Towards Conceptual Modeling for XML

Erik Wilde
Swiss Federal Institute of Technology Zürich (ETHZ)

Abstract:
Today, XML is primarily regarded as a syntax for exchanging structured data, and therefore the question of how to develop well-designed XML models has not been studied extensively. As applications are increasingly penetrated by XML technologies, and because query and programming languages provide native XML support, it would be beneficial to use these features to work with well-designed XML models. In order to better focus on XML-oriented technologies in systems engineering and programming languages, an XML modeling language should be used, which is more focused on modeling and structure than typical XML schema languages. In this paper, we examine the current state of the art in XML schema languages and XML modeling, and present a list of requirements for a XML conceptual modeling language.

1 Introduction

Due to its universal acceptance as a way to exchange structured data, XML is used in many different application domains. In many scenarios, XML is used mainly as an exchange format (basically, a way to serialize data that has a model defined in some non-XML way), but an increasing number of applications is using XML as their native data model. Developments like the *XML Query Language (XQuery)* [BCF+05] are a proof of the fact that XML becomes increasingly visible on higher layers of the software architecture, and in this paper we argue that current XML modeling approaches are not sufficient to support XML in this new context. Rather than treating XML as a syntax for some non-XML data model, it would make more sense to embrace XML's structural richness by using a modeling language with built-in XML support.

By "XML conceptual modeling language" or "conceptual modeling language with built-in XML support", we do not mean a language supporting XML's syntactic idiosyncrasies, such as the choice of delimiters for attribute values, or maybe even attributes at all. What we mean is support for the fundamental structural richness of XML, which could be described as a mixture of hierarchical (through containment) and relational (through references) structuring capabilities. As an additional property of XML, the document-oriented features such as mixed content should be considered part of the fundamental structures of XML.

According to the classification of WAND and WEBER [WW02], the issue discussed in this paper is a *conceptual modeling grammar* used for constructing *conceptual modeling scripts*, which only is a part of the larger field of a complete conceptual modeling language (the other parts are a *method* for applying the grammar, and the overall *context*). An XML conceptual modeling grammar should have the formal foundations to be able to check conceptual models for certain properties, and to perform automatic or assisted semi-automatic mappings to XML schema languages (or *logical modeling* languages, to keep in line with the taxonomy from the database world).

XML has its origins in a very syntax-centered view of the world, and even though XML increasingly outgrows these origins, they still influence the way people think about XML. A still often-heard claim is that “XML has no data model”, meaning that XML is just a syntax for exchanging trees. Technically as well as historically, this is a correct statement, but when looking at how XML and associated technologies develop, it can be seen that a constantly growing stack of technologies is layered on top of XML, one of these being the *XML Infoset* [CT04]. For all practical purposes, the Infoset is “the data model” of XML today, and it is interesting to see that technologies such as XML Schema[1] and XQuery technically are not really XML technologies, but Infoset technologies. The *SOAP* [Mit03] Web Service data exchange format in its latest version 1.2 has also made the switch from being a technology for transmitting XML from one point to the other, to moving Infosets around.

One of the goals of this paper is to point out that XML is growing to be a real challenge to the world of data modeling dominated by the relational model, and that so far there is no established conceptual modeling language for XML. Database-oriented people often point out that models defined with the established ER, ORM, or UML methods can be simply mapped to XML structures (maybe with some small extensions to the underlying model). Technically, this is correct, but for practical purposes, the mismatch between the relational model and XML structures can be considerable, resulting in awkward XML structures.[2] Only as long as XML is regarded as exchange format only, this mismatch may be accepted and can be alleviated by coming up with powerful and efficient mapping methods.

However, XML is penetrating the applications at an amazing pace. For many programming languages, native support for XML has been implemented at least in research projects (e.g., E4X [Eur04] for ECMAScript and XJ [HRS$^+$05] for Java), database vendors are rapidly moving to support XQuery (which has been integrated into SQL with *SQL/XML* [Int05a]), and standard XQuery APIs (such as *XQJ* proposed by JSR 225) are being developed due to popular demand. An increasing number of databases is supporting native XML storage (which, in most cases, in reality is Infoset storage rather than XML storage). Consequently, viewing the world through relational glasses and simply seeing XML as a possible way to serialize relational data will become less appropriate in the future, and defining an abstraction layer for XML that eventually will be the equivalent of the ubiquitous ER model [Che76] is one of the main challenges for the XML community for the next years. Section 4 gives a short overview over the current state of the art, and it can be seen that there is an urgent need for research in this area.

The likely cause for this absence of a conceptual modeling language is the fact that the two main XML users, database-oriented users and document-oriented users, come from very different backgrounds. While database-oriented users are still using their ER

[1]In the context of this paper, an XML schema with a lowercase ‘s’ refers to the abstract concept of an XML schema language, i.e. a way of defining document classes or types of XML documents. *XML Schema* with an uppercase ‘S’ refers to the W3C schema language [TBMM04, BM04], which from now on we refer to by using the name *XSD*. The overall amount of confusion that has been caused by this poorly chosen name probably is substantial, but until now none of the proposed more specific acronyms such as XSD or WXS has been generally accepted.

[2]Accordingly, the generic mapping of native XML structures to the relational model, as suggested by FLORESCU and KOSMANN [FK99], is technically working, but results is data models which are virtually impractical to work with.

models (often extending them a little) and then map them to XML (treating XML as a logical model only), document-oriented users traditionally are not accustomed to "conceptual modeling" at all, they simply use whatever schema language is available to them, and then define their document model using this schema language, basically directly implementing a logical model. With the emergence of XSD, however, this changed, because in XSD, there are many ways to do the same thing, which compromises the resulting schema's clearness and this makes it rather cumbersome to start with the logical model directly. One of the most important movements in the world of technologies is the question whether XSD is a good way to define XML models, and the most important outcome of a W3C "XML Schema User Experience Workshop" held in the Summer of 2005 was that many participants considered XSD too complex to be used by non-experts directly.

As described by MOHAN and SENGUPTA [MS05], even though there is no real conceptual modeling language today, with the combination of tools and schema languages users are trying to compensate for this by using graphical interfaces and schema generation tools. We argue, however, that the schema language level is the wrong level for XML modeling, because it is carrying a lot of implementation issues with it and often has been developed more with validation in mind than with data modeling. The list of requirements presented in this paper thus is a step towards a conceptual modeling language, and should not be regarded as a plea for a new schema language. Instead, schemas in different schema languages (which we consider to be on the logical modeling layer) could be derived automatically or semi-automatically from an XML conceptual model, if required.

2 XML Schema Languages

Historically, schema languages have been developed for validating documents, with the *Standard Generalized Markup Language (SGML)* [Int86] being the first markup language to invent the DTD schema language. SGML had the requirement for a DTD hard-wired into the language, there was no concept of *well-formed* (i.e., schema-less) SGML. XML introduced the concept of well-formedness, and thus allowed schema-less documents to exist, but it still hard-wired the DTD schema language and some special hooks associated with it (such as the document type declaration) into the specification.

However, the apparent weaknesses of DTDs led to a phase where many improvements to DTDs were proposed, ranging from simply adding namespace and datatype support (*DT4DTD* [BGP00]), to complex object-oriented models (*SOX* [DFH+99]). After a period of consolidation, the W3C's XSD language was proposed, and because of the many other standards that the W3C has since then layered on top of it, this language will remain an important part of the XML landscape for some time to come. XSD is a rather complex schema language, hard to implement and to understand, and only few experts know and use it well. In a study by BEX et al. [BMNS05], it has been show that the majority of XSDs found on the Web are in fact invalid, and that the majority of the remaining valid XSDs are structurally equivalent to DTDs, which means that they do not use any of XSDs new features (most of them are probably generated from DTDs using schema conversion tools).

In an effort to develop smaller and cleaner languages, the *RELAX NG* [Cla01] language is one of the few schema languages which are being used in real-world projects. RELAX NG is similar to DTD and XSD in that it also is grammar-based, specifying a schema as

	SGML DTD	XML DTD	XSD	RELAX NG
attribute order				
simple datatypes	(✓)	(✓)	✓	(✓)
all/interleave	✓		(✓)	✓
exceptions (inclusions/exclusions)	✓			
non-deterministic content models				✓
local element declarations			✓	✓
element/attribute choices				✓
generalized mixed content	✓			✓
ANY/wildcards	(✓)	(✓)	✓	✓

Table 1: Features of XML Schema Languages and SGML DTDs

a set of rules for constructing valid instances. As an alternative approach, the *Schematron* [Int05b] language is a rule-based language, which often is used in conjunction with grammar-based languages to test for additional constraints.

There are a number of schema languages to choose from (even though most of them are rather exotic), and it can be difficult to decide which schema language to choose for a given project. Simplicity and expressiveness of the language are often contradictory goals, with DTDs being simple to understand but relatively poor in terms of expressiveness, and XSD being hard to master but rich in its functionality. RELAX NG has achieved a balance between these goals that is appreciated by a small user community, but still is only used by a tiny fraction of XML users.

The ultimate goal of a schema language as it is discussed here is to define a document class, which means a way to differentiate valid and invalid documents. Table 1 summarizes the most important grammar-based schema languages so far and lists their support of features which might be considered useful for document class modeling. Even though SGML DTDs are not an XML schema language, it is interesting to include this schema language in this comparison for reference.

So far, no schema language has defined mechanisms for constraining attribute order, and therefore it seems to be universally accepted that attribute order in XML documents is insignificant. Changing this would probably introduce far more problems with existing software than the gain in expressiveness could justify.

Simple datatypes like “date” or “time” are required in many scenarios, because they allow to specify the required content in a very specific way. SGML and XML DTDs only support datatypes for attributes, and only a very small set of different types. XSD introduces a large and application-oriented datatype library, which by many users is considered to be one of the most useful features of XSD. RELAX NG does not have a datatype library of its own, but provides a mechanism for reusing existing datatype library, and very often RELAX NG schemas use XSD’s simple datatypes.

The SGML *all* content model had been removed from XML DTDs for reasons of implementation complexity. It has been re-introduced by XSD with severe limitations (it may not be combined with any other model group). RELAX NG also has re-introduced it as *interleave* with slightly altered semantics (and without the restrictions of XSD’s all model group).

SGML and XML DTDs disallowed non-deterministic content models, and XSD did the same, because of backwards compatibility reasons with DTDs, and because of the language's design goal to not only validate documents, but to also be usable for type annotations (which should be done unambiguously). RELAX NG allows non-deterministic content models, which can be very useful for some modeling tasks, but also can be undesirable for type annotations or other unambiguous interpretations of a document.

SGML and XML DTDs only have global element type declarations, so there is no possibility for different definitions of the same element type to exist. In XSD, however, element declarations can be local, so it is possible for the same element name to be used differently in different contexts. However, XSD defines a constraint which requires that occurrences of the same element name in the same context must use the same type. RELAX NG has no such restriction, and therefore it is possible for two elements in the same context to have the same name, but a different definition.

SGML and XML DTDs and XSD treat elements and attributes as something totally different, which means that the attributes of an element and its content cannot have any interdependencies. In practice, however, it is often required for attributes and element content to be interdependent, and RELAX NG allows this kind of generalization, allowing users to specify choices between elements and attributes.

SGML had a rich way of defining mixed content, but due to the "pernicious mixed content" problem, this has been reduced to the much simpler way of simply flagging an element as being mixed or not in XML DTDs and XSD. RELAX NG re-introduces the rich mixed content model of SGML DTDs, and thus allows content models to contain character data only in certain places within an element's content model.

SGML introduced the ANY content model to allow any content within an element, and this feature is available in XML as well. However, this kind of wildcard for an element's content may often be too uncontrolled, because there is no way to limit the wildcard to certain vocabularies. With the introduction of XML Namespaces [BHL99], however, it has become much easier to identify vocabularies, and both XSD and RELAX NG thus allow wildcards to be restricted to content from a certain namespace.

This comparison of the most popular grammar-based schema languages shows that the established languages (DTD and XSD) lack many features which are considered useful in a number of XML scenarios. RELAX NG is a more powerful language and provides almost all features lacking in the other languages.

3 Requirements

Starting from the list of features described in the previous section and from the list of general XML schema language goals defined by the W3C working group [MM99], a list of requirements for an XML conceptual modeling language can be defined. The list presented here shows only the features which can be considered specific for an XML conceptual modeling language, while the basic features required from conceptual modeling languages, described for example by Borgida [Bor85], apply as well.

In the following list, we have listed all features that we consider to be essential for an conceptual modeling language which would map well to XML-oriented technologies, but the features are defined in a way which leaves a lot of freedom for the actual design of such a language.

(1) Formal Foundation — The language must have a solid formal foundation, which makes it possible to reason about the conceptual models. Reasoning about conceptual models may include comparing models (is one model a proper subset of the other), testing models for certain properties (e.g., non-deterministic content), and operations such as removing redundancies or detecting possible candidates for reusing parts of the model.

(2) Graphical Notation — One of the main purposes of a conceptual modeling language is to define a model which is easy to understand, not cluttered with implementation details from underlying technologies, and which can be used by various parties in a project to communicate about the conceptual model. Thus, a graphical notation is of utmost importance, and even though the underlying model must be based on a solid formal foundation, the graphical notation must be intuitive and simple enough to be understandable by people who are not experts in XML technologies. The graphical notation must provide means to collapse or expand parts of the model, and to visualize only parts of the model.

(3) Hierarchical and Referential Structures — XML basically has two different kinds of relations between containers, the hierarchy of the XML tree, and the references most often implemented by attributes. Both kinds of relationships have different constraints and consequences, and while for some modeling purposes it may be only a question of the logical model of how a relationship is being mapped to an XML schema, in other cases users may wish to explicitly specify the type of relationship they would like to use.

(4) Schema Language Mappings — The conceptual model is meant for data modeling and communications within a project. For concrete implementations working with the data, very often it may be necessary to derive a schema from the conceptual model. Mappings should be provided for the most popular schema languages (DTD, XSD, RELAX NG), and while it might not be reasonable to automatically generate schemas from the model, this should be a guided process, ensuring that the schema implements as many constraints from the model as possible. After generating a schema, it should be possible to find out which parts of the model were not mapped to the schema. Notice, however, that it may very well be the case the there never is a schema for a data model, because applications are directly working with the data, based on their understanding from the data model.

With the requirement (3) taken into consideration, it may even be possible to provide mappings to relational structures, if the conceptual model does not explicitly specify features which cannot be easily or reasonably mapped to relational structures. In this case, it would be possible to implement an XML conceptual model using a relational logical model.

(5) Exceptions (Inclusions and Exclusions) — Exceptions are a powerful modeling tool and should be included in the modeling language. As a generalization of SGML's concept of inclusions, it should be considered to add a new type of inclusion which only applies to mixed content, so that included content of this type is only allowed to appear in locations where character data is allowed. As a generalization of SGML's concept of exclusions, there could be two types of exclusions, one only applying to previously included content, and the other to all allowed content (the latter is SGML's definition of exclusions).

(6) Non-deterministic Content — Non-deterministic content can be a very expressive and powerful way of defining content models, but it may also lead to problems of ambiguity. The modeling language should allow non-deterministic content, but should also provide mechanisms to locate potential problem areas, so that users can find possibly problematic content models, and may decide whether to keep them for their simplicity, or replace them with less elegant solutions for the sake of simpler processing.

(7) Treating XML Nodes Consistently — Rather than treating elements and attributes differently, they should be treated consistently, using the same language constructs for both types of nodes. Attribute usage has some additional constraints which do not apply to elements (no order, no repetition, and no complex content), but apart from that, there should be no fundamental difference (RELAX NG sets a good example with treating elements and attributes consistently). Ideally, comments and processing instructions are also included in this generalization, and using the inclusion mechanism from (5), it would be trivial to specify the default XML behavior which allows comments and processing instructions anywhere in the document content.

(8) Model Groups — For combining different XML nodes, model groups must be provided which allow users to freely combine the nodes in any way they like (provided the restrictions on attributes described above). XSD's model groups *sequence*, *choice*, and *all* (without the limitations imposed by XSD) are a well-known foundation and could be used as the available model groups of the language.

(9) Reuse of Content — Reuse of content on all levels must be supported, which means that content may be reused (mainly elements and attributes), and that model groups also may be reused. As an additional way to support reuse, it would be possible to support "inheritance", where a node may inherit the properties (i.e., allowed content) from another node, and add additional content. This would implement a mechanism similar to XSD's derivation by extension (but it should not have the implementation-inspired limitations of XSD's derivation by extension).

(10) Generalized Mixed Content — Continuing the argument from (7), text nodes also should be treated consistently with other nodes, allowing them to appear anywhere in content models. This departure from the simple mixed content approach of XML DTDs and XSD allows greater expressiveness, and is available in SGML DTDs and RELAX NG.

(11) Open Content — While a data model should be as precise as possible, it should also be as flexible possible, and open content (allowing content to appear which is not specified in the data model) is an important requirement in some applications. It should be possible to limit the open content with regard to the namespace name, so that only content from specified namespaces is allowed.

(12) Intra- and Inter-Document Relationships — While reference relationships within documents (such as DTD's ID/IDREF or XSD's identity constraints) are essential for capturing data model constraints, their limitation to intra-document relationships is very restrictive. Instead, more general approaches are required, such as the proposal published by FAN et al. [FKS02]. Ideally, reference relationships as described by requirement (3) should not only be support for inter-document scenarios, but on a generalized foundation, even working across different document classes.

This list of requirements specifies the desirable traits of a conceptual modeling language for XML. Depending on the roots and the focus of a language designer, some of the requirements may be viewed as being more important than others, but regardless of this question of priorities, all these issues must be addressed by a language that may claim to serve as a generic and universally applicable way to express models for XML documents.

Naturally, this could also serve as an conceptual modeling language for SGML, because of the similarities of both languages. However, some of the conceptual language features would probably be XML-specific, such as the support for XML Namespaces, and it would be desirable to not sacrifice any of these features.

4 Related Work

In a survey conducted by SENGUPTA and MOHAN [SM03], a number of conceptual and formal models for XML have been described. Generally, it can be said that many works are heavily influenced by either trying to only model the expressiveness of XSD, or by catering only to the needs of the data-oriented view of XML (for example, not modeling mixed content at all or leaving out the concept of open content).

In Section 4.1, we briefly cover the approaches which could be classified as being mainly from the world of database-driven conceptual modeling. Section 4.2 contains the more formal approaches to modeling XML, which do not concentrate on creating an easy-to-use conceptual model, but formal models which can be used for reasoning about XML models and documents.

4.1 Conceptual Models

Most models of XML which could be called "conceptual models" have been created in an attempt to extend or adapt the relational model to XML's properties. This approach of creating a modeling language has the problem that some of the main properties of XML, such as its hierarchic structure, the ability to specify alternatives, mixed content, and schema-less content, do not fit very well into the traditional database modeling world.

One recently published approach is *Conceptual XML (C-XML)* [ELAK04]. However, from the publication it remains unclear whether it can capture the full complexity of content models, and the support of mixed content is not described. Furthermore, the authors of C-XML claim that it is model-equivalent with XSD, which means that rather than being a conceptual model (i.e., abstraction and/or generalization) for XML or XSD, it is a visual language for XSD. C-XML is still under development and further publications may explain more of the language's design.

FENG et al. [FCD02] describe an attempt to derive XSD schemas from semantic networks. The approach is to view the semantic network as the abstract model, and then to map this model to an XSD schema. Starting from the semantic network approach, however, results in severe limitations of the model. For example, mixed content is not supported at all, and it is impossible to create complex content models which combine different model groups.

ER extended for XML (EReX) [Man04] extends the classical ER model with *categories*, *coverage constraints*, and *order constraints*. Categories allow derivation-like specifications, making it possible to reuse entities and extend their definitions. The additional constraints allow to cater for the more XML-specific concepts, in particular hierarchy and order. An actual schema is written in a language called *XGrammar*, which is a grammar-based language similar to the well-known schema languages. An algorithm is presented how an XGrammar can be derived from an EReX schema. EReX and XGrammar do not support mixed content, so the language is useful for deriving XML grammars from ER-like schemas, but fails to support some XML-specific features.

ER for XML (ERX) [Psa03] is an approach how ER schemas can be derived from XML DTDs. It presents interesting approaches for mapping XML document structures to relational structures, but also views XML as a logical model and solves the problem how to map this onto a conceptual model which is relational in nature.

The *X-Entity* [LSdR03] approach is based on XSD (leaving out any features which are not explicitly part of XSD) and does not address mixed content, the hierarchical structure of the resulting XML structures, or the sequence of elements. It thus supports only a rather limited subset of XML's features.

Extensible Entity Relationship Modeling (XER) [Sen03] is closely aligned with DTDs and XSD (up- and down-translations are described in the paper) and as such can be regarded as a close match to the requirements listed in Section 3. Some aspects, however, are still unsolved, such as the question of wildcards, missing support for exceptions, and the question whether non-deterministic content models are allowed. It is unclear whether there is a formal foundation to the XER model, but the modeling features are a useful contribution to the question of how to visualize an XML model.

While it is quite natural to compare ER models with modeling approaches for XML, because in both cases the focus is on data modeling, this is less obvious for other approaches, which may include data modeling, but also focus on other issues. A well-known example for this is the *Unified Modeling Language (UML)* [Obj04], a highly complex modeling language with thirteen different diagram types (up from nine in UML 1.x). UML's *class diagram* is the diagram type which most closely resembles data modeling (e.g., it has classes with attributes which are connected using relationships). Apart from the general issues concerning the soundness of the underlying formal model raised by SCHEWE [Sch01] and others, it is also observable that the hierarchical nature of XML is not directly supported in UML's basic data model.

However, through more complex features of UML, in particular *stereotypes*, it is possible to extend the basic modeling language with additional features, making it possible to extend UML with arbitrary features. CONRAD et al. [CSF00] have added features to create a mapping between UML class diagrams and XML DTDs, and CARLSON [Car01] has described a similar technique for mapping UML class diagrams to XSDs. However, in both cases, the modeling features in UML have been used to match a specific XML schema language, rather than creating an XML conceptual modeling language.

While it certainly would be possible to find a UML class diagram representation for a given XML conceptual modeling language, it seems that UML per se does not provide the necessary features to be considered an XML conceptual modeling language. On the contrary, since UML's formal foundations are somewhat shaky, it would probably make more sense to design a self-contained XML modeling language with its own well-defined and solid formal foundations, and then create an UML class diagram representation for it as a way to represent and transfer XML conceptual models into the world of UML.

4.2 Formal Models

With other goals in mind, formal models for XML and XML schema languages have been developed as well. The foundations have been laid with the work on tree and hedge grammars by MURATA et al. [MLM01], and this work on grammars as well as newer approaches such as *Heterogeneous Nested Relations (HRN)* [SM03] still is underway, describing XML formally. This formal work is essential for the solid foundation of a conceptual modeling language, and better cooperation between the formal language community and the data modeling community could help to bridge the gap between these two fields.

5 Future Work

The survey and requirements presented in this paper are a first step towards a conceptual modeling language for XML. We believe that the future of XML will inevitably lead to some conceptual model of XML, and a coordinated effort of data modeling experts and formal language specialists would probably lead to a more promising result than the current efforts, which so far have not gained enough traction to produce a promising candidate. With the foreseeable success of XQuery, it will become increasingly important to be able to visualize XML models and to talk about XML models.

6 Conclusions

While XML technologies have been developing rapidly in the last years, there has been different progress in different areas. XML technologies which make it easier for software developers to work with existing XML structures have made a lot of progress, demonstrated by the availability of new APIs or even better integration into the language.

Other areas of XML-related technologies receive less attention, and the area of XML-based data models is one such area. XML's origin as a syntax for structured data is a reasonable explanation for this. With more XML technologies being available for developers, XML becomes increasingly visible within applications, and it is this transformation from a pure exchange syntax to the data model of databases and even applications, which makes it desirable to work with XML-aware tools during the modeling phase of software development.

This paper presents a motivation of why XML should be taken seriously on the conceptual modeling layer, how this compares to the present domination of XML schema languages, and which properties an XML conceptual modeling language should have. By listing a set of requirements for an XML conceptual modeling language, the most important features of such a language are described from the requirements point of view.

There has been some previous work in the area of XML modeling, but most of it lacks support for some essential feature of XML, concentrates on modeling a certain schema language, or lacks the user-friendliness that a conceptual modeling language must provide.

We thus think that there is an open research agenda in the area of XML conceptual modeling, and that a solution for this open problem would be beneficial to XML developers in general, and in particular to software designers working on XML-centric software.

References

[BCF+05] Scott Boag, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie, and Jérôme Siméon. XQuery 1.0: An XML Query Language. World Wide Web Consortium, Working Draft WD-xquery-20050404, April 2005.

[BGP00] Lee Buck, Charles F. Goldfarb, and Paul Prescod. Datatypes for DTDs (DT4DTD) 1.0. World Wide Web Consortium, Note NOTE-dt4dtd-20000113, January 2000.

[BHL99] Tim Bray, Dave Hollander, and Andrew Layman. Namespaces in XML. World Wide Web Consortium, Recommendation REC-xml-names-19990114, January 1999.

[BM04] Paul V. Biron and Ashok Malhotra. XML Schema Part 2: Datatypes Second Edition. World Wide Web Consortium, Recommendation REC-xmlschema-2-20041028, October 2004.

[BMNS05] Geert Jan Bex, Wim Martens, Frank Neven, and Thomas Schwentick. Expressiveness of XSDs: From Practice to Theory, There and Back Again. In *Proceedings of the Fourteenth International World Wide Web Conference*, pages 712–721, Chiba, Japan, May 2005. ACM Press.

[Bor85] Alexander Borgida. Features of Languages for the Development of Information Systems at the Conceptual Level. *IEEE Software*, 2(1):63–72, 1985.

[Car01] David Carlson. *Modeling XML Applications with UML: Practical e-Business Applications*. Addison Wesley, Reading, Massachusetts, April 2001.

[Che76] Peter Pin-Shan Chen. The Entity-Relationship Model — Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[Cla01] James Clark. RELAX NG Specification. Organization for the Advancement of Structured Information Standards, Committee Specification, December 2001.

[CSF00] Rainer Conrad, Dieter Scheffner, and Johann Christoph Freytag. XML Conceptual Modeling Using UML. In Alberto H. F. Laender, Stephen W. Liddle, and Veda C. Storey, editors, *Proceedings of the 19th International Conference on Conceptual Modeling*, volume 1920 of *Lecture Notes in Computer Science*, pages 558–571, Salt Lake City, Utah, October 2000. Springer-Verlag.

[CT04] John Cowan and Richard Tobin. XML Information Set (Second Edition). World Wide Web Consortium, Recommendation REC-xml-infoset-20040204, February 2004.

[DFH+99] Andrew Davidson, Matthew Fuchs, Mette Hedin, Mudita Jain, Jari Koistinen, Chris Lloyd, Murray Maloney, and Kelly Schwarzhof. Schema for Object-Oriented XML 2.0. World Wide Web Consortium, Note NOTE-SOX, July 1999.

[ELAK04] David W. Embley, Stephen W. Liddle, and Reema Al-Kamha. Enterprise Modeling with Conceptual XML. In Paolo Atzeni, Wesley W. Chu, Hongjun Lu, Shuigeng Zhou, and Tok Wang Ling, editors, *Proceedings of the 23rd International Conference on Conceptual Modeling*, volume 3288 of *Lecture Notes in Computer Science*, pages 150–165, Shanghai, November 2004. Springer-Verlag.

[Eur04] European Computer Manufacturers Association. ECMAScript for XML (E4X) Specification. Standard ECMA-357, June 2004.

[FCD02] Ling Feng, Elizabeth Chang, and Tharam Dillon. A Semantic Network-Based Design Methodology for XML Documents. *ACM Transactions on Information Systems*, 20(4):390–421, October 2002.

[FK99] Daniela Florescu and Donald Kossmann. Storing and Querying XML Data using an RDMBS. *Bulletin of the Technical Committee on Data Engineering*, 22(3):27–34, September 1999.

[FKS02] Wenfei Fan, Gabriel M. Kuper, and Jérôme Siméon. A Unified Constraint Model for XML. *Computer Networks*, 39(5):489–505, August 2002.

[HRS$^+$05] Matthew Harren, Mukund Raghavachari, Oded Shmueli, Michael G. Burke, Rajesh Bordawekar, Igor Pechtchanski, and Vivek Sarkar. XJ: Facilitating XML Processing in Java. In *Proceedings of the Fourteenth International World Wide Web Conference*, pages 278–287, Chiba, Japan, May 2005. ACM Press.

[Int86] International Organization for Standardization. Information Processing — Text and Office Systems — Standard Generalized Markup Language (SGML). ISO 8879, 1986.

[Int05a] International Organization for Standardization. Information Technology — Database Languages — SQL — Part 14: XML-Related Specifications (SQL/XML). ISO/IEC 9075-14, July 2005.

[Int05b] International Organization for Standardization. Information Technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based Validation — Schematron. ISO/IEC 19757-3, February 2005.

[LSdR03] Bernadette Farias Lósio, Ana Carolina Salgado, and Luciano do Rêgo Galvão. Conceptual Modeling of XML Schemas. In Roger Chiang, Alberto H. F. Laender, and Ee-Peng Lim, editors, *Proceedings of the 5th ACM International Workshop on Web Information and Data Management*, New Orleans, Louisiana, November 2003.

[Man04] Murali Mani. EReX: A Conceptual Model for XML. In Zohra Bellahsene, Tova Milo, Michael Rys, Dan Suciu, and Rainer Unland, editors, *Proceedings of Second International XML Database Symposium*, volume 3186 of *Lecture Notes in Computer Science*, pages 128–142, Toronto, Canada, August 2004. Springer-Verlag.

[Mit03] Nilo Mitra. SOAP Version 1.2 Part 0: Primer. World Wide Web Consortium, Recommendation REC-soap12-part0-20030624, June 2003.

[MLM01] Makoto Murata, Dongwon Lee, and Murali Mani. Taxonomy of XML Schema Languages Using Formal Language Theory. In *Proceedings of 2001 Extreme Markup Languages Conference*, Montreal, Canada, August 2001.

[MM99] Ashok Malhotra and Murray Maloney. XML Schema Requirements. World Wide Web Consortium, Note NOTE-xml-schema-req, February 1999.

[MS05] Sriram Mohan and Arijit Sengupta. Conceptual Modeling for XML — A Myth or a Reality? In Zongmin Ma, editor, *Database Modeling for Industrial Data Management: Emerging Technologies and Applications*. Idea Group Inc., Hershey, Pennsylvania, 2005.

[Obj04] Object Management Group, Framingham, Massachusetts. *UML 2.0 Superstructure Specification*, October 2004.

[Psa03] Giuseppe Psaila. From XML DTDs to Entity-Relationship Schemas. In Manfred A. Jeusfeld and Óscar Pastor, editors, *Conceptual Modeling for Novel Application Domains, ER 2003 Workshop Proceedings*, volume 2814 of *Lecture Notes in Computer Science*, pages 378–389, Chicago, Illinois, October 2003. Springer-Verlag.

[Sch01] Klaus-Dieter Schewe. UML: A Modern Dinosaur? A Critical Analysis of the Unified Modelling Language. In Hannu Jaakkola, Hannu Kangassalo, and Eiji Kawaguchi, editors, *Information Modelling and Knowledge Bases XII*, volume 67 of *Frontiers in Artificial Intelligence and Applications*, pages 185–202. IOS Press, Medford, New Jersey, 2001.

[Sen03] Arijit Sengupta. XER — Extensible Entity Relationship Modeling. In *Proceedings of XML 2003*, Philadelphia, Pennsylvania, December 2003.

[SM03] Arijit Sengupta and Sriram Mohan. Formal and Conceptual Models for XML Structures — The Past, Present, and Future. Technical Report 137-1, Indiana University, Information Systems Department, Bloomington, Indiana, April 2003.

[TBMM04] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. XML Schema Part 1: Structures Second Edition. World Wide Web Consortium, Recommendation REC-xmlschema-1-20041028, October 2004.

[WW02] Yair Wand and Ron Weber. Research Commentary: Information Systems and Conceptual Modeling — A Research Agenda. *Information Systems Research*, 13(4):363–376, December 2002.