the proposal of each author. Let us note that (1) an empty cell represents that the authors do not comment anything about the element in question, (2) the symbol '–' indicates that the authors propose to translate the corresponding element but they do not explain their strategies and, (3) the symbol 'X' means that the authors explicitly state that they do not translate that element.

The first conclusion that can be extracted from Table 2 is that the proposal of [26, 27] is the most complete approach. Furthermore, in order to compare the way in which each element is considered by each author, we have gathered together the rows in five groups (represented by alternating grey/white colors in Table 2) using a mixture of two criteria. The first and the last group are determined according to the number of works that deal with each element. In particular, the first group includes the elements for which the majority of works propose a translation. In contrast, the elements of the last group are analyzed by either few works or none. The rest of the groups have been determined according

	[2,3]	[4,30]	[6-8]	[12,19]	[13,15,16]	[20]	[21]	[26,27]	[32]	[36]	[39,40]	[41]	[44]
Class	complex Type	element	complex Type + elem.	element	complex Type + elem.	complex Type + elem.	elem.	complex Type + elem.	complex Type + elem.	elem.	complex Type	complex Type	element
Attribute	elem. or attrib.	element	elem. or attrib.	element	element	elem. or attrib.	elem.	element	element	attrib.	elem. or attrib.	elem. or attrib.	elem. or attrib.
Associat.	—	referenc. via associat. element	referenc. via associat. element	referenc. via associat. element	key / keyref referenc.	referenc. with XLink	referenc. via associat. element	key/keyref referenc.	referenc. via associat. element	referenc. via associat. element	key / keyref referenc.	nested element	
Generalizat.	G2	G2, G4	G2, G4	G2	G2	G2	G1	G2	G3	G1	G4		
Built-in data type	simple-Type	simple-Type	simple-Type			simple-Type	simple-Type	simple-Type	simple-Type	simpleT. or string	simpleT. or string	simple-Type	
Abstract class	abstract elem.		abstract elem.	abstract elem.		abstract elem.		abstract elem.	abstract elem.	x			
Multiplicity of attribute		Min/Max Occurs attrib.	Min/Max Occurs attrib.	Min/Max Occurs attrib.		Min/Max Occurs attrib.		Min/Max Occurs attrib.				Min/Max Occurs attrib.	
Multiplicity of associat.			Min/Max Occurs attrib.		Min/Max Occurs attrib.	Min/Max Occurs attrib.	Min/Max Occurs attrib.	Min/Max Occurs attrib.	Min/Max Occurs attrib.		Min/Max Occurs attrib.	Min/Max Occurs attrib.	
Model	root elem.	root elem.	root elem.		root elem.			root elem.			root elem.	root elem.	
Note	—	documen-tation element	documen-tation element	documen-tation element				documen-tation element	documen-tation element	documen-tation element			
User defined data type		complex-Type elem.	derived through generaliz. from simpleT.			complex-Type elem.	simple-Type elem.	complex-Type elem.	derived through generaliz. from simpleT.		derived through generaliz. from simpleT.	derived through generaliz. from simpleT.	
Composit.	child elem.		child element	child element				nested elem. + referenc. elem.	nested elem. + referenc. attrib.	child element	child element		
Restriction in the value of attribute	RVA1, RVA2, RVA3	RVA1, RVA2, RVA3	RVA2, RVA3			RVA1, RVA3		RVA3		RVA3	RVA1, RVA2, RVA3	RVA1, RVA2	
Aggregat.		Ag2	Ag1, Ag2	Ag1				Ag1	Ag3	Ag1			Ag4
Complex attribute		element	complex-Type + elem.			complex-Type + elem.		element					
Package	—		XML name-space					elem. whithout attribute or XML name-space			XML name-space		
Associat. class						complexT. + elem. + references		complexT. + elem. + references			complexT. + elem. + references	Different options nesting based on navigation	
Dependency		Type's attrib. content						x	nested elem. + referenc. attrib.				attrib. relation ="deleg ation"
Derived attribute								attrib.+ elem. with name constraint +condition as fixed.	elem. with name, type attrib. for data type				
Stereotype					S1			attrib					
Method								x	x				

Table 2. Element–based comparison.

to the similarities or differences among the translation strategies followed by the authors. Next, we explain in detail each group.

As we have said, the *first group* includes the elements that are considered by almost every proposal. It is no wonder that the elements of this group are the most widely used UML Class Diagram elements: class, attribute, association, generalization and built–in data type. With regard to the strategy followed to translate classes, attributes and built–in data types, the authors more or less agree. This is not the case of the associations and the generalizations.

As for associations, a study of the possible translation strategies is presented in [26, 27] proposing four different options: (1) nested element, (2) key/key references of elements, (3) references via association element and (4) references with XLink and XPointer. These authors, like our proposal [13, 15], decide to use the second option, but, as can be seen in Table 2, the other three options are more frequently used by the other authors.

Something similar occurs with regard to generalizations. A study of possible translation strategies is presented in [28] proposing three different options (which we have called G1, G2 and G3) and, besides, a fourth option (G4) is proposed in [6–8] and [39, 40]. Of these, the option G2 is the most frequently used by the authors. The meaning of each option is the following:

– G1: All classes are transformed into `complex types` and `elements` of these types. The generalization relation between a class and its subclasses is mapped to containment between the corresponding elements.
– G2: The root class is mapped to an `element` and a `complex type`. The subclasses are mapped to `complex types` derived by extension from the `complex type` of the superclass.
– G3: `Elements` are declared for every class and they are organized in a substitution group.
– G4: The root class is mapped to an `element` and a `complex type`. Specialization classes are mapped to types derived by restriction from the type of the superclass.

The *second group* includes the UML Class Diagram elements for which all the proposals that deal with them propose the same transformation. This is the case of: abstract class, multiplicity of attribute, multiplicity of association, model and note. The elements of the *third group*, user defined data type and composition, are not translated in the same way by the authors but there are very few differences among the proposals.

Different approaches are proposed with regard to the two elements of the *fourth group*. As for the restriction in the value of attribute, in general, the authors decide to propose the use of several options based on the kind of restriction. In the case of aggregation the option we have called Ag1 is the most frequently used. The meanings of the abbreviations used to express the proposed translation strategies of these two elements are the following.

Restriction in the value of attribute

– RVA1: Range restrictions are declared using `min/max Exclusive` and `min/max Inclusive` properties.

– RVA2: Pattern restrictions are mapped to templates for the instances of an attribute.
– RVA3: Enumeration restrictions are translated as enumeration of possible values of attributes.

Aggregation is rendered by

– Ag1: a nested child `element` linked to its remote `element` definition.
– Ag2: a proxy `element` linked to its remote `element` definition.
– Ag3: an XML `element` with `ref attribute` pointing to the referencing class.
– Ag4: an `attribute` named 'relation' with value 'aggregation'.

As we have said before, the elements of the *last group* are considered by none or almost none of the authors. Specially remarkable is the fact that methods are not translated by any author, and some authors even state that methods of a class do not have an equivalent representation in the XML Schema [32]. We also want to note that only two proposals ([26, 27] and [13, 15]) take into account UML stereotypes. The proposal of [26, 27] translates stereotypes without considering their properties, whereas the proposal developed within our research group [13, 15] is the only work that specifies transformation rules for the general application of UML profiles. According to this proposal (denoted as S1 in Table 2), firstly, an XML schema is created for each UML profile and, later, the stereotyped elements are translated into XML elements of a specific complex type (the definition of the complex types is based on the XML schemas previously created).

Finally, we want to note that, due to space reasons, we have not included in Table 2 elements that appear only in one paper. They are: qualified association [27], XOR constraint [27], n-ary association [27], primary key of a class [41], identity constraint (such as unique, pk, exists or isa constraints) [16] and null, non-nillable or not-absence constraints [16].

6 Conclusions and Future Work

In this paper we have presented a survey of current approaches to the transformation of UML models to XML schemas. A conclusion is that there are numerous papers that propose transformations of UML class diagrams into XML schemas. The rest of the UML diagrams are translated in the papers into XML documents but not into XML schemas. We have compared the UML Class Diagram to XML Schema transformations from two different viewpoints: first, considering the features of each one of the transformations and, second, checking what UML Class Diagram elements each proposal takes into account.

With regard to the feature–based comparison, the majority of authors propose a unidirectional transformation (from UML to XML Schema), implemented by means of XSLT, making use of an intermediate structure (instead of performing a direct translation). Generally a UML metamodel for the representation of the UML class diagrams is defined, and some authors propose to use other metamodels for achieving a transformation with better features. In this respect, we consider that the proposal developed within our research group [13, 15, 16] is a noteworthy work, with some remarkable features (traceability, target incrementality and M2M2T approach).

As for the element–based comparison, the majority of works propose a translation for the most widely used UML Class Diagram elements (class, attribute, association, generalization and built–in data type). Nevertheless, there is no agreement about the way of translating associations and generalizations. As another conclusion, the proposal of [26, 27] is the most complete approach, but a weakness of this proposal is that it does not specify complete transformation rules for the general application of UML profiles. However, based on the widespread use of UML profiles as a way to add domain specific semantics to UML class models, we consider it necessary to have at our disposal a general way for translating stereotyped class models. For this reason, in [13] we have made a proposal in this sense.

There are several lines of future work. One important issue is the transformation of other kinds of UML diagrams to XML schemas. For instance, the representation of a UML state machine by an XML schema would allow us to store the execution traces of the state machine in an XML document conforming with that XML schema. We will incorporate the results of this research to our general framework of *evolution* of UML/XML models [13, 15, 16]. In particular, the development of an evolution tool is an ongoing project.

References

1. W. Bast A. Kleppe, J. Warmer. *MDA explained. The Model Driven Architecture: Practice and Promise.* Addison–Wesley, 2003.
2. M. Bernauer, G. Kappel, and G. Kramler. Representing XML Schema in UML -An UML Profile for XML Schema. Technical report, Business Informatics Group, Ins. of Soft. Tech. and Inter. Sys., Vienna University of Technology, November 2003. Available at `http://www.big.tuwien.ac.at/research/publications/papers03.html`. Last visited: June 2007.
3. M. Bernauer, G. Kappel, and G. Kramler. Representing XML Schema in UML - A Comparison of Approaches. In N. Koch and et al., editors, *Procs of the Int. Conf. on Web Engineering (ICWE)*, volume 3140 of *LNCS*, pages 440–444, 2004.
4. A. Bertolino. Initial Recommendations on Advantage Testing Technologies. Technical report, D09, November 2004. Available at `http://www.imsglobal.org/telcert/D09_Testing_Research_v1.0.pdf`. Last visited: June 2007.
5. P. Caceres, E. Marcos, and B. Vela. A MDA–Based Approach for Web Information System Development. In *Proceedings of Workshop in Software Model Engineering (WiSME) in UML'2003*, San Francisco, USA, 2003.
6. D. Carlson. *Modeling XML Vocabularies with UML: Part II*. Available at `http://www.xml.com/pub/a/2001/09/19/uml.html`. Last visited: June 2007.
7. D. Carlson. *Modeling XML Applications with UML: practical e-business applications.* Addison Wesley, 2001.
8. D. Carlson. *Modeling XML Vocabularies with UML: Part III*, 2001. Available at `http://www.xml.com/pub/a/2001/10/10/uml.html`. Last visited: June 2007.
9. J. Carter, J. Barnett, M. Bodell, R. Hosn, and D. Burnett. State chart XML (SCXML): State Machine Notation for Control Abstraction. W3C working draf,

February 2007. Available at `http://www.w3.org/TR/2007/WD-scxml-20070221/`. Last visited: June 2007.

10. J. Conallen. *Building Web Applications with UML*. Addison–Wesley, 2000.
11. K. Czarnecki and S. Helsen. Feature-based Survey of Model Transformation Approaches. *IBM Systems Journal*, 45(3):621–646, 2006.
12. S. Damodaran. RosettaNet: Adoption Brings New Problems, New Solutions. In *Proceedings of the XML 2005 Conference and Exhibition*, Atlanta, November 2005.
13. E. Domínguez, J. Lloret, B. Pérez, A. Rodríguez, A. L. Rubio, and M. A. Zapata. MDD-based Transformation of Stereotyped Class Diagrams to XML Schemas in a Healthcare Context. 2007. Accepted for Publication in CMLSA 2007.
14. E. Domínguez, J. Lloret, A. L. Rubio, and M. A. Zapata. An MDA-Based Approach to Managing Database Evolution. In A. Rensink, editor, *Proceedings of the Workshop Model Driven Architecture: Foundations and Applications*, volume TR-CTIT-03-27 of *CTIT Technical Report*, pages 97–102, 2003.
15. E. Domínguez, J. Lloret, A. L. Rubio, and M. A. Zapata. Evolving XML Schemas and Documents Using UML Class Diagrams. In K. V. Andersen et al., editor, *Proceedings of the Database and Expert Systems Applications (DEXA 2005)*, volume 3588 of *LNCS*, pages 343–352, 2005.
16. E. Domínguez, J. Lloret, A. L. Rubio, and M. A. Zapata. Validation of XML Documents: From UML Models to XML Schemas and XSLT Stylesheets. In T. Yakhno and E. Neuhold, editors, *Proceedings of the Advances in Information Systems Conference (ADVIS 2006)*, volume 4243 of *LNCS*, pages 48–59, 2006.
17. E. Domínguez, J. Lloret, and M. A. Zapata. An Architecture for Managing Database Evolution. In S. Spaccapietra, S. T. March, and Y. Kambayashi, editors, *ER'02 Workshops*, volume 2784 of *LNCS*, pages 63–74, 2002.
18. N. Guelfi and A. Mammar. A Formal Framework to Generate XPDL Specifications from UML Activity Diagrams. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 1224–1231, 2006.
19. B. Heikkinen. *Component-based Modelling with UML and XML-Schemas in RosettaNet*, 2002. Available at `http://smealsearch2.psu.edu/95558.html`. Last visited: June 2007.
20. M. Hucka. *SCHUCS: An UML-Based Approach for Describing Data Representations Intended for XML Encoding.* Sys. Biol. Workbench Develop. Group, 2000.
21. M. Jeckle. Practical Usage of W3C's XML-Schema and a Process for Generating Schema Structures from UML Models. In *Proceedings of the 2nd International Conference of Advances in Infrastructure for E-Business, Science and Education on the Internet, August 2001*, Rome, Italy, August 2001.
22. F. Jouault and I. Kurtev. Transforming Models with ATL. In *Satellite Events at the MoDELS 2005 Conference, LNCS 3844*, pages 128–138. Springer, 2006.
23. M. Kay. *XSLT Programmer's Reference (2nd Edition)*. Wrox Press Ltd., Birmingham, UK, UK, 2003.
24. N. Koch and A. Kraus. The Expressive Power of UML-based Web Engineering. In *Second International Workshop on Web-oriented Software Technology (IWWOST02)*, pages 105–119, Malaga, Spain, 2002.
25. A. Kraus and N. Koch. Generation of Web Applications from UML Models Using an XML Publishing Framework. In *Proc. of IDPT 2002*, Pasadena, USA, 2002.
26. T. Krumbein. *Logical Design of XML Databases by Transformation of a Conceptual Schema.* Master's Thesis (in German), HTWK Leipzig, 2003.
27. T. Krumbein and T. Kudrass. Rule-Based Generation of XML Schemas from UML Class Diagrams. In *Proceedings of the XML Days at Berlin, Workshop on Web Databases (WebDB)*, pages 213–227, 2003.

28. I. Kurtev, K. V. Berg, and M. Aksit. UML to XML-Schema Transformation: a Case Study in Managing Alternative Model Transformations in MDA. In *Proceedings of the Forum on specification and Design Languages (FDL'03)*, Frankfurt, Germany, sep 2003. European Electronic Chips & Systems design Initiative.
29. H. Liu, Y. Lu, and Q. Yang. XML Conceptual Modeling with XUML. In L. J. Osterweil, H. D. Rombach, and M. L. Soffa, editors, *International Conference on Software Engineering (ICSE 2006)*, pages 973–976, 2006.
30. E. Marchetti. Automatic XML Schema Generation from UML Application Profile. *Elektrotechnik und Informationstechnik (e&i) Journal of Springer Verlag*, 122(12):485–487, 2005.
31. T. Mens, P. Van Gorp, D. Varró, and Gabor Karsai. Applying a Model Transformation Taxonomy to Graph Transformation Technology. *Electronic Notes in Theoretical Computer Science*, 152:143–159, 2006.
32. K. Narayanan and S. Ramaswamy. Specifications for Mapping UML Models to XML Schemas. In *Proceedings of the 4th Workshop in Software Model Engineering (WiSME 2005)*, Montego Bay, Jamaica, 2005.
33. G. T. Nicol. XTND - XML Transition Network Definition, November 2000. Available at `http://www.w3.org/TR/2000/NOTE-xtnd-20001121/`. Last visited: June 2007.
34. H. M. Noh, B. Wang, C. J. Yoo, and O. B. Chang. An Extension of UML Activity Diagram for Generation of XPDL Document. In *APWeb*, volume 3399 of *Lecture Notes in Computer Science*, pages 164–169. Springer, 2005.
35. B. Novikov and E. Gorshkova. Exploiting UML Extensibility in the Design Phase of Web Information Systems. In *Proceedings of the Baltic Conference, BalticDB&IS 2002*, pages 49–64, Tallinn, Estonia, 2002.
36. OASIS. *ebXML Business Process Specification Schema v1.01*, May 2001. Available at `http://www.ebxml.org/specs/ebBPSS.pdf`. Last visited: June 2007.
37. OMG. MOF 2.0 Core Final Adopted Specification Document, 2004. ptc/03-10-04. Available at `http://www.omg.org/`.
38. OMG. MOF 2.0 XMI Mapping Specification, v2.1, 2005. Document formal/05-09-01. Available at `http://www.omg.org/`.
39. W. Provost. *Enforcing Association Cardinality*, 2002. Available at `http://www.xml.com/lpt/a/2002/06/26/schema_clinic.html`. Last visited: June 2007.
40. W. Provost. *UML for W3C XML Schema Design*, 2002. Available at `http://www.xml.com/lpt/a/2002/08/07/wxs_uml.html`. Last visited: June 2007.
41. N. Routledge, L. Bird, and A. Goodchild. UML and XML Schema. In Xiaofang Zhou, editor, *Thirteenth Australasian Database Conference (ADC2002)*, Melbourne, Australia, 2002. ACS.
42. F. D. Salim, R. Price, S. Krishnaswamy, and M. Indrawan. UML Documentation Support for XML Schema. In *Australian Software Engineering Conference*, pages 211–220, 2004.
43. J. Singh. Mapping UML Diagrams to XML. Master's Thesis, Jawaharlal Nehru University, New Delhi, 2003.
44. I. C. Wu and S. H. Hsieh. An UML-XML-RDB Model Mapping Solution for Facilitating Information Standardization and Sharing in Construction Industry. In *Proceedings of the 19th International Symposium on Automation and Robotics in Construction*, pages 317–321, Gaithersburg, Maryland, September 2002.