

# Literate Documentation for XML Schema: CAS Proposal

Kevin Reiss  
Proposal to enroll in LIS459  
kmreiss@uiuc.edu

June 18, 2003

## 1 Introduction

XML<sup>1</sup> is a meta-language for defining markup languages. XML allows the user to define a domain-specific markup language for a particular application. This markup language is expressed in a machine-readable form. When an XML markup language is properly used and designed it adds descriptive markup(1) to a document. Descriptive markup defines the logical parts that exist within a particular document instead using markup to define presentational or procedural information relating to how a document will be displayed.

XML and XML-related technologies currently form the basis of the World Wide Web. Most documents on the web are marked up using the XML-related application HTML<sup>2</sup>. XML is also in heavy use as a format for message protocols and data-exchange in distributed web applications. Other XML applications are appearing for everything from Turing-complete programming languages (XSLT<sup>3</sup>) to graphics formats (SVG<sup>4</sup>). While most XML applications are not as exotic these two examples, the number of XML applications is proliferating at a rapid rate.

The best known and, until recently, the only way to define a markup language using XML is the DTD<sup>5</sup>. A DTD provides a way to formally specify, in machine-readable form, the elements of an markup, the patterns that those elements can appear in, and the at-

tributes that those elements can have. XML document instances that are marked up using the elements of a particular DTD can be checked for conformance to that DTD. This process is known as validation.

As XML has become more popular the need for more expressive and powerful formalisms with which to define the elements, attributes, and their patterns of occurrence has arisen. DTDs have proved inadequate for applications that need to provide support for the XML Namespace mechanism and that need provide strong typing of the element and attribute content of XML documents(6). The new group of formalisms that have been developed in response to these needs are referred to as XML schema.

However, XML schema languages still do not satisfy many of the needs of both application developers and encoders. This is because current XML schema languages lack a formal (machine-readable) mechanism for defining the semantic relationships between the elements and attributes in an XML markup language beyond the generic parent-child and sibling relationships that are implicit in the tree data structure that is produced when an XML document is parsed(4).

XML schema deal only with the description of the allowable syntax of a markup language. Any semantics that a markup language designer does explicitly denote reside only in the natural language documentation that is provided as comments within an XML schema. In addition to lacking a formal semantic description mechanism the natural language documentation that is provided with XML schema is currently processed and displayed in a rather ad

---

<sup>1</sup>eXtensible Markup Language

<sup>2</sup>Hyper Text Markup Language

<sup>3</sup>eXtensible Stylesheet Language Transformations

<sup>4</sup>Scaler Vector Graphics

<sup>5</sup>Document Type Definition

hoc manner that is inconsistent across different XML applications. This inconsistency makes assimilating new markup languages difficult for users.

## 2 Major Issues

This project seeks to enhance the usability and expressiveness of XML schemas by experimenting with formal mechanisms for describing the semantics of a markup language. This will enhance the information modeling power of XML. The system developed will also draw upon best practices in software documentation to develop a system that will result in the generation of more useful and consistent natural language documentation for the end-users of a particular markup language. The following four topic areas will form the basis for a literature review of the relevant subject matter and will serve as a guide for designing the system fully described in section 3.2.

### 2.1 Information Modeling

Using XML to create a new markup language creates a model of the class of documents that can be described with a markup language. XML's increased use in data-centric applications also demonstrates that XML is being used as a modeling vehicle for applications outside of the document sphere for which it was originally designed. This types of data-centric applications have typically been modeled using database technologies. XML is also the currently focus of research on the semantic web. The semantic web is the effort to create machine readable semantics using ontologies that describe the meaning of the content of XML documents.

With XML's use as a modeling vehicle in these three areas (documents, data and semantics) there has been much debate over XML's effectiveness as a modeling tool for representing such modeling methodologies as object-oriented design. But the fact that developers are using XML and XML schematics to model complex applications is undeniable. While schemas are adequate for defining syntax the ability to provide formal semantic information about the problem domain of the language designer would im-

prove the modeling power of XML schema. A review of the different information modeling paradigms will be undertaken. Particular attention will be paid to how these different paradigms are being explored in current research on information modeling using XML schema.

### 2.2 Literate Programming

Some XML-related work has touched upon the concept of Literate Programming originated by Knuth(2), these efforts, which include the TEI<sup>6</sup> and Docbook SGML/XML applications, seek to produce both the actual formal schematic and end-user documentation from the same source document. Literate Programming provides a good example of how a markup designer might use an application that provides both end user documentation and XML schema source code for both the syntax and the semantics of a documents components.

Literate Programming languages will be examined. Other means of provided structured documentation, such as Javadoc, will also be reviewed. More importantly the XML-related systems for Literate Programming will be evaluated for useful ideas on how to develop such a system discussed in section 3.2.

### 2.3 Markup Language Designers and Encoders

Much consideration has been given to general documentation needs of both software developers and end-users. Documentation should represent the markup language in a fashion that is useful to both XML application developers and encoders. Application developers are those who must use an XML schema as the basis for developing a processing application for a particular markup language. Encoders are the end-users that encode documents with XML using text or GUI based editing tools.

The work practices and methodologies that have been used by large-scale XML encoding projects will be considered. The actual guidelines and other documentation relating to such projects will be exam-

---

<sup>6</sup>Text Encoding Initiative

ined. This will help inform decisions regarding what types of end-user documentation are needed for XML schema and how such documentation is used. Encoding practices can also provide insight into how machine-readable semantic descriptions of the relationships between the parts of a markup language might make encoding more efficient and effective.

## 2.4 XML Schema Languages

There are a variety of XML schema languages that have different formal roots in the mechanisms they use to define different XML vocabularies(3). They fall into four general categories<sup>7</sup>:

- Grammar-Based *Ex*: SGML DTDs, XML 1.0 DTDs, RELAX NG *Validity Constraints*: Defines an EBNF grammar that governs what content models can appear.
- Rule-Based *Ex*: Schematron *Validity Constraints*: Specifies formal rules that supply constraints on the content and structure of a document instance.
- Object-Oriented *Ex*: W3C XML Schema *Validity Constraints*: Uses object-oriented design to specify the content models of an XML vocabulary.
- Semantic *Ex*: RDF Schema *Validity Constraints*: Defines properties between any types of object. doesn't provide for formal validation of the syntax of a document instance, but could potentially be used as a model for defining the semantics of a document in a machine readable fashion.

Each particular schema language's facilities for describing the syntax and the semantics of a markup language will be examined. The provisions of each XML schema language for creating structured documentation within the schema will also be reviewed. Schema languages that have strong internal documentation schemas will be more easily adapted to the type of Literate Programming system described in section 3.2.

<sup>7</sup>The first 3 categories are taken from Eric Van Der Vlist(6)

## 3 Project Plan

This project will be executed in two parts:

1. A literature review of the relevant issues surrounding the creation, development of and use of literate programming (2.2), information modeling with XML (2.1), the needs and practices of XML language designers and encoders (2.3), and the issues surrounding the different types XML Schema languages that are currently in use (2.4).
2. A demonstration program will be constructed that allows for the documentation of the semantics of an XML markup language in language-independent fashion. This program will use the Literate Programming paradigm.

### 3.1 Literature Review

The literature review will cover the research material and previous development efforts in the areas of XML schema design, Literate Programming systems, XML-based software documentation programs, and other relevant information modeling methodologies. This will create a narrative that encompasses the history of software documentation systems, the history of information modeling using XML and current research on how to enhance the capabilities of XML to better serve the the purposes of documentation and modeling.

### 3.2 Demonstration Program

The demonstration program will be designed for use by the language designer of an XML application. True to the Literate Programming methodology discussed in section 2.2, the markup language designer will specify all three aspects of an XML schema within the same source file: (1) natural language documentation intended for end-users, (2) the formal grammar and syntax of the XML markup language, and (3) the proposed method to define the semantics of the markup language.

This file will be accepted by a processing system that will process all three of these different aspects

of the language and produce machine readable versions of the XML schema (syntax) and the new layer (semantic) produced from the semantic definitions. These types of syntax and semantic layers and their impact on XML processing are currently being explored in the BECHAMEL project(5). It is likely that the types of properties and relations described by BECHAMEL, currently done using Prolog, will serve as the model for the types of semantic declarations that will be defined with this system.

This process will also generate high-quality end-user documentation. It is expected that the new, semantic layer will enhance the end-user documentation by providing a basis for visualizations that could not be produced from the syntax declarations within an XML schemata. Once generated the two machine-readable grammars of the syntax and semantic layers could potentially become part of a processing pipeline within an application that checks both the syntax and semantics of an XML document.

This list contains some of general and specific questions that will need to be addressed throughout the development of the new Literate Programming system:

- Will the literate programming system be language dependent or language independent? Will it only produce one kind of XML schema, or can it be developed in such a fashion that it is general enough to produce any kind of XML schema?
- What particular syntax and logic rules would be best used by markup designers to represent the semantics of a document?
- How and what software will be used to process the added semantic layer within an XML schema?
- How will this new semantic information be used by application developers when it is generated?
- How will the new semantic layer of information be used to enhance end-user documentation?

The project as described is quite broad. It is expected that literature review will help narrow the focus of the project to a particular area of the problems and research areas discussed in this document.

The demonstration program will likely only represent a small portion of the functionality that could be achieved with the system as fully described in this document.

## References

- [1] COOMBS, J. H., RENEAR, A. H., AND DEROSE, S. J. Markup systems and the future of scholarly text processing. *Communications of the ACM* 30, 11 (1987), 933–947.
- [2] KNUTH, D. E. Literate programming. *The Computer Journal* 27, 2 (May 1984), 97–111.
- [3] MURATA, M., LEE, D., AND MANI, M. Taxonomy of xml schema languages using formal language theory. *Extreme Markup Languages 2001*.
- [4] RENEAR, A., DUBIN, D., AND SPERBERG-MCQUEEN, C. M. Towards a semantics for xml markup. In *Proceedings of the 2002 ACM symposium on Document engineering* (2002), ACM Press, pp. 119–126.
- [5] SPERBERG-MCQUEEN, C., HUITFELDT, C., DUBIN, D., AND RENEAR, A. Drawing inferences on the basis of markup. *Extreme Markup Languages 2002*.
- [6] VAN DER VLIST, E. *XML Schema*, 1st ed. O'Reilly and Associates, June 2002.