January 2000

ABCs of XML: The Librarian's Guide to the eXtensible Markup Language

Norman Desmarais
Providence College, normd@providence.edu

Chapter 1

XML and SGML

Before we begin to discuss XML, the eXtensible Markup Language, we need to understand XML's relationship to its parent, SGML (Standard Generalized Markup Language). This study will be brief. Those who want more technical comparisons between XML and SGML should consult the Comparison of SGML and XML technical note at http://www.w3.org/TR/NOTE-sgml-xml. After a brief examination of this relationship and its implications, we consider the Document Type Definition (DTD) which both languages use. We shall consider alternatives to the DTD and tools to build them. We shall also discuss the concepts of validity and well-formedness which determine whether or not an XML processor will be able to understand the document. We shall then explore some of the applications that use XML which are being developed for libraries.

XML is a subset of SGML (ISO 8879:1986 as amended and corrected). Initially conceived for use on the World Wide Web, XML can be used for any type of electronic publication. While SGML is a text processing standard that describes how a document should be laid out and structured, XML is a dialect of SGML that describes the information content of a document.

SGML really didn't catch on very well because it is too complicated, requiring a steep learning curve that corresponds to high costs. People were also reluctant to incur the expenses of hiring a consultant to implement and manage SGML. Instead, they focused on using the HyperText Markup Language (HTML) which, in its pure form, is an application of SGML with a Document Type Definition (DTD). But HTML, as it is used in practice, is mostly presentation oriented. It defines how information is displayed, such as the color or font size of a word. It doesn't say anything about the actual meaning of the word. XML, on the other hand, has nothing to do with display. It only describes information.

HTML has a fixed set of tags; but XML lets users define their own tags, making it much more flexible and vendor independent. In other words, users can create XML documents in one application and use them in another without requiring a conversion. Because XML accommodates a virtually unlimited number of tags, it can describe the information content of a document more precisely. An XML tag can describe the meaning of any word or term, such as a person's name, a product name, date, or whatever.

For example, searching for the term "pocketbook" in HTML could retrieve items dealing with a purse, a billfold or wallet, or a small book because there has been no way to identify the sense or the meaning of the search term. With XML, developers can tag data to distinguish the different meanings of a word

such as "bill" to identify a personal name, a bird's beak, a charge, a paper currency, or a proposed law -- something which is impossible with HTML.

This is particularly important for numbers which have no inherent meaning in themselves but derive their meaning from the context. While 1000 might be a good price, in dollars, to pay for a new state-of-the-art laptop computer, it would be a very bad average number of days to wait for delivery of purchases. The tag that puts the number in context and gives it meaning becomes critical. The tags also have associations or structure that permit finding, manipulating, acting on, and interacting with the data much more easily.

XML Processor

XML was designed to be easy to implement and to work with both SGML and HTML. It is intended to work in such a way that the Web will be able to handle generic SGML in much the same way that is now possible with HTML. XML describes the information content of documents, called XML documents. It also describes, to some degree, how computer programs will process those documents. XML uses a software module called an XML processor or parser to read XML documents and to provide access to their content and structure. The XML processor interfaces with an application, such as a web browser or word processor, which allows further manipulation of the document. XML permits handling data in a variety of formats and from a variety of sources without costly programming or time delays of data conversion.

Data Type Definition and Alternatives

Those familiar with SGML know that SGML documents use a Document Type Definition (DTD) to specify the structure of a document. The acronym is sometimes referred to as a Document Type Declaration or as a Data Type Definition. Some people prefer the latter term because it is broader and more all-encompassing. Sometimes, the term XML-Data Schema is used to avoid any confusion. The XML DTDs or XML Data Schemas can use as data formats the structural relationships between elements and the XML vocabularies which identify those elements used in particular data formats. Only "valid" (more on this later) XML data require a DTD.
The DTD is only needed at the time of document creation and not for browsing. It consists of two parts: a formal part for the computer to understand the structure of the information and an informal part for a person to identify what the information is or what it means. It is not envisioned that people will write their own DTDs from scratch but rather use tools like Microstar Software Ltd.'s Near & Far Designer which is a visual modeling tool for building SGML and XML structures that can also convert DTD formats.

The XML Data Schema performs the function of a DTD and specifies the nature of the various elements in a document. It consists of two parts, still in working draft mode. XML Schema: Structures defines the language of an XML document and proposes facilities for describing the structure and constraining the contents of XML documents. The schema language provides a superset of the capabilities found in XML 1.0 document type definitions. XML Schema Part 2: Datatypes specifies a language for defining datatypes to be used in XML Schemas and, possibly, elsewhere. XML Schemas, then, will probably serve as application-specific DTDs, such as for particular types of business or industry, such as libraries, museums, and so on. They will specify the vocabulary and formalize the syntax for applications, provided the respective organizations can come to agreement. Schemas will use a schema document in standard XML syntax rather than a separate DTD syntax.

MARC

The MARC format could easily serve as the XML Data Schema for library applications (1), as it contains all the information needed by librarians and anybody working in the book industry. In fact, MARC has been referred to as the "grand-daddy of all DTDs." Already about 30 years old, the MARC format is beginning to show its age. The Library of Congress began, in 1995, to consider the feasibility of using SGML to encode USMARC records. It subsequently released MARC DTDs that define USMARC data in SGML format (Appendix 1). Early in 1998, the library announced software to convert between USMARC and SGML (see: http://www.loc.gov/marc/#marcdtd for details.) The objective is to make machine-readable bibliographic data more open and interchangeable in the Internet environment. It would be relatively easy to use or convert SGML into XML so Web pages will not just display the layout, but also be able to interpret the semantic structure of its content. Bibliographic records can be interchanged between XML and MARC without any data loss (see Appendix 2 and 3); and many of the problems associated with the MARC format, including foreign characters, romanization, and authority control, become insignificant (see Appendix 4.
Catalog librarians might want to distinguish between personal authors, joint authors, corporate authors, etc. and title proper, collective title, uniform title, etc., while book dealers putting together a catalog or students preparing a bibliography are just interested in author and title information without any finer distinctions.

The MARC format as an XML Data Schema could serve all of these functions by describing the syntax for this purpose or for exporting information about books, titles, and authors to a relational or object-relational database where

another XML Data Schema could describe row types and key relationships. XML Data Schemas can provide a common vocabulary for ideas which overlap between syntactic, database, and conceptual schemas making all features usable together as appropriate.

Using the MARC format as an XML Data Schema can also serve to specify books by, rather than about, Charles Darwin, for example. Current engines for searching the Web cannot make this distinction because the document structures do not make it. So they mix both types of books together. Likewise, if a researcher wants to distinguish Darwin's manuscripts from printed editions or facsimile editions, this too is difficult.

Because every database describes its data differently, a search engine cannot understand the various schemas which describe how each database is built. XML could easily categorize books in a standard way, using the MARC format, so that agents could search bookstore sites by author, title, ISBN, or other criteria in a consistent way, much like Z39.50 engines do for library catalogs. Unlike with Z39.50, however, XML would not require re-mapping of the data. A searcher could identify only stores that stock a particular title, for example, instead of retrieving all the places which cite or mention it. One could also limit the search only to articles that review that book.

Building DTDs

Logos Research Systems, Inc. developed a small utility that "round-trips" MARC records to "well-formed" XML and back. This utility, called MARCXML, uses a very simple XML format that isn't designed to contain any knowledge of the MARC records other than the basic structure. In other words, because it only encodes the raw structure of the file, it has no built-in understanding of USMARC or any other MARC format and cannot detect improper use of tags during an XML to MARC conversion. It doesn't know what's allowed to repeat, what's obsolete, etc. There is a small DTD; but the converter doesn't require it to rebuild the record. Nor does it perform character set conversions. The resulting XML file can be viewed in a browser or edited in a word processor and then recompiled back into a structurally valid MARC record. The XML parser is built into the converter; so the whole thing runs as one small program. A demonstration of the conversion in both directions can be run from Logos Research Systems's web site at http://www.logos.com/marc. The company also has a free MARCType Parser which parses USMARC records into an annotated human-readable document online.

The Graphical Communication Association Research Institute and the Data Interchange Standards Association plan to build libraries of Document Type

Definitions, Java applets, template scripts, forms, and object definitions to allow businesses to process the components of XML messages.

Microsoft Corp. and IBM have also developed Document Content Descriptions (DCD), written entirely in XML, to replace DTDs. They submitted the proposal to the World Wide Web Consortium (W3C) on July 31, 1998. The W3C will form a working group to consider it and to flesh out the protocol (see www.w3.org/TR/NOTE-dcd). DCDs intend to provide similar functionality to DTDs; but they won't require users to learn XML and a DTD protocol. They will also support other XML-related components such as:

Resource Description Framework (RDF) which provides an infrastructure to support metadata (data about data) across Web-based activities and can be used to describe the content and content relationships available at a particular Web site, page, or digital library;

XML-Schema which proposes facilities for describing the structure and constraining the contents of XML documents.

XML Namespaces which qualifies tag names used in XML documents by associating them with their source and provides a simple method for qualifying certain names used in XML documents by associating them with namespaces.

XPointer (XML Pointer Language) which supports addressing into the internal structure of XML documents. This allows going directly to a specific section, paragraph, table, or other element contained in a document instead of referencing the document as a whole.

"Valid" and "Well-Formed" XML

We have already used the terms valid and well-formed; but we have not yet mentioned what they mean. Consistency is an important part of XML implementation. Each type of industry will have to come to some agreement on the metatags its members will use and develop appropriate DTDs for use by its members. If we are to be able to compare book prices, for example, booksellers will need to use the same tags. Using synonymous terms like price, list price, selling price, retail price, wholesale price, discount price, or charge to describe the same entity will only perpetuate the confusion and hinder data sharing. In the chaotic world of the Web, it may be difficult to achieve that level of agreement.

In the world of XML and SGML, the DTD specifies the tags and their meanings so applications can work with and manipulate the data. Documents that make use of internal or external Data Type Definition files are known as "valid" XML and require an XML parser to check incoming data against the rules defined in

the DTD to verify that the data were structured correctly.

A developer usually decides which parser to integrate into an application or builds one from scratch; so an end-user has no control over which parser to use. There are two kinds of parsers: validating and non-validating. A validating parser “validates” documents by ensuring that they conform to the DTDs specified within them. A non-validating parser only checks that the document has a sound logical structure and syntax ("well formed"). Internet Explorer 4 uses a non-validating parser; but IE5 beta-1 uses a validating parser. Netscape Navigator 5.0 will use a non-validating parser written in C which is faster and more efficient than a Java parser.

Parsers must know what the elements mean and how to process them. (We’ll discuss elements in more detail in the next chapter.) XML allows developers to describe element names in a recognizable manner to avoid conflicts between elements with the same name. For example, some applications may have valid reasons for distinguishing between list price, retail price, wholesale price, and discount price, depending on a company's business rules and markets. Using different tags just to be different, however, will only lead to confusion. The W3C is considering a proposal to make every element name subordinate to a universal resource identifier (URI) to ensure that names remain unambiguous even if chosen by multiple authors.

The creators of the XML format admit that it is primarily intended to meet the requirements of large-scale Web content providers. However, valid XML files can be used outside the Web, such as in an SGML environment, for electronic publishing or as a means to produce print products.

Data sent without a DTD is known as "well-formed." That data can be used implicitly to describe itself. XML encoded data is self-describing and can also be used to model and deliver structured data without any reference to documents. This allows XML's open and flexible format to exchange or transfer information. It also makes XML extremely powerful, as it can be used for any type of information, such as articles, chapters, books, maps, weather forecasts, invoices, purchase orders, order forms, and so on.

XML only requires data to be "well-formed." In other words, its logical structure must appear sound. Its elements must be properly nested with all opening tags subsequently closed. Otherwise, browsers will look for a closing tag until it times out, resulting in slower speed of delivery to end-users.

Applications

XML will be particularly useful for enabling a Web client to interact with two or

more sources of data at the same time. These sources may have different formats or require access through dissimilar interfaces. If they have common tags giving meaning to the data attributes, XML applications will be able to work with them. Search engines will be able to understand and use contextual information when performing a full-text search, producing more relevant results. They will also be able to work across incompatible hardware and software platforms and with incompatible protocols.

OCLC's SiteSearch, for example, can load XML records, build interfaces, and permit powerful cross database searching by using XML templates. The template allows extracting information from a database of records stored in XML format. It can also map MARC records to XML for editing and creation of new records. That information can then be used as a form to populate another record, creating a populated template which can then be displayed in HTML, XML, or another format.

While reading an article, for example, a student can consult a footnote, get an abstract of a reference, verify whether the local library or a remote library has the item, or prepare a purchase requisition – all without leaving one's place in the article. Another example would be the ability to go from a citation in an index or bibliography to the full text of the cited work.

Endeavor Corp. is working on a product called ENCompass which it expects to introduce in the year 2000. It will support various DTDs, such as Dublin Core for digital content, Text Encoding Initiative (TEI) for full text content, and Encoded Archival Description (EAD) for archival material. It will also allow searching across disparate databases with different file structures, terminology, etc. to provide an integrated display of resources regardless of whether they're described in MARC, TEI, EAD, or Dublin Core.

The product will allow users to go from a catalog record citation, a reference or footnote, or an abstract or bibliographic citation to the full text from different aggregators. One could start from a catalog record or a citation to look up an ISBN in *Books in Print* to get ordering information, availability, and pricing. One could also consult an author biography in Who's Who or search related items in other databases.

ExLibris is developing its Aleph search engine which recognizes XML encoding. It is also working on the ability to index based on XML encoding. It plans to support XSL, the eXtensible Stylesheet Language (see chapter 3), when the draft becomes finalized.

Many library system vendors are experimenting with XML applications. Data

Research Associates, for example, is experimenting with passing XML-encoded patron data to OCLC. XML may bring more uniformity to the different ways vendors authenticate clients.

XML can take the strain off Web sites by moving much of the processing load from the Web server to the Web client where a Java applet downloaded to the user's PC, for example, may do the processing. Dynamic HTML also uses this approach; but an applet can do more with XML-tagged content because the data possess much more meaning. This will have a dramatic impact on Electronic Data Interchange (EDI) and electronic commerce which we'll discuss further in chapter 7.

A Web client can interact with multiple views of the same data, unlike in the HTML world where each different view requires a trip back to the server for re-presentation. An XML client can sort, filter, and manipulate data locally, making it much easier to customize a presentation to the needs of the person using the data. A catalog librarian can have one view of the data; a library patron can have another; accounts payable can have yet another. By supporting a greater granularity of information, XML allows individuals to extract relevant information from several sources and reassemble it into any format (data re-purposing).
Intelligent Web agents can also be programmed to find and deliver information to meet profiles tailored to each individual. Those profiles could be customized in terms of both content and presentation. After retrieving the information, the browser could use the tags to omit irrelevant portions and display only those portions of a document most likely to interest the reader. This could reduce information overload and accelerate learning. This capability would also allow individuals to create "custom" documents or reports with the most recent information formatted and delivered any way they want it.

NICEM (National Information Center for Educational Media) has converted its database from a simple relational database management system to an XML-based intranet system in an effort to provide higher-quality new records. By storing the bibliographic records as XML documents, the data are useable by any system that meets standardized criteria.

NICEM has also developed a machine-aided indexing (MAI) application to accompany the new system. The MAI program utilizes the NICEM thesaurus of 4300 subject terms along with 6000 syntax rules written by NICEM editors. It uses natural language processing to read the bibliographic record and then suggest subject terms to the editor. This process takes only seconds and provides the editor with terms he or she can decide to use or ignore. The editor can also select subject terms from the thesaurus to include in the record. MAI

provides for faster indexing and promotes consistency in indexing from editor to editor. XML could also be used to discover new relationships between data, whether in searching or display.

The Association of American Publishers (AAP) and John Wiley & Sons have collaborated on establishing a clearinghouse for metadata in an attempt to establish standardized systems for managing digital book and journal publishing. The Metadata Information Clearing House (Interactive) – MICI -- is an online interactive database that will allow publishers to input comments, questions, and descriptive information about their metadata procedures. It will also let users know who's working on what. It will also allow publishers to identify and locate digital content and to establish links and relationships among different kinds of content..

The MICI project is expected to eventually ensure more comprehensive and accurate search engine results, content reuse, and analysis of customer activity. It will probably also incorporate the Digital Object Identifier (DOI). More information on MICI can be obtained from the AAP Web site (http://www.publishers.org/mici.htm) or from John Wiley's site (http://wileynpt.com/mici).
The AAP also surveys the systems used by publishers to process and transmit online bibliographic and descriptive content to online retailers. This includes such basic information as book jacket images, catalog entries, and author background.

Outlook

XML is not a single, fixed format like HTML, nor is it a replacement for HTML. XML is a metalanguage that lets users design their own markup languages to meet specific application or industry needs. It provides a standard way for describing and exchanging data regardless of its nature, how the sending system stored it, or how the receiving system will use it.

HTML will continue in use for the foreseeable future because of the cost and labor to tag existing documents. The change will be evolutionary; but its impact will be revolutionary. Versions of HTML subsequent to HTML 4.0 will use XML syntax and support XML tags. As Web pages get updated and modified, the code will gradually get converted to XML, producing a richer, more relevant, and better organized Web.

In this chapter, we examined XML's relationship to its parent, SGML and some of its implications. We looked at the Document Type Definition (DTD), alternatives to it, and tools to build DTDs. We discussed the related concepts of validity and well-formedness and explored some library applications that use

XML. Now that we have a foundation, we can turn our attention to the structure of an XML document.

Notes

Lam, K. T. Moving from MARC to XML.
Http://home.ust.hk/~lblkt/xml/marc2xml.html