The Case for Conceptual Modeling for XML

Arijit Sengupta[1] and Erik Wilde[2]
[1]Wright State University
[2]ETH Zürich (Swiss Federal Institute of Technology)

February 2006 (TIK Report 244)

Available at http://dret.net/netdret/publications#wil06d

Abstract

Because of its success, XML is increasingly used in many different application areas, and is moving towards the center of applications, evolving from an exchange format to the native data format of application components. These developments suggest that similar to other core areas of application design, XML should be designed conceptually before the implementation tasks of designing markup and writing schemas are approached. In this paper, we describe why conceptual modeling will become an important part of the XML landscape, what issues need to be addressed, and what the requirements for a conceptual modeling language for XML are.

1 Introduction

Since its introduction in 1998, XML has become one of the most influential standards in the IT industry. XML's spectrum is much wider than originally anticipated, when XML was merely planned to be "SGML on the Web" (the title of the W3C working group which created XML). XML today spans many dimensions, it is used for very small data packets (for example, sensor and actuator messages in building automation) or for encoding genome sequences of several gigabytes. It is used as a pure over-the-wire encoding, but it also increasingly penetrates applications through a multitude of related technologies, it is even starting to get built into programming languages [22, 28]. It is used in contexts where data models already exist and modeling methodologies have been established, or it is used on an ad-hoc basis because XML-related technologies are wide-spread, easy to use, and thus have a low barrier-to-entry.

In this paper, we claim that despite XML's success in a multitude of areas, so far no conceptual modeling language has evolved, and we claim that such a modeling language would be beneficial in many scenarios. XML is moving ahead towards the core of applications, and the closer XML gets, the more components will have to deal with XML, and a concise and formally sound conceptual model would greatly increase XML's utility for building software. Already today, many developers are struggling with XML Schema's complexity and its inappropriateness as a conceptual model.

In a recent study [14], it has been shown that conceptual modeling is used by many software developers, mainly for the purpose of database design, but also for software engineering. The most frequently used methods were ER diagramming, data flow diagramming, systems flowcharting, workflow modeling, RAD, and UML. While the more complex methods such as RAD and UML may only be used in bigger projects, at least some kind of conceptual modeling is used in many cases. Right now, there is no established way how XML-centric software projects could on the one hand use conceptual modeling, and on the other hand create a model which also contains XML-specific features.

In this paper, we first briefly look at the different view that different users of XML, possibly having backgrounds, may have (Section 2). We then move on to describe the consequences of these different views in the form of open questions that need to be solved for a conceptual modeling language for XML to become a reality (Section 3). We follow with some requirements, structured into functional and non-functional issues (Section 4). After this, we give a brief overview of two existing approaches and the general approach we envision for designing a conceptual modeling language for XML (Section 5). We then conclude the paper with listing related work and identifying future work issues.

2 Views of XML

One of the main reasons why so far there have been no coordinated efforts to come up with a true XML conceptual modeling language probably is the wide diversity of XML usages. On the one hand, XML may appear in rather low layers in applications where it is simply used to provide an encoding for structured data. On the other hand, XML-centric applications may have a data model that is built around XML and they use this data model using different kinds of XML-oriented software components, for example XSLT-based transformers or XQuery-based querying. In the latter case, ironically, data may never really be encoded in XML syntax in some serialized document. Instead, the software components rely on the XML-based data model and data transfer between components may use any bilaterally negotiated encoding, for example as a binary structure for better performance.

As can be seen, XML can be used on very different layers of a software architecture, and if it is purely used as an encoding for instances of an otherwise specified model, there is no need for an conceptual XML model to exist. All that is necessary are mappings of the application model and the XML encoding. An example for this kind of XML-as-syntax approach is *XML Metadata Interchange (XMI)* [40], which is often mistaken as a way for modeling XML instances. Instead, XMI is a way for serializing MOF-compliant models (i.e., instances of MOF-compliant metamodels). As such, XMI does not have anything to do with XML itself; it is simply using XML as a syntax for encoding structured information (in this case, a model). The same can be said for *ORM-ML* [15], which is only a way of encoding ORM models using XML.

Thus, when talking about *conceptual modeling for XML* we mean models which are designed to describe and cover the full extent of XML. Whether this full extent should be XML 1.0 [7], namespace-compliant XML 1.0 [6], the Infoset [13], the XPath 1.0 data model [11], Infoset+PSVI [51], or the XQuery 1.0 and XPath 2.0 Data Model [25] remains to be seen. Currently, the XQuery 1.0 and XPath 2.0 Data Model seems to be the most promising candidate for becoming the de-facto data model for XML, in particular because XQuery and the next version of SQL/XML (informally named SQL/XML:2007) are both based on this data model.

Apart from the question of whether or not XML applications require the use of a conceptual model and whether it should be based on a specific flavor of existing data models, there also is an interesting diversity of views of XML. Since data modeling is not a totally objective and fully defined process, it is heavily influenced by the experience and knowledge of the person creating a model. There are many different flavors in data modeling, but the two most important which are always identified in the context of XML are people with a history in (relational) databases, and people with a history in document processing.

The most striking difference between these two domains is that data within a database is tightly connected with the overall structure of the database; it cannot be extracted from the database without a loss of information, data always co-exists with the schema. In contrast, most document formats use self-contained document models, so that each document (fragment) contains all the information which is necessary to process it. This difference is the reason why people with a database background need to get used to the self-describing nature of XML and the fact that there may not even be a model (i.e., a schema), while people with a document-processing background do not find this unusual in any way.

Apart from this difference in relationship between data and the model, there are other differences between the two worlds.

2.1 Database Background

The database world has a well-established methodology of different modeling layers, in most cases these are the *conceptual*, the *logical*, and the *physical* layer. Starting from this world-view, it might be tempting to stick to the established conceptual modeling methodology and view XML as a different way to represent data on the logical layer, so XML becomes an implementation of a conceptual model.

Conceptual Model		ER*-Models	???
Logical Model	DDL	SQL	XML Schema
	DML	SQL	XQuery

Figure 1: Modeling Layers and Languages

Such a mapping from a conceptual model from the relational database world to XML structures is not very hard to define (because XML's tree structure is more versatile than the table-oriented relational model), and several mappings have been proposed [26, 30, 34], some of them based on ER-models, other on table structures. Based on the modeling layers and languages shown in Figure 1, these approaches present a path from the relational realm (conceptual or logical layer) to the logical layer in the XML realm.

Reverse approaches have been presented as well, taking an existing XML schema and mapping it to a relational model, either conceptual or logical [17, 27, 31, 37, 42]. In both cases, however, the two worlds are mixed, and depending on the particular approach, this for example may lead to XML structures which are awkward to deal with. While technically this process results in XML structures containing all the information specified in the model, working with this XML with XML technologies such as XQuery, XSLT, or DOM can become very cumbersome.

The other and equally important problem is that any model from the relational world naturally takes only into account the features inherent to the relational approach, while XML features are not supported at all. Among these features are important concepts such as mixed content, complex content models including alternatives, and the general idea of hierarchically structured data. Any modeling language which should be used to exploit the full power of XML should support these features, and thus the mapping approach is an interesting approach for migration and integration scenarios, but not appropriate for XML-centric scenarios because there is no relational legacy.

Thus, while the world of relational databases has a strong background in using conceptual modeling, it also has led many people to believe that the relational-oriented ER-models that have been successfully used for 30 years are the only way to do conceptual modeling. This, however, is only true if the implementation target is a relational database system, and for other targets (such as XML), there may be a model impedance mismatch when employing relational-oriented conceptual models for XML-oriented logical models.

2.2 Document Background

The majority of the database world has successfully lived with the relational model for the last 20 to 30 years. In document engineering and processing, however, this model has never really caught on. In document engineering, the established practice is to use schema languages for defining documents on the logical level, and conceptual models are seldom used systematically. Renear et al. [44] have noted that this may lead to problems because "both content developers and application engineers must rely upon prose documentation, or, worse, conjectures about the intention of the markup language designer." Thus, having a higher level model in document engineering often also would be useful.

One of the main obstacles in document engineering is that very often document models are much more open than traditional relational models. Document schemas often allow a multitude of different combinations of the basic building blocks of the markup vocabulary. This kind of model thus is easier to describe and understand than most relational models, because the variability between different instances is high. Lee et al. [35] discuss the practical and fundamental problems of mapping semistructured models to relational databases. For restrictive document schemas (basically, those that could be mapped easily to relational models), it is much easier to understand and specify the semantics of all possible combinations of structural elements.

Many of XML's features that database-oriented users find minor or not really important are central to document engineering. Mixed content, open content (the ability to allow markup from other vocabularies, which may not be known in advance), recursive content models, and the ability to specify complex content models are indispensable from a document engineering viewpoint, and these features must be accessible on any modeling layer. So any conceptual modeling language not providing access to these features will not be usable for document engineering.

A last interesting difference between database and document views is that in most document engineering applications, there exists the concept of a document as a self-contained entity. It may have links to other documents or to other content, but it is more or less a stand-alone entity. In XML, this is often implemented in the sense that these entities are implemented as documents, and any connections with other entities are modeled with linking mechanisms such as XLink [16]. While these inter-document links are out of scope of any established XML schema language, a conceptual modeling language should be able to cover the full spectrum of a document engineering model, supporting the concepts of self-contained documents, as well as intra- and inter-document associations.

3 Open Questions

Dfferent views of XML and different applications of XML make it hard to define one conceptual model of XML which is appropriate and useful for all or at least the majority of XML users. One of the main challenges of the conceptual model thus is to find the set of requirements which is best suited to create a language which on the one hand is not overly complex, but on the other hand flexible and user-friendly enough to be of utility for a large share of XML users.

Most approaches so far (as presented in Section 6) have their background in a very specific world-view and the goal to improve the support for XML with that world-view in mind. This world-view often is an existing conceptual model which then is extended to cater for some XML-specific features. In most cases, the selection of supported XML-specific features is directly driven by the application scenario. For example, an astonishing number of XML-oriented extensions of the ER-model do not support mixed content, even though this is one of the core features of XML.

The core questions that need to be approached when working towards conceptual modeling for a large share of XML users are the following:

1. *Choose a data model:* What exactly is XML? The actual content of XML documents is not the same for all XML users, but the XQuery 1.0 and XPath 2.0 Data Model [25] is becoming the universal data model for XML. This, however, goes beyond what can be represented in XML documents, because of the concepts of sequences and the PSVI contributions.

2. *Define a schema formalism:* Based on the chosen data model, a formalism must be defined which is able to describe and constrain instances of the model. The main directions so far seem to be either ER-inspired or based on tree grammars, and this reflects the core of the problem: Choose associations or hierarchies. A truly flexible formalism should be able to cover both aspects.

3. *Design user interfaces:* Finally, the schema formalism should be used as the foundation for user-friendly notations, the most important being a graphical notation and textual notations. It would be wise to follow the example of RELAX NG and its compact syntax and provide XML- as well as non-XML-based textual notations.

While there have been several efforts to work on solutions in all of these areas, to our knowledge so far there has been no attempt to approach all of these fields. Only a coordinated effort will lead to a conceptual methodology which would have a chance to become a widely accepted technology for XML users.

4 Requirements

One of the first questions that we need to answer is what would be the requirements of a conceptual modeling method for XML. In this paper, we concentrate on the premise that

XML is not simply a mechanism for modeling documents, but a ubiquitous format for representing any type of data. As such the requirements from a conceptual modeling methodology for XML will be no different from those of any current conceptual modeling methodology, although some of the nuances of XML are bound to affect the way structures are conceptualized for eventual modeling in XML. CHEN [10] motivates his original presentation of the Entity Relationship approach as a method of unification of different levels of conceptualization of information. He identifies four levels of information representation — (i) concepts of objects and their associations in our minds, (ii) structures of such information actually represented by data, although not in any specific form, (iii) structures with a distinct form of data without any enforced access paths, and finally (iv) structures of data with access-path dependency. CHEN identifies conceptual models such as the Entity-Relationship models to concern with levels i and ii.

The problem we face with modeling XML structures is that a lot of the design is driven by the fact that the underlying representation is in XML, and hence hierarchical. One of the challenges posed in this paper focuses on letting go of the enforced data structures, representation, and access paths.

To be consistent with software engineering methodologies, we present the requirements under two subcategories: functional requirements that are essential for the application to incorporate; and non-functional requirements that would likely be important to achieve, although not absolutely critical.

1. **Functional Requirements**

    R1. Structure and Access-path Independence: As described earlier, one of the biggest challenges in the design of a conceptual model is to ensure that the basics of the model are not influenced by the underlying structure or access paths, but reflects only the conceptual components of the data.

    R2. Reflection of the Mental Model: To be consistent with CHEN's modeling levels, the conceptual model must be consistent with the designer's mental concept of objects and their associations. Although the model we are proposing is ultimately for use with XML, most conceptual structures are not hierarchical, and hence the model should allow non-hierarchical structures.

    R3. Modeling of constraints: The model must provide a methodology for expressing constraints that govern the data - such constraints may include cardinality/arity constraints, uniqueness constraints, ordering constraints, etc. At the functional level, only constraints integral to the data representation should be included.

    R4. Formal development: The model should not be developed ad-hoc but should be based on a formal foundation. This indicates that it should be possible to formally define component of the model and express the underlying concept logically.

2. **Non-functional Requirements**

    For non-functional requirements, we include two types of requirements that are brought forth because of the domain of documents and the nuances documents bring in the model, and also ensure that the model is user-friendly.

User-Centric Requirements Realizing that conceptual models are primarily used for design and presentation, usability of the model would be a high priority, and hence some of the requirements must be from the users' perspective.

R5. Graphical Notation: In order to be user-friendly, the model must use consistent graphical notations for model components that can be easily created using specialized or generic design tools.

Document-Centric Requirements Of course, since the model is ultimately for XML documents, some document-specific and XML-specific characteristics may need to be explicitly handled which can be specified with or without the use of constraints.

R6. Textual nuances: For representing documents that may require textual characteristics such as mixed content, reusable content, as well as open content should be supported.

R7. XML-specific characteristics: Often it might be necessary to incorporate specific characteristics of XML in the model, *e.g.*, namespaces, parameter entities, references/links.

5 Base Models

The earlier discussion on the functional requirements indicates that a new way of thinking about document structures may be necessary in order to conceptualize the structures. It is true that formal models for XML exist, and some amount of work has gone into the development of the *Document Object Model (DOM)* [36] and the XQuery formal semantics [18] models. However, all of these models use a highly dedicated tree-based structure, and require an access-driven thinking process which we are trying to avoid.

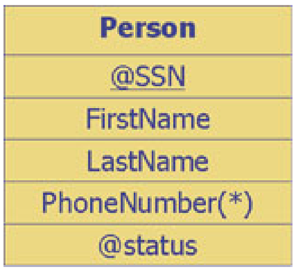
In this section we propose three related concepts that form the basis of a conceptual model for documents. We discuss three basic components of this model — a conceptual component representing the designers' mental image of the structure, a formal component developing the formal basis of the structure, and finally a logical component further formalizing the concepts, their associations, and constraints on them.

5.1 Modeling the Mental World

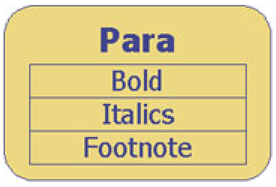
Capturing the mental model of the designers is a difficult problem, specifically within the scope of XML, because of its structural complexity and close link to a hierarchical structure. In order to think different from the tree structure, we make a first effort at developing a conceptual model based on the Entity-Relationship model [10], although the dependency on the ER model may need to be reduced in the future.

In an earlier work [47], we introduced *Extensible Entity Relationship (XER)*, a graphically oriented conceptual modeling technique that attempts to graphically capture different concepts of XML. XER is based on the ER model, although it would be inaccurate to identify XER as an extension of the ER model. XER uses the ER concepts with somewhat varied semantics to enable the capability of capturing complex and heterogeneous types with the potential ordering constraints. The XER model includes all the basic constructs of the ER model, and some new constructs of its own. The primary constructs in XER are briefly described below. Complete description on XER is available from [47].

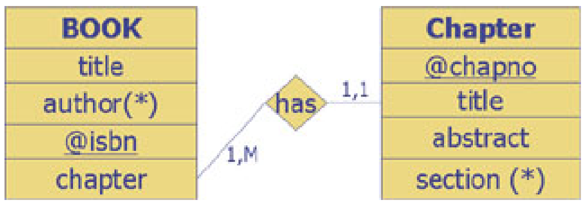
- **XER Entity:** The XER entity is the basic conceptual object in XER. An XER entity is represented using a rectangle with a title area showing the name of the entity and the body showing the attributes. XER attributes are properties of entities and can be atomic or complex, potentially optional or multi-valued. Attributes are ordered by default, and the ordering in the diagram is top-to-bottom. A XER entity can be of the following types:
  - **A. Ordered Entity:** XER entities are ordered by default. An ordered entity indicates that the attributes in the entity must appear in the same order they are presented in the diagram. An example of an ordered entity is shown below:



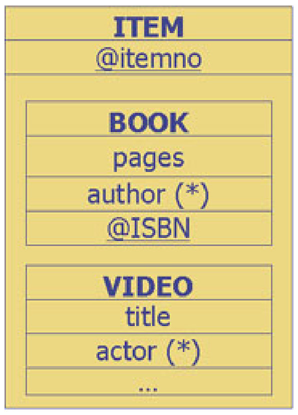
  - **B. Unordered Entity:** Following the path of entity-sets in ER, XER also has support for unordered entities (by placing a question mark (?) in front of the entity name), in which all attributes are required but may appear in a document in any order.
  - **C. Mixed Entity:** Following the document requirement of mixed content models, XER supports the mixed entity, in which both text as well as elements are allowed. The mixed entity (as in the "mixed" attribute in the XML schema) is represented in XER using a solid rounded outer rectangle as shown below:



- **XER Relationships:** Relationships denote a connection between two or more entities, and are introduced in XER when a complex entity contains a complex element as one of its sub-elements. Relationships can be one-to-one, one-to-many or many-to-many. The cardinality of relationships is equivalent to the `minOccurs` and `maxOccurs` tags present in the XML schema. Relationships may or may not be named, and labels along the connectors indicate participation constraints for a relationship and the connecting entity. An example of a XER relationship is shown below:



- **XER Generalizations:** The term “generalization” refers to the concept of having an entity that can have different sub-entities (similar to an IsA hierarchy). In XER, a generalization is represented using a covering rectangle containing the specialized XER entities as shown in below. This is equivalent to using the “xs:choice” tag in XML Schema.



- **Other XER Concepts:** Like the ER model, XER can also have weak entities, ternary relationships, and aggregations having similar semantics. Due to space constraints, these have been omitted. Figure 2 shows a complete XER diagram combining all of the constructs described here.

5.2 Formal Model

Although graphical notations are user-friendly and preferred by designers, for highly complex designs, they often prove to be the bottleneck in the process of understanding the design. Moreover, graphical methods are extremely difficult to use for automated processing, because of their lack of precision, and the presentation overload. It should be noted, however, that the lack of precision is due to the chosen representation (graphical components) and not the modeling methodology itself. Conceptual models can be represented in a more precise manner using a more formal approach based on types, expressions, relations and constraints. There is an abundance of literature recognizing the need for a such an approach to data modeling [3, 49, 50].

Formal modeling methodologies, however, are characterized by reasonably moderate mathematical notation. In the absence of automated support for validation and maintenance, the notation is prone to errors. This in turn limits the use of mathematical notation for conceptual designs in a practical setting.

A first approach to the formal model is based on the non-first normal form (NF2) [29], in which the First normal form (1NF) restriction of the relational is relaxed, and at the

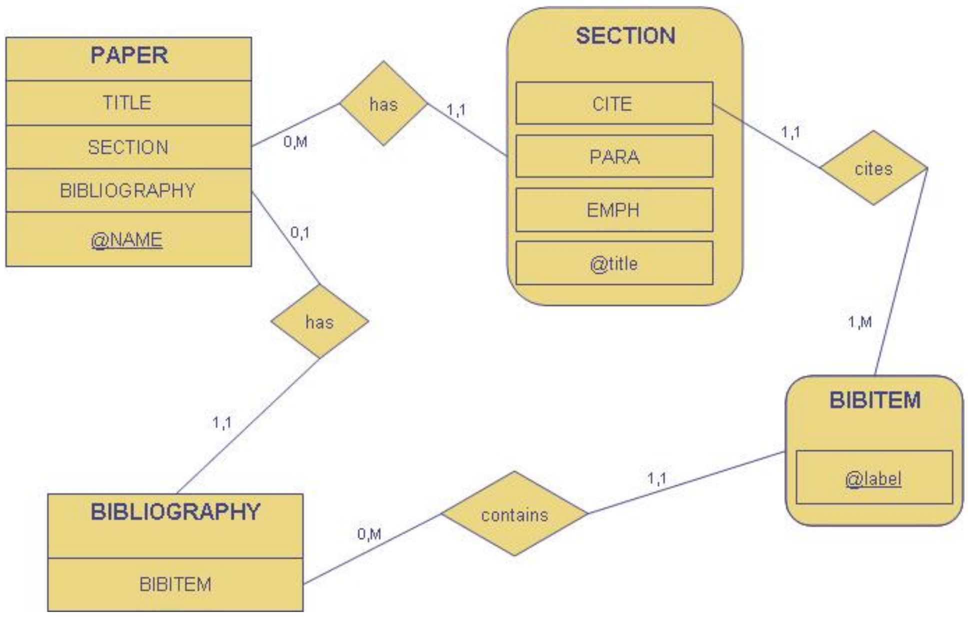


Figure 2: A Complete XER Diagram

same time, additional complexity of variability of structure is introduced. We call this model *Heterogeneous Nested Relations (HNR)*. A discussion on the complete HNR model is outside the scope of this paper, however, a short discussion of the HNR scheme is given below.

HNR scheme Let the set A represent the set of all attribute names. Define an HNR scheme D as follows:

Let A be the universal set of attribute names and scheme names. The HNR scheme for a document is of the form $R < A_1, A_2, \ldots, A_n >$ where each A_i is either an atomic attribute or a document scheme of a sub-document.

Types: The following are types in this language:

- Basic/domain type $\mathcal{D}$
- Tuple type: If $\tau_1, ..., \tau_n$ $n \geq 1$ are types and $A_1, ..., A_n$ are attribute names, then $[A_1 : \tau_1, ..., A_n : \tau_n]$ is a tuple type.
- Set type: If τ is a type then $\{\tau\}$ is a set type.
- Heterogeneous type: If $\tau_1, ..., \tau_n$ $n \geq 1$ are types then $\tau_1 \mid \tau_2 \mid ... \mid \tau_n$ is a heterogeneous type. Objects of such a heterogeneous type τ can be defined recursively as:

$$DOM(\tau_1 \mid \tau_2 \mid ... \mid \tau_n) = \bigcup_{i=1}^{n} DOM(\tau_i)$$

Subtypes: τ_i is a direct subtype of a heterogeneous type $\tau_1 \mid \tau_2 \mid ... \mid \tau_n$. The subtype relationship is the reflexive, transitive closure of the direct subtype relationship.

Paths: To traverse from one type to another type, we use the notion of simple path expressions (SPEs) containing constructs for traversal to named children and descendants from any given type. In the above definition, all A_is are considered children of the type R, and descendant is the reflexive transitive closure of the child relation.

5.3 Modeling Constraints

Constraints are essentially conditions set on the data and their associations. A graphically oriented conceptual model is limited to basic constraints such as cardinality that can be easily added as notational constraints to the graphical language, but in order to model constraints in general, one needs a more expressive medium. We are considering a specification language such as PVS [41] in which all concepts, associations, and even instances can be represented using first-order logic notations based on sets. This aspect of the model will primarily be useful for model verification purposes; something that will significantly influence the impact of conceptual models in the design of XML-based applications.

5.4 Problems and Limitations

As mentioned above, a single graphically oriented modeling method is very difficult to achieve for complex structures, and XER has the limitations of any predominantly graphical method with respect to scaling and expressive ability. An XER diagram, even for moderately sized schema, could be overly complex. The use of parameter entities in DTDs and named complex types in schema simplifies the textual representation of the DTD or schema, but appropriately representing such features in XER would require relationships created between the named type and every entity its used in, thereby making the diagram overly complex. XML model groups such as "ANY" is another problematic aspect, and presumably there is no graphical representation that can appropriately capture the semantics of the "ANY" model group. A formal representation does not have such issues, although it does suffer from the fact that it is less user-friendly, and potentially not suitable for design and presentation purposes. One question may naturally arise - why we need to use a formal method (or its equivalent in a specification language) instead of the XML schema language itself. The biggest reason for that is to enable logical processing that will allow automated reasoning and verification of the model, and any constraints posed on the model.

Although individually limited, once a graphical and a formal approach are hinged together, the combined power is sufficient for solving several of the modeling issues we discussed earlier. The merit of our approach is that the basics of the formal model is equivalent to the graphical representation, so any XER diagram can be easily translated to a formal representation, which can then be augmented with the additional constraints for the complete model. A proper design approach will require proper use of both techniques to be stable and expressive.

6 Related Work

When looking at related work, it can be seen that the different views of XML as described in Section 2 and shown in Figure 1 need to be taken into account. For some approaches towards XML models, the problem was mostly how to map the own conceptual model to an XML logical model. Methods for this have been described by Shanmugasundaram et al. [48] and Fernández et al. [26]. For these, XML is merely an implementation issue and does not affect at all the conceptual layer. Strictly speaking, these approaches are not really targeting what we propose in this paper, but they are interesting nonetheless, because they highlight the fact that XML schema languages should not be considered conceptual modeling languages. Similarly, the inverse has also been done, using non-XML databases to store XML data, as described for relational data by FLORESCU and KOSSMANN [27] and for ORM data by KLETTKE and MEYER [31]. In this direction of mapping XML structures to a non-XML logical model, it has become apparent that the mappings, though technically feasible, result in solutions which are inefficient and hard to query.

The other way around, there are also approaches which try to map given XML schemas (which are considered logical models) to some conceptual model. This group of approaches deals with a problem that an application designer using some given conceptual modeling language would like to integrate XML data which is described by an XML schema. Again, in this group of approaches there is no attempt to create a conceptual model with specific support for XML, but even better than the previous group of approaches in this group it becomes apparent how hard it is to map some of XML's native and essential features to existing conceptual modeling languages. The *ER for XML (ERX)* approach by PSAILA [43] derives ER schemas from DTDs.

A third category of approaches could be identified as approaches trying to extend existing conceptual modeling languages to also support XML-oriented features. Depending on the language of choice to start with, the approaches take different ways, but in most cases the starting language is relational in nature and needs to be extended to better cope with XML's properties of hierarchies, mixed content, and complex content models. CONRAD et al. [12] add features to the UML language to create a mapping between UML class diagrams and XML DTDs. ECKSTEIN[1] and ECKSTEIN [19] extend this work by making the transition to XML Schema. CARLSON [9] describes a similar technique for using UML class diagrams to design XML Schemas. The *Extensible Entity Relationship Modeling (XER)* [47] approach is described in Section 5.1 and is an XML-oriented extension of the ER approach. Similarly, *ER extended for XML (EReX)* by MANI [39] and its underlying formalism *XGrammar* [38] are attempts to extend the traditional ER model. The work by BIRD et al. first focused on ORM [5], then on UML [45], in both cases modifying the original conceptual modeling language with constructs to better support the generation of XML Schemas.

As a fourth category, there are formal models which have been created to study some of XML's properties or properties of XML schema languages. While these models often have a solid formal foundation, in most cases they have not been created with the intention of designing a conceptual modeling language. Thus, these models often do not have nice "user interfaces" in the form of user-friendly textual or graphical notations. Graph-based models for XML have been described by several groups, all of the concentrating on the hierarchical nature of XML documents. The *Regular Rooted Graph Grammars* (R^2G^2) [4] approach has been invented mainly for static type checking for the *Xcerpt* [46] query language. It is based on tree grammars and extends the regular tree grammar model with references. What is currently missing in R^2G^2 is support for inter-document references. The *XML Declarative Description (XDD)* approach proposed by WUWONGSE et al. [52] is an attempt to on the one hand describe XML documents and collections of XML documents, as well as XML schemas such as DTD and XML Schema and constraints in general. *X-Entity* by LÓSIO et al. [37] is another approach to extend the ER model with constructs specific for XML support.

Finally, work towards conceptual modeling for XML has begun, but is in its early stages. The *Conceptual XML (C-XML)* approach be EMBLEY et al. [20] looks promising, but is

[1]Who changed his name after being the first author of the previous work [12] under the name of CONRAD.

still ongoing research work and only little information has been published so far. Feng et al. [24] introduce a two-layer approach and call the two layers *semantic* and *schema* (instead of conceptual/logical), the conceptual layer then is based on semantic networks. Their approach, however, is tightly coupled with XML Schema.

Additionally, related work has been done in various areas related to the core idea of conceptual modeling for XML, for example in the area of intra- and inter-document relationships. Fan et al. [23] propose a *Unified Constraint Model (UCM)* for XML, which is an generalization of XML Schema's identity constraints. Working in a similar direction, Buneman et al.[8] propose *Keys for XML*, with a special focus on relative keys.

Since the conceptual model language for XML will provide abstractions which eventually have to be mapped onto concrete markup structures (as defined by XML schemas), it is also interesting to what kind of markup design constitutes "good" markup. Working towards canonicalization for creating "good" XML structures, Embley and Mok [21] have developed an *XML Normal Form (XNF)* for normalizing XML markup. Using the same name but a different approach, Arenas and Libkin [1] also describe how to normalize XML, their approach uses BCNF as the guiding principle.

7 Future Work

In this paper, we propose a direction of research towards the design of a conceptual model for XML structures. The proposed model differs from some of the current work in this area by incorporating different levels of data model development, including structure-independent aspects as well as document-centric aspects. In addition, we have identified some of the research issues in this area, which will need to be addressed.

Once the graphical and formal models are established, this research will need to achieve two main goals. The first, a critical requirement for any XML-based applications, is a methodology for translating the conceptual model to its logical equivalent, and eventually to XML schema, and also for reverse-engineering existing XML applications to generate the conceptual model. The second area of research would involve automated verification of models to ensure that all requirements are appropriately fulfilled and all constraints are properly handled. The work we have started doing on using the PVS system as a specification language platform will allow us to use some of the theorem proving techniques built-into PVS for the purpose of automated verification and validation of models. We expect several chains of research to emerge from this area in the future.

8 Contributions and Conclusions

In this paper, we describe the case for conceptual modeling for XML. We argue that the trend of XML to move towards the "center" of applications (as opposed to a pure exchange format) already results and increasingly will result in an increased demand for modeling capabilities beyond (i.e., on a higher level than) schema languages. While pure mapping approaches from relational models to XML serializations have been sufficient in a world where applications are inherently relational by design, XML-centric approaches to software design require XML support on the conceptual modeling layer.

We claim that current approaches to conceptual modeling for XML have not yet brought forth a convincing candidate. The main reasons for this are that some approaches are specific to their application areas, while others focus too much on keeping the preferred conceptual modeling method of choice (in most cases ER or UML) and just extend it a little to cater for some (and often not all) of XML's specific features.

A successful candidate for a conceptual modeling language for XML would need to be based on a well-defined data model, have a solid formal foundation which caters for XML's hierarchical nature as well as for intra- and inter-document associations, and provide graphical and textual notations which are user-friendly. Because of the extremely wide area of XML applications, we do not believe that such a language can be created by a single group of researchers or XML users. Instead, a coordinated effort (ideally managed by some neutral organization such as the W3C) should be made to bring together interested XML users to work towards such a language.

References

[1] Marcelo Arenas and Leonid Libkin. A Normal Form for XML Documents. *ACM Transactions on Database Systems*, 29(1):195–232, March 2004.

[2] 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*, Shanghai, November 2004. Springer-Verlag.

[3] G. Di Battista and M. Lenzerini. Deductive Entity Relationship Modeling. *IEEE Transactions on Knowledge and Data Engineering*, 5(3):439–450, June 1993.

[4] Sacha Berger and François Bry. Towards Static Type Checking of Web Query Language. In Stefan Brass and Christian Goldberg, editors, *17th GI-Workshop on the Foundations of Databases*, pages 28–32, Wörlitz, Germany, May 2005.

[5] Linda Bird, Andrew Goodchild, and Terry A. Halpin. Object Role Modelling and XML-Schema. In Laender et al. [33], pages 309–322.

[6] Tim Bray, Dave Hollander, Andrew Layman, and Richard Tobin. Namespaces in XML 1.1. World Wide Web Consortium, Recommendation REC-xml-names11-20040204, February 2004.

[7] Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible Markup Language (XML) 1.0 (Third Edition). World Wide Web Consortium, Recommendation REC-xml-20040204, February 2004.

[8] Peter Buneman, Susan B. Davidson, Wenfei Fan, Carmem Hara, and Wang-Chiew Tan. Keys for XML. *Computer Networks*, 39(5):473–487, August 2002.

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

[10] 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.

[11] James Clark and Steven J. DeRose. XML Path Language (XPath) Version 1.0. World Wide Web Consortium, Recommendation REC-xpath-19991116, November 1999.

[12] Rainer Conrad, Dieter Scheffner, and Johann Christoph Freytag. XML Conceptual Modeling Using UML. In Laender et al. [33], pages 558–571.

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

[14] Islay Davies, Peter Green, Michael Rosemann, and Stan Gallo. Conceptual Modelling — What and Why in Current Practice. In Atzeni et al. [2], pages 30–42.

[15] Jan Demey, Mustafa Jarrar, and Robert Meersman. A Conceptual Markup Language That Supports Interoperability between Business Rule Modeling Systems. In Robert Meersman and Zahir Tari, editors, *Proceedings of 2002 Confederated International Conferences DOA, CoopIS and ODBASE*, volume 2519 of *Lecture Notes in Computer Science*, pages 19–35, Irvine, California, October 2002. Springer-Verlag.

[16] Steven J. DeRose, Eve Maler, and David Orchard. XML Linking Language (XLink) Version 1.0. World Wide Web Consortium, Recommendation REC-xlink-20010627, June 2001.

[17] Ronaldo dos Santos Mello and Carlos A. Heuser. A Rule-Based Conversion of a DTD to a Conceptual Schema. In Kunii et al. [32], pages 133–148.

[18] Denise Draper, Peter Fankhauser, Mary F. Fernández, Ashok Malhotra, Kristoffer Rose, Michael Rys, Jérôme Siméon, and Philip Wadler. XQuery 1.0 and XPath 2.0 Formal Semantics. World Wide Web Consortium, Candidate Recommendation CR-xquery-semantics-20051103, November 2005.

[19] Rainer Eckstein and Silke Eckstein. Conceptual Modeling XML-Schemata Using UML. In Janis Grabis, Anne Persson, and Janis Stirna, editors, *Forum Proceedings of the 16th Conference on Advanced Information Systems Engineering*, pages 122–131, Riga, Latvia, June 2004.

[20] David W. Embley, Stephen W. Liddle, and Reema Al-Kamha. Enterprise Modeling with Conceptual XML. In Atzeni et al. [2], pages 150–165.

[21] David W. Embley and Wai Yin Mok. Developing XML Documents with Guaranteed "Good" Properties. In Kunii et al. [32], pages 426–441.

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

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

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

[25] Mary F. Fernández, Ashok Malhotra, Jonathan Marsh, Marton Nagy, and Norman Walsh. XQuery 1.0 and XPath 2.0 Data Model (XDM). World Wide Web Consortium, Candidate Recommendation CR-xpath-datamodel-20051103, November 2005.

[26] Mary F. Fernández, Wang-Chiew Tan, and Dan Suciu. SilkRoute: Trading between Relations and XML. In *Proceedings of the Nineth International World Wide Web Conference*, pages 723–745, Amsterdam, Netherlands, May 2000. Elsevier.

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

[28] 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.

[29] Gerhard Jaeschke and Hans-Jörg Schek. Remarks on the Algebra of Non First Normal Form Relations. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 124–138, Los Angeles, California, March 1982. ACM Press.

[30] Carsten Kleiner and Udo W. Lipeck. Automatic Generation of XML DTDs from Conceptual Database Schemas. In Kurt Bauknecht, Wilfried Brauer, and Thomas A. Mück, editors, *Tagungsband der GI/OCG-Jahrestagung*, pages 396–405, Vienna, Austria, September 2001.

[31] Meike Klettke and Holger Meyer. XML and Object-Relational Database Systems — Enhancing Structural Mappings Based on Statistics. In Dan Suciu and Gottfried Vossen, editors, *The World Wide Web and Databases: Third International Workshop WebDB 2000, Selected Papers*, volume 1997 of *Lecture Notes in Computer Science*, pages 151–170, Dallas, Texas, May 2000. Springer-Verlag.

[32] Hideko S. Kunii, Sushil Jajodia, and Arne Sølvberg, editors. *Proceedings of the 20th International Conference on Conceptual Modeling*, volume 2224 of *Lecture Notes in Computer Science*, Yokohama, Japan, November 2001. Springer-Verlag.

[33] 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*, Salt Lake City, Utah, October 2000. Springer-Verlag.

[34] Dongwon Lee, Murali Mani, Frank Chiu, and Wesley W. Chu. NeT & CoT: Translating Relational Schemas to XML Schemas using Semantic Constraints. In *CIKM 2002: Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 282–291, McLean, Virginia, November 2002. ACM Press.

[35] Mong-Li Lee, Sin Yeung Lee, Tok Wang Ling, Gillian Dobbie, and Leonid A. Kalinichenko. Designing Semistructured Databases: A Conceptual Approach. In Heinrich C. Mayr, Jirí Lazanský, Gerald Quirchmayr, and Pavel Vogel, editors, *Proceedings of the 12th International Conference on Database and Expert Systems Applications (DEXA 2001)*, volume 2113 of *Lecture Notes in Computer Science*, pages 12–21, Munich, Germany, September 2001. Springer-Verlag.

[36] Arnaud Le Hors, Philippe Le Hégaret, Lauren Wood, Gavin Thomas Nicol, Jonathan Robie, Mike Champion, and Steven Byrne. Document Object Model (DOM) Level 3 Core Specification. World Wide Web Consortium, Recommendation REC-DOM-Level-3-Core-20040407, April 2004.

[37] 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.

[38] Murali Mani, Dongwon Lee, and Richard R. Muntz. Semantic Data Modeling Using XML Schemas. In Kunii et al. [32], pages 149–163.

[39] 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.

[40] Object Management Group, Framingham, Massachusetts. *MOF 2.0/XMI Mapping Specification, v2.1*, September 2005.

[41] Sam Owre, John M. Rushby, and Natarajan Shankar. PVS: A Prototype Verification System. In Deepak Kapur, editor, *Automated Deduction — CADE-11, 11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga Springs, New York, June 1992. Springer-Verlag.

[42] Giuseppe Psaila. ERX: An Experience in Integrating Entity-Relationship Models, Relational Databases, and XML Technologies. In Akmal B. Chaudhri, Rainer Unland, Chabane Djeraba, and Wolfgang Lindner, editors, *International Conference on Extending Database Technology — EDBT 2002 Workshop on XML-Based Data Management (XMLDM)*, volume 2490 of *Lecture Notes in Computer Science*, pages 242–265, Prague, Czech Republic, March 2002. Springer-Verlag.

[43] 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.

[44] Allen Renear, David Dubin, and C. Michael Sperberg-McQueen. Towards a Semantics for XML Markup. In Ethan V. Munson, Richard Furuta, and Jonathan I. Maletic, editors, *Proceedings of the 2002 ACM Symposium on Document Engineering*, pages 119–126, McLean, Virginia, November 2002. ACM Press.

[45] Nicholas Routledge, Linda Bird, and Andrew Goodchild. UML and XML Schema. In Xiaofang Zhou, editor, *Proceedings of the Thirteenth Australasian Database Conference*, volume 5 of *Conferences in Research and Practice in Information Technology*, pages 157–166, Melbourne, Australia, January 2002.

[46] Sebastian Schaffert and François Bry. Querying the Web Reconsidered: A Practical Introduction to Xcerpt. In *Proceedings of 2004 Extreme Markup Languages Conference*, Montréal, Canada, August 2004.

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

[48] Jayavel Shanmugasundaram, Eugene Shekita, Rimon Barr, Michael Carey, Bruce Lindsay, Hamid Pirahesh, and Berthold Reinwald. Efficiently Publishing Relational Data as XML Documents. *The International Journal on Very Large Data Bases*, 10(2-3):133–154, December 2001.

[49] A. H. M. ter Hofstede and H. A. Proper. How to Formalize It? Formalization Principles for Information Systems Development Methods. *Information and Software Technology*, 40(10):519–540, 1998.

[50] Bernhard Thalheim. *Entity-Relationship Modeling: Foundations of Database Technology.* Springer-Verlag, Berlin, Germany, 2000.

[51] 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.

[52] Vilas Wuwongse, Kiyoshi Akama, Chutiporn Anutariya, and Ekawit Nantajeewarawat. A Data Model for XML Databases. *Journal of Intelligent Information Systems*, 20(1):63–80, January 2003.