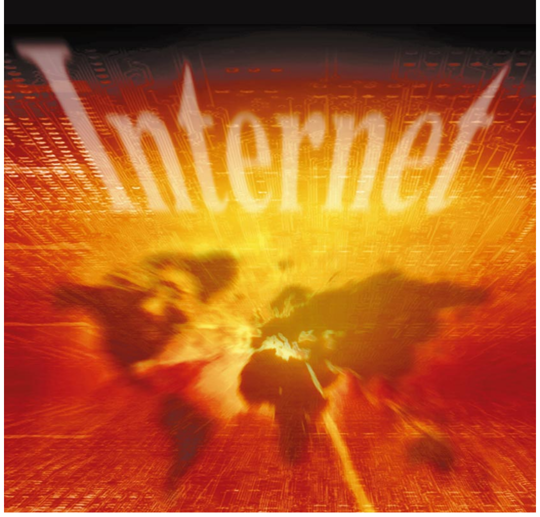
Extensible Markup Language is powerful, not all-emcompassing. Learn how to separate the hype from the real deal.



Jaideep Roy and Anupama Ramanujan

XML: Data's Universal Language

You've probably heard a lot lately about Extensible Markup Language—a new tag-based language for describing data. Trade journals and industry magazines devote cover pages to it, and application vendors build in support for it and promote its virtues. The most enthusiastic supporters claim XML will open the door for the next generation of Internet applications. But will XML live up to all the hype?

THE XML STORY

XML is a subset of the Standard Generalized Markup Language (SGML), a complex standard for describing document structure and content. XML is a language for *organizing*—not merely presenting—data (see "XML Sets Stage for Efficient Knowledge Management" in this issue). Because it is a *metalanguage* (a language for describing other languages), XML lets you define your own customized markup languages for different document classes.

A nonproprietary specification, XML is a project of the World Wide Web Consortium; W3C's XML Working Group supervises its development. The W3C issued version 1.0 as a *recommendation* (meaning it deems the language appropriate for widespread use) on 10 February 1998, but XML is still evolving with new features and functionalities.

Inside

XML, SGML, and HTML: A Closer Look

Read More about It

Supporting Features and Technologies

THE HEART OF XML

Thanks to its general nature, SGML is flexible but not simple. The Hypertext Markup Language, an application of SGML that contains only a fixed set of tags, is simple but hardly flexible. But XML is both flexible and simple, a combination that makes it powerful and useful (see the "XML, SGML, and HTML: A Closer Look" sidebar).

More flexible than a fixed-format markup language such as HTML, XML adds context and gives meaning to data. XML lets you define your own custom tags to represent data logically. Figure 1 shows a sample XML document: shipping address information for a customer named John Smith, for order number A9999. Using such a document, we can identify key relationships about different data items with respect to the entire "customer" entity. This document is *self-describing*, because tags describe the information it contains.

XML is simple because its rules for creating a markup language to encapsulate data are straightforward. For example, XML documents contain

Figure 1. Sample XML document.

```
<?xml version="1.0"?>
<customer order_number="A9999">
<first_name>John</first_name>
<middle_initial/>
<last_name>Smith</last_name>
<shipping_address>
   <street>123 Street</street>
   <city>New York</city>
   <state>NY</state>
   <zip>12345</zip>
   <country>USA</country>
</shipping_address>
</customer>
```

1520-9202/00/$10.00 © 2000 IEEE

simple tags, with data stored between them as plain text. The tags usually come in pairs and can nest to multiple levels. Similarly, because XML data is stored as ordinary text, you can use a standard text editor to create and edit XML documents.

Moreover, XML supports the Unicode standard (http://www.unicode.org), a character-encoding system that supports all major languages. For example, you could specify Unicode Transformation Format 16-bit encoding form as

```
<?xml version="1.0"
encoding="UTF-16"?>.
```

Unicode support lets XML accept virtually all the characters used in the world. Along with software to process XML properly, this feature can be a huge benefit in developing applications that span national and cultural boundaries.

XML documents essentially have a rooted tree structure. For many applications, this structure is powerful enough to represent complex data—and writing software programs that manipulate tree-structured data is easy.

THE XML GRAMMAR

A document type definition (DTD) defines an XML document's legal structure, or *grammar*, which specifies

- what markup tags are available,
- where they may occur, and
- how they all fit together.

Figure 2 shows a sample DTD for the XML document specified in Figure 1.

Figure 2. Sample document type definition for XML document shown in Figure 1.

```
<!ELEMENT customer (first_name,middle_initial?,
  last_name,shipping_address)>
<!ATTLIST customer order_number ID #REQUIRED>
<!ELEMENT first_name (#PCDATA)>
<!ELEMENT middle_initial (#PCDATA)>
<!ELEMENT last_name (#PCDATA)>
<!ELEMENT shipping_address
  (street,city,state,zip,country)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT country (#PCDATA)>
```

Elements are the most common form of markup. They are delimited by angle brackets and in most cases identify the nature of the content they surround (for example, <first_name>). Attributes are name-value pairs that occur after the element name, as in customer order_number ="A9999" in Figure 1.

By definition, all XML documents must be *well formed*,

XML, SGML, and HTML: A Closer Look

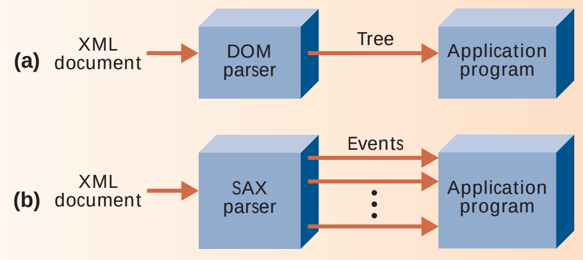
Standard Generalized Markup Language, an international standard (ISO 8879) published in 1986, describes a standard format for embedding descriptive markup in a document and a method for describing document structure. SGML lets you create documents that are independent of any specific hardware or software, and it supports an endless variety of document structures. So why was it never widely used? Because SGML is too general and complex, so writing processing programs with it is difficult.

Extensible Markup Language is a simplified version of SGML. Because XML leaves out the more complex and less-used features, it is easier to understand, easier to use for developing applications, and better suited for delivery and interoperability over the Web. But it is still SGML, and you can parse and validate XML files the same way as with any other SGML file.

Hypertext Markup Language, on the other hand, is an *application* of SGML. It uses a small subset of tags that conform to a single SGML specification. Using a small tag set greatly simplifies building applications and has contributed to HTML's widespread success as a Web publishing language. But this simplicity comes with a price: HTML's fixed format limits its usefulness in other areas.

HTML and XML, though both derived from SGML, differ. While HTML concerns the presentation of data, XML adds context and gives meaning to data. HTML is a rigid specification that uses inflexible tags, whereas XML lets you define your own custom tags so that you can represent data logically in a structured manner. In short, HTML focuses on presentation, and XML focuses on content.

Figure 3. Processing XML documents.



(a) A Document Object Model (DOM) parser converts an XML document to a tree structure (stored in memory) before giving it to an application program, and (b) a Simple API for XML (SAX) parser generates events that an application program receives.

meaning they must obey XML syntax. All the elements must match (all elements potentially containing data must have both the start and end tags). Empty tags can use a special XML syntax—for example, <middle_initial/> in Figure 1. Elements must nest properly, attribute values must always be quoted, and so on (see the XML specification at http://www.w3.org/XML). A well-formed document is easy for a computer program to read.

An XML document is *valid* if it is well formed, has a proper DTD, and obeys all the DTD's rules. Thus, all valid documents are well formed, but not vice versa.

The XML standard does not require that you use a DTD, but doing so enables validity checking of tags and standardized construction and naming of documents. Moreover, embedding relevant business rules in a DTD ensures that XML documents adhere to those rules. But careful DTD design is key to deploying effective and scalable XML applications. The DTD should cover all possible document cases and also allow future enhancements.

PROCESSING XML DOCUMENTS

An XML document doesn't do anything by itself; it must be combined with an application program that does something useful with it. The *parser* is the interface between an XML document and the application program that uses it. This XML processor is a software module that reads XML documents and gives application programs access to those documents' internal structure and content. The parser's main purpose is to preprocess XML documents and provide application programs with data that is easier to work with. Two common types of XML parser application programming interfaces (APIs) are

- *Document Object Model* (DOM), a tree-structure-based API issued as a recommendation by W3C in 1998 (http://www.w3.org/TR/PR-DOM-Level-1), and
- *Simple API for XML* (SAX), an event-driven API developed by XML-DEV mailing-list members.

DOM, a platform- and language-neutral interface, lets programs and scripts dynamically create, access, and update documents' content, structure, and style. DOM represents an XML document as a tree whose nodes are elements, text, and so on. All nodes originate from one root node, called the *document object*. As Figure 3 shows, an XML processor generates the tree and hands it to an application program. DOM provides APIs to access and manipulate these nodes in the DOM tree. A DOM-based XML processor creates the XML document's entire structure in memory. DOM is best suited for

- structurally modifying an XML document,
- dynamically creating new XML documents, and
- sharing with other applications the document in memory.

An XML processor with SAX does not create a data structure. Instead, it scans an input XML document and generates events, such as element start or end. The application programs implement handlers that receive these events and do appropriate processing. SAX is best suited for

- handling large documents that do not fit in memory, and
- extracting the contents of specific elements.

The good news about XML parsers is that you do not necessarily have to write them: Parsers are already available in different languages, and most of them are free. A popular example is the XML4J Java parser from IBM.

KEY APPLICATION AREAS

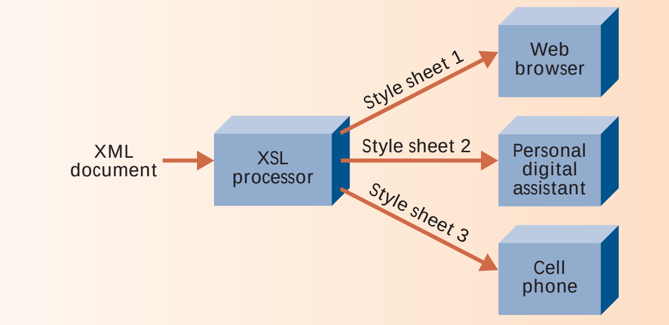
XML provides flexible document-definition and processing capabilities. It lets you reformat data for multiple devices and platforms. Because XML separates display instruction from content definition, Web designers can alter their Web site's look and feel by using Extensible Stylesheet Language (XSL) documents to apply different style sheets to the same XML document. Thus, you can use the same content for devices such as personal digital assistants (PDAs) and wireless devices that do not use HTML for display processing (see Figure 4).

XML excels in making online information search and retrieval fast and efficient. The reason is that XML documents also store *meta-information* (information about information). By looking at the tags, we can determine what each data item is—the customer's first name, last name, address, and so on. Search engines can use this feature to efficiently search and retrieve documents. For example, we could process search queries such as "find all documents where customer's last name is Smith"—a decisive advantage over HTML.

One of the hottest application areas of XML is messaging. XML enables seamless and efficient transfer of data between applications. Because it is text based, all platforms can easily understand it. XML can be the least common denominator for representing information. Thus, it is the perfect medium for exchanging information between organizations or within an organization across different platforms.

Because XML promises to improve the way companies exchange and present information over the Internet, it is becoming popular with developers of next-generation business-to-business (B2B) e-commerce applications. XML can benefit e-commerce by enabling back-end systems to communicate business transaction information. For example, business partners can standardize a specific XML syntax that describes a purchase order, and automate that information's transfer across the Internet. XML is ideal for building these systems because it allows formatting data for easy-to-process, platform-neutral exchange between business partners. With B2B e-commerce expected to reach $1.3 trillion by 2003 and $7.3 trillion by 2004, XML promises to be a key enabling technology.

Figure 4. Formatting an XML document for display using XSL.



LIMITATIONS AND DRAWBACKS

Despite the hype surrounding XML, it isn't a one-stop solution for all application development issues. A poor choice for building internal stand-alone systems, XML also falls short when security and efficient low-level communication are critical.

XML is limited in terms of the data types it supports. It is a text-based format and does not have facilities for directly supporting binary data or other complex data types such as multimedia data. (XML can, however, support binary data if that data is encoded as text. For example, the Java Developers Kit 2.0 supports a base-64 binary encoder that can be used to transform binary data into text.) XML also lacks data typing. Even though it is excellent in validating a document's structure using DTDs, it doesn't check for errors in data contained within a document. However, this may soon change with the adoption of the XML Schema standard, currently being developed by the W3C.

Another limitation is the lack of client-side XML support. Among the popular Web browsers, only Microsoft's Internet Explorer 5.0 and the recently released Netscape Communicator/Navigator 6.0 offer some support for processing and displaying XML documents.

A major hindrance in adopting XML is the lack of standard vocabularies or tag sets. There is no clear consensus yet on how to define key business terms such as "customer" or "invoice" within vertical or horizontal industry segments. For example, one company might define an XML tag for a purchase order using the customer's name and order number, whereas another company might just use the order number. Information can get lost or misinterpreted when data passes between these two companies. Exchanging data between companies in different industries can exacerbate this problem. Standard XML vocabularies, at least for specific industries, would ensure that systems could exchange data in a consistent manner.

A NOTE ABOUT SECURITY

Although XML will greatly simplify Internet-based B2B messaging, it does not by itself provide any security features. This limitation poses a problem because the Internet is a public network, and messages can be stolen or modi-

Read More about It

- **Jon Bosak and Tim Bray, "XML and the Second-Generation Web,"** ***Scientific American*****, May 1999; http://www.sciam.com/1999/0599issue/0599bosak.html.**
- **Robin Cover, ed.,** ***The XML Cover Pages*****; http://www.oasis-open.org/cover.**
- **"Extensible Markup Language (XML)," W3C, MIT/INRIA/Keio Univ., Cambridge, Mass./Le Chesnay Cedex, France/Fujisaway, Japan; http://www.w3.org/XML.**
- **Elliotte R. Harold,** ***XML Bible*****, IDG Books Worldwide, Foster City, Calif., 1999.**
- **Hiroshi Maruyama et al.,** ***XML and Java: Developing Web Applications*****, Addison Wesley Longman, Reading, Mass., 1999.**
- **David Megginson,** ***Structuring XML Documents*****, Prentice Hall, Upper Saddle River, N.J., 1998.**
- **"XML Developer Center,"** ***msdn online*****, Microsoft Corp., Redmond, Wash.; http://msdn.microsoft.com/xml/default.asp.**
- **"XML Zone,"** ***developerWorks*****, IBM Corp., Armonk, N.Y.; http://www.ibm.com/developer/xml.**

Supporting Features and Technologies

Designers and developers are continually adding supporting features and technologies to XML.

XHTML. Extensible Hypertext Markup Language is the result of rewriting HTML (version 4.0) as an XML application. XHTML creates a middle ground between HTML and XML. It will open up Web access to more devices and increase the capabilities of devices that already offer such access, such as cell phones, personal digital assistants, and other miniature devices (see http://www.w3.org/TR/xhtml1).

XSL. Extensible Stylesheet Language lets you apply rules for formatting, including presentation format (for example, font size), to XML documents. XSL can transform XML documents into different formats such as HTML, PDF, or even audio. Once XSL converts an XML document into HTML, you can view that document using any browser. XSL can also transform one XML document into another (see http://www.w3.org/Style/XSL).

XML schema. An XML schema defines the elements that can appear in an XML document along with attributes and their default values, if any. It also defines the document's structure: the parent and child elements, the number of child elements, the sequence in which the elements can appear, and whether an element can be empty or include text. In addition, it can enforce data typing. An XML schema provides a more powerful mechanism than DTDs for describing an XML document's structure (see http://www.oasis-open.org/cover/schemas.html).

XML namespace. An XML namespace is a collection of element types and attribute names identified by a universal resource indicator (URI). Any element type or attribute name in an XML namespace can be uniquely identified by a two-part name: the URI of its XML namespace and its local name. An XML namespace distinguishes between duplicate element types and attributes so that you can mix two or more XML languages in one document without any conflict or ambiguity (see http://www.w3.org/TR/REC-xml-names).

XPointer. The XML Pointer language supports making specific references to an XML document's internal structure. XPointer provides a mechanism to refer to elements, character strings, selections, and other parts of the document (see http://www.w3.org/TR/WD-xptr).

XLink. XML Linking Language specifies constructs that you can insert into XML documents to describe links between objects. XLink describes the simple unidirectional hyperlinks of today's HTML, as well as more sophisticated bidirectional, multidirectional, and typed links (see http://www.w3.org/TR/xlink).

fied during transmission. One possible solution is to use cryptographic protocols such as the Secure Sockets Layer to secure communications. Commercial products (such as X/Secure from Baltimore Technologies) that can be used to digitally sign, encrypt, verify, and decrypt XML documents are also emerging. W3C and the Internet Engineering Task Force are also developing a digital signature standard for XML documents.

Another security issue arises when XML documents refer to resources such as DTDs stored on inadequately secured external systems. An attack on these systems that slightly modifies these external resources can create havoc in XML processing. The easiest solution is to copy necessary resources to local secure systems. But this approach reduces flexibility, especially if the resources are being shared. Security is a critical issue that designers of Web-based XML applications must address.

A WORK IN PROGRESS

XML is fast becoming the key language for an increasing number of new applications. It is poised to fundamentally alter the way we deliver and use information and to enable the creation of new, powerful applications. It has the potential to shape the future of the Web, among other things. But XML is still a work in progress. Designers and developers are adding new features to it and developing new technologies around it (see the "Supporting Features and Technologies" sidebar). ■

Jaideep Roy *is vice president of information technology at Bear Stearns & Co., where he designs and develops Web-based financial applications. Contact him at jroy@bear.com.*

Anupama Ramanujan *is a computer consultant at AT&T Labs, where she builds B2B e-commerce applications. Contact her at ramanujan@att.com.*

The views expressed in this article are entirely those of the authors and do not in any way reflect the views of Bear Stearns & Co. or AT&T.