XML: A Door to Automated Web Applications

HTML allows the structural markup of Web documents. XML takes document markup to the next level, with machine-readable semantic organization of networked data.

ROHIT KHARE
MCI Internet Architecture
ADAM RIFKIN
California Institute of Technology

In Japanese culture, your *meishi* conveys your place in the company, even in society, as well as your name, phone number, and e-mail address. That is to say, in Japan, business cards *matter*. They convey complex metadata about the people who carry them.

Like people, Web pages come in an abundance of shapes and sizes (and sounds). What makes them machine interpretable—and therefore a new medium for communicating information globally—is Hypertext Markup Language. HTML allows the *structural* markup of Web documents, distinguishing the elements of a page with tags and declaring the physical relationships among the various document elements (for example, "this is a new paragraph" or "this is emphasized from its surrounding text"). This organizes the display of information and allows humans to read and use it.

To give machines this capability, however, requires *semantic* markup, identifying what each particular element means on its own (for example, "this is a home street address" or "this is an e-mail address"). Semantic markup would change what is now simply displayed content to machine-readable, structured content.

The eXtensible Markup Language (XML) specification, first released as two working drafts in Spring 1997 by the World Wide Web Consortium (see the sidebar, XML Timeline), makes it dramatically easier to develop and deploy domain- and mission-specific Web pages. In this article, we describe the evolution of the Web's data representation from display formats to structural markup to semantic markup.

THE DEVELOPMENT OF MARKUP LANGUAGES

The idea that structured documents could be exchanged and manipulated if published in a standard, open format dates back to work done in the 1960s. In one endeavor, a committee of the Graphic Communications Association (GCA) created GenCode to develop generic formatting codes for clients who wanted to send the same data

1089-7801/97/$10.00 ©1997 IEEE

XML TIMELINE

The World Wide Web Consortium (W3C) is a vendor-neutral international industry consortium founded in 1994 to develop common protocol specifications for the evolving WWW. When an important new issue relevant to the Web arises W3C team members organize a workshop to collect information, opinions, and ideas. This leads to the formation of a working group with specific short-term goals, watched over by an editorial review board of experts elected by W3C members.

W3C results may enter the IETF standards track. For example, W3C and the IETF are currently in partnership on the development of HTTP 1.1 and Distributed Authoring and Versioning.

The W3C SGML working group was chartered for a single year from July 1996–1997 with Jon Bosak of Sun Microsystems as chair, and Dan Connolly as the W3C staff contact person. Table A presents some of the milestones in the XML effort.

Table A. Milestones in the XML specification development.

Month	Activity
July 1996	W3C work on SGML officially began. Initial XML draft charter can be viewed at (http://www.w3.org/pub/WWW/MarkUp/SGML/Group/"#charter).
September 1996	Report on the Generic SGML activity at the Seybold Conference, San Francisco, Calif.
November 1996	Bert Bos' presentation at the SGML 96 Conference, Boston, Mass., (http://www.w3.org/TR/WD-xml-961114.html).
January 1997	Peter Flynn began the FAQ "Commonly Asked Questions about XML" (http://www.ucc.ie/xml/).
February 1997	Imperial College, London, formed (and continues to host) the "xml-dev" mailing list for XML developers (http://www.lists.ic.ac.uk/hypermail/xml-dev/).
March 1997	First XML Conference in San Diego, Calif., by the Graphic Communications Association. Revised XML Syntax Working Draft (http://www.w3.org/TR/WD-xml-lang-970331.html).
April 1997	Initial XML Linking Working Draft (http://www.w3.org/TR/WD-xml-link-970406.html).
	XML was formally presented by the W3C at the Sixth International World Wide Web Conference, Santa Clara, Calif. (http://www.w3.org/pub/WWW/Conferences/WWW6/).
May 1997	Approval of a technical corrigendum to ISO 8879:1986 to align features of XML with the SGML standard at the May 1997 meeting of ISO/IEC JTC1/SC18/WG8, Barcelona (http://www.sgmlsource.com/8879rev/n1929.htm).
June 1997	New versions of the XML Syntax and Linking Working Drafts available.
December 1997	SGML/XML 97 Conference, Washington, D.C., will see the release of near-final specifications and the initial edition of a standard style sheet language for XML publishing applications based on DSSSL (ISO/IEC 10179), along with the public text and extensions Web browsers will need to implement it.

from different typesetters to different vendors. GenCode allowed them to maintain an integrated set of archives despite the records being set in multiple types.

In another effort, IBM developed the Generalized Markup Language for its internal publications, in part to assist with the management of documents of all types, from manuals and press releases to legal contracts and project specifications. GML was designed to be reusable for batch processors so that books, reports, and electronic editions could be produced from the same source file(s).

GML had a "simple" input syntax for typists, including the tags we recognize today denoted by <> and </>. Of course, GML also permitted lots of "cheating": Typists avoided markup with tags wherever they could. As a result, markup minimization erred on the side of making these documents easier for humans to type and read than for general-purpose processing by computer applications. In fact, so few document types were interchanged at that time that people wrote special compilers bound to each particular kind of document to handle the inputting of the appropriate data formats.

As more document types emerged (each requiring specially suited tagsets), so did the need for a standard way to publish and manipulate what we later learned to call the Document Type Definition. (DTD is the definition of a document type in SGML, consisting of a set of markup tags and their interpretation. HTML, for example, is defined by its own DTD.) Representatives of both the GenCode and GML communities joined in the early 1980s to form the American National Standards Institute committee on Computer Languages for the Processing of Text. The committee's goal was to standardize the ways of specifying, defining, and using markup in documents.

The Standard Generalized Markup Language was published as ISO standard 8879:1986.[1] It was designed to be formal enough to allow proofs of document validity, structured enough to handle complex documents, and extensible enough to support management of large information repositories. The two key elements of SGML were its syntax (which evolved from IBM's GML) and its semantics (which came from the typesetters through the GCA).

Although suffering from some of the problems attendant to "design by committee," SGML was nonetheless successful in furnishing an interchange language that could be used to manipulate and exchange text documents. By the late 1980s, organizations such as CERN (the European Laboratory for Particle Physics in Switzerland) had adopted it. It was in a CERN laboratory that Tim Berners-Lee borrowed the funny-looking idiom for his then-new hypertext application. Indeed, the inventor of the World Wide Web picked a very small set of SGML's structural markup concepts for his Hypertext Markup Language in 1990 to complement the style sheets he had designed so that his Web browser could understand typesetting hints.

By the time Mosaic took off worldwide in 1993, people were using HTML as a hammer and seeing nails everywhere. However, HTML only furnishes an elegant but nonetheless small tagset (even as of HTML 4.0,[2] released in July 1997), and no single tagset will suffice for all of the kinds of information on the Web. Two of the people involved in early HTML specification, Dan Connolly and Dave Raggett, believed that to help solve this problem HTML should be formalized to rigorously use generic SGML, and began work toward this goal in the mid-1990s. Although to date most of the HTML files on the Web are not absolutely conformant to SGML, this initiative had the significant effects of involving the SGML community with the Web, and shifting the agendas of some companies to unite SGML with Web technologies.[3]

It is fair to say, however, that the ISO 8879:1986 SGML standard was a large, cumbersome specification designed as though the PC revolution had never happened. In fact, the specification ignored the most basic results in machine parsing from the computer science compiler studies in the 1960s. Not only was the long specification difficult for people to read and understand, but it was also hard for computers to process and manipulate. By 1996, SGML was still not ready for interactive parsing, authoring, or (thanks to its batch heritage!) incremental display.

However, SGML offers some attractive functionalities not available in HTML. The most important of these are:

- *Extensibility*: Authors can define new tag names and attribute names for documents by specifying their syntax and semantics.
- *Structure*. Documents can be containers for other documents, with arbitrary nesting. This allows complex documents to be constructed from simpler documents.
- *Validation.* If desired, any SGML document can reference a description of its grammar so applications can validate that the document conforms to its specified structure. Furthermore, this process of validation can be automated.

WHY XML?

In 1996, a new team backed by the World Wide Web Consortium worked to realize these benefits for the Web in a format more usable—both by humans and computers—than SGML.

XML[4] is a simplified (but strict) subset of SGML that maintains SGML's features of extensibility, structure, and validation. XML is a standardized text format designed specifically for transmitting structured data to Web applications. In addition, because it is the goal of XML to be easier to learn, use, and implement than full SGML, it will be easier to define and validate document types, to author and manage SGML-defined documents, and to transmit and share such documents across the Web.

The new subset, like SGML, is a metalanguage for describing the markup of different types of documents. However, its specification is 26 pages—versus 500 for SGML. It is therefore a practical solution to the Gordian knot of being precise and extensible without sacrificing simplicity.

XML is not a replacement for SGML. In fact, many features of SGML were left out to keep XML simple. XML therefore does not support several standard (but complex) features of SGML: It does not allow markup minimization, for example, and it requires that empty elements be self-identifying.

However, the real strength of XML for the Web is that it was designed with network delivery concerns in mind. Current SGML users could therefore choose XML for data exchange over a network. A well-formed XML document is unambiguous, so a general-purpose SGML browser or editor could read the tags and create a tree of the hierarchical structure without having to read its DTD. Since XML is a valid subset of SGML, the translation from SGML to XML is straightforward.

Another powerful feature of XML is the way in which it changes the linking model by allowing authors to specify different types of document relationships. New linking technology allows the management of bidirectional and multiway links, as well as links to a span of text (within the same or other documents) as a supplement to the single-point linking afforded by HTML's existing HREF-style anchors.

With XML's syntax for both linking and defining tagsets simplified over SGML, any parser can handle any document. As a result, anyone can issue a DTD specialized to particular data needs. Furthermore, as with SGML, XML DTDs can be composed together to create more complex, validatable new document types from simpler, validated DTDs. In addition, authors can stylize how the document's appearance is formatted in a browser or other Web-compliant application using Cascading Style Sheets (CSS) or the Document Style, Semantic, and Specification Language (DSSSL).

The working draft for XML 1.0* provides a complete specification in two parts: the extensible markup language itself,[5] and methods for associating hypertext linking and style sheet mechanisms with XML.[6] Although XML is not backward-compatible with existing HTML documents, since it modifies the syntax and semantics of document tag annotations, HTML 4.0-compliant documents can easily be converted to XML.

A BRIEF INTRODUCTION

Recall that SGML is an internationally standardized language for defining sets of tags. HTML represents just one of the element type tagsets that can be created using SGML. The complete specification of HTML 3.2 as an SGML DTD means that documents can be verified to be HTML 3.2-compliant.

As with SGML documents, XML documents are composed of *entities*, which are storage units containing text and/or binary data.[7] Text is composed of character streams that form both the document's character data content and the document's metadata markup. Markup describes the document's storage layout and logical structure. XML also provides a markup mechanism to impose constraints on the storage layout and logical structure of documents,[8] and it provides mechanisms that can be used for strong typing.[9]

In style and structure, XML documents look quite similar to HTML documents. However, when Web servers with XML content prepare data for transmission, they typically must generate a context wrapper with each XML fragment, including pointers to an associated DTD and one or more style sheets for formatting. Web clients that process XML must be able to unpack the content fragment, parse it in context according to the DTD (if needed), render it (if needed) in accordance with the specified style sheet guidelines, and correctly interpret the hypertext semantics (such as links) associated with each of the different document tags.

Note that a DTD is not required for an XML document. Instead, an author can simply use an application-specific tagset. However, a DTD is useful because it allows applications to validate the tagset for proper usage. A DTD specifies the set of required and optional elements (and their attributes) for documents to conform to that type. In addition, the DTD specifies the names of the tags and the relationships among elements in a document (for instance, nesting of elements).

Two examples of "electronic business card" (bCard) documents illustrate the power and simplicity of XML. In the first example, the DTD is given as part of the XML document. In the second example, the DTD is not given, but it exists in an externally defined document.

Example 1: Annotated Attribute-Value Pairs

Let's write a simple XML document that only contains tags annotated with attribute-value pairs; that is, there will be no content in the document other than the tags themselves. These tags can then be parsed and processed by software programs.

Our simple example is a document for maintaining a list of people's electronic business cards. Suppose, then, we want each "bCard list tag" to contain five attributes: a person's first name, surname, company, e-mail address, and Web page address.

We can specify default values to attributes to guarantee that every tag has the same number of attribute-value pairs (although some values may be null). The declaration of default attributes is lexically scoped by the bCard element (although in this case it has no effect, since none of the elements omit an attribute).

```
<!DOCTYPE bCard
"http://www.cs.caltech.edu/~adam/schemes/bCard">
```

```
<?xml default bCard
      firstname = " "
      lastname  = " "
      company   = " "
      email     = " "
      webpage   = " "
?>
<bCard
      firstname = "Rohit"
      lastname  = "Khare"
      company   = "MCI Internet Architecture"
      email     = "khare@mci.net"
      webpage   = "http://pest.w3.org/"
/>

<bCard
      firstname = "Adam"
      lastname  = "Rifkin"
      company   = "Caltech Infospheres Project"
      email     = "adam@cs.caltech.edu"
      webpage   = "http://www.cs.caltech.edu/~adam/"
/>
</bCard>
```

Note how XML's formatting is readable by both humans and machine: Empty lines immediately following a ">" or immediately preceding a "<" in the document are ignored by the parser, and white space inside tags is ignored (which is not true for HTML).

Example 2: Embeddable Tags

As a text-based format, XML is designed for storing and transmitting data. This can be done either through arbitrary attribute-value pairs, as demonstrated in the first example, or by strategically embedding tags around content to give that content more meaning.

For example, consider the following XML snippet:

```
<!doctype html>
<html version="-//W3C//DTD HTML Experimental
970324//EN">
<head>
<title> Adam's bCard List </title>
</head>
<body>

<h1> Adam's bCard List </h1>

<bCard MONTH=7 YEAR=1997>
<FIRSTNAME> Adam </FIRSTNAME>
<LASTNAME> Rifkin </LASTNAME>
<COMPANY> Caltech Infospheres Project </COMPANY>
<EMAIL> adam@cs.caltech.edu </EMAIL>
<WEBPAGE> http://www.cs.caltech.edu/~adam/
</WEBPAGE>
</bCard>

<bCard MONTH=8 YEAR=1997>
<FIRSTNAME> Rohit </FIRSTNAME>
<LASTNAME> Khare </LASTNAME>
<COMPANY> MCI Internet Architecture </COMPANY>
<EMAIL> khare@mci.net </EMAIL>
</bCard>

<hr/>
<address><a href="mailto:adam@cs.caltech.edu">
Adam Rifkin</a> </address>
<!— Created: Wed Jul 16 12:22:32 MET DST 1997 —>
<!— hhmts start —>
Last modified: Wed Jul 16 22:32:42 MET DST
<!— hhmts end —>
</body>
</html>
<!— Keep this comment at the end of the file
Local variables:
mode: sgml
sgml-declaration:"~/SGML/html.decl"
sgml-default-doctype-name:"html"
sgml-minimize-attributes:t
sgml-nofill-elements:("pre" "style" "br")
sgml-live-element-indicator:t
End:
—>
```

Note that the DTD is not embedded in the document. We could specify it elsewhere if we need to validate the tagset and content data structures, or we could omit the DTD.

By binding a meaning to the XML tag <bCard>, we understand what is contained in that element: the start tag, the end tag, and the contents in between those tags. In this case, the bCard element has two attributes, MONTH and YEAR, the values of which correspond to the month and year that bCard entry was added to the document. However, the DTD might specify that the bCard element must contain the FIRSTNAME, LASTNAME, COMPANY, and EMAIL elements, and might contain a WEBPAGE element as well. Additionally, the DTD might specify that any WEBPAGE element that does appear in a valid electronic business card document must be nested within a bCard element.

Once a document type's elements have been specified in a DTD, style sheets, scripts, and programs can be associated with any element in that document type. For example, a custom script might execute when someone clicks on that business card entry, opening up a separate window that displays the entry in a particular font, color, and arrangement. Or, a style sheet associated with business cards might display all

entries of MCI employees with the MCI logo stamped across the middle of the card.

The forms of metadata provided in example 1 (attribute-value pairs) and example 2 (start-end tags) demonstrate the different ways that document content can be marked up with metadata to allow that content to be searched, retrieved, or filtered. Metadata spans a wealth of information, from digital signatures and authentication seals, to prices and timestamps, to links to related information.

To summarize, XML allows authors to specify their own document syntax, hypertext link semantics, and presentation style. Once we can create new tags and elements with new attribute-value metadata, we can reencode any systematic, structured document format using XML.

Integrating XML Content with HTML

HTML, with its millions of users and billions of documents, will not be lost in the transition to XML. Even in an XML-centric world, most documents will use the idioms of paragraphs, headings, and lists—and Web page authors will use <P>, <H1>, and <LI> tags just as they do today. Newfangled XML markup will emerge around new, semantically significant data structures. We can expect to integrate XML business cards in the middle of an HTML home page with minor disruption—or, similarly, digitally sign or encrypt individual portions of a document. Client software and network tools will evolve gracefully alongside these changes. Because today's tools can cope reasonably well with unrecognized tags and today's HTTP 1.1 can compress slightly more verbose XML, there won't be a need for wholesale changes.

XML APPLICATIONS

Some of the first "generic" applications of XML,[10] which should begin appearing this year, will be able to

- use Web clients (browsers and other Web-compliant user programs) to mediate between multiple heterogeneous databases;
- distribute some of the processing load from Web servers to Web clients;
- use Web clients to present different views of the same data to different users; and
- employ agents to tailor information discovery and filtering to the customized needs of individual users.

The following examples give an idea of the broad range of applications of XML now in development.

XML for Scientific Applications

Many communities have struggled to codify tacit knowledge of how their data is structured and manipulated into common file formats. Many have even adopted SGML, realizing the value of textual, standardized interchange formats for long-term stability. However, user organizations often borrowed the SGML committee design mentality as well, laboring over a single, "big bang" DTD release for their domain. After going down this path, the chemistry community recently adopted XML as a more flexible base for its evolving Chemical Markup Language (CML).

CML[11] will use XML to manage molecular information. Although CML was originally developed against SGML, there are compelling reasons to base it on XML as well. CML documents can hold extremely complex information structures, acting either as an interchange or archival mechanism, and interface easily with modern relational and object-oriented database architectures.

CML takes advantage of the fact that XML documents need not be valid and can simply be *well formed*. Essentially, this means that although a document is syntactically correct (for example, the start and end tags balance, ATTRIBUTEs are quoted, and so on), the document itself might not be valid (for example, it might contain an unknown tag). Therefore XML is better suited than SGML to situations in which documents have already been validated (for example, because the authoring software is authenticated, or because the documents have already passed through a validating parser) and you may simply want to manipulate them. Thus, although all CML documents must be validatable against the CML DTD, it is possible to manipulate them even without the DTD or, indeed, any knowledge of chemistry at all.

When the XML specification is complete, XML will allow new markup tagsets to evolve out of rapid experimentation by a community that needs the DTDs. For example, XML fragments have evolved over the past year (concurrently with the XML effort) with the explicit goal of supporting mathematics in documents. Note that because XML DTDs are composable, defining a document type for mathematical formulas will allow any author to include equations by composing its DTD with the mathematics DTD.

Mathematical Markup Language (MathML)[12] is an XML application for describing the structure and content of mathematical expressions. It allows the markup of complex formulas, something that has been needed by mathematicians and computer scientists since the earliest days of HTML.

Sophisticated mathematical notation is highly symbolic, and the relation between meaning and notation is often subtle. This has ramifications for the *say what you mean* aspect of semantic markup. To keep in line with the philosophy behind mathematical expressions, MathML describes expression structure together with its mathematical context. About two dozen MathML tags describe abstract notational structures, and another four dozen provide a way of unambiguously specifying the intended meaning of an expression. MathML content and presentation tags can interact to capture the nuances of meaning in traditional

equations. The MathML working draft also discusses how renderers might be implemented and how they should interact with browsers.

XML for Automated Distribution

Recent efforts led by Microsoft and Netscape demonstrate the efficacy of XML for pushing and pulling information over the Web.

Microsoft's proposed Channel Definition Format (CDF)[13] lets a Web site use XML to publish existing HTML content in a channel for desktop CDF-compliant "push" client browsers (from vendors such as Microsoft, PointCast, AirMedia, and BackWeb). XML also provides a way to embed arbitrary data and annotations within the broadcast HTML for use with scripts. CDF permits a Web publisher to offer frequent updates of information from any Web server for automatic delivery to compatible receiver programs on PCs or other information appliances.

As an XML application, the CDF specification allows Web publishers to push information by allowing them to specify the channels, the content available, the update schedule, and other information such as a delay period between when the data is received and when it is browsable (to synchronize readers in multiple distributed locations, for instance). CDF overcomes a serious problem with the push platforms of today (by vendors such as PointCast, Backweb, Microsoft, and Marimba) in that the publisher and the subscriber must use the same technology. If content providers push their information as CDF documents and data streams, then any user with a CDF-compliant client browser can read that information. This open standard puts flexibility in the hands of the user, who can now pick a custom client application for reading, and makes wider audiences available to the content providers as well.

Netscape is using XML for a different style of application: "pulling" metadata and other information. Called Meta Content Framework (MCF),[14] it provides the specification for a data model to describe the information organization structures (meta content) for collections of networked information, using XML syntax to represent instances of this data model.

Handheld Device Markup Language

Handheld Device Markup Language (HDML)[15] addresses the constraints of pocket-size devices: a few lines of display, a limited keypad, tens of kilobytes of memory, and a wireless connection to the Internet. HDML, like HTML, is an information publishing and interaction description language, but it extended HTML with new tags, the semantics of which were unfortunately clear only to Unwired Planet.* Although HDML was designed before XML was available, Unwired Planet is presently looking into revising HDML to be based on XML—a solution that would let them use device-specific cascading style sheets and preloaded binary compression dictionaries to separately settle the "pocket-size" constraints issue across platforms.

WEB AUTOMATION

The applications of XML we have explored so far allow authors to design custom tagsets. As an author, you can define a DTD to precisely *say what you mean* about the information content of a document by using the tags for interpreting and supplementing that content. However, XML is useful for another, entirely different reason: Because it honors machine-readability (one of the basic tenets of the Web), it opens up new application areas for automation.

Automated Interpretation

Put simply, XML automates the extraction of data. For example, "electronic business cards" embedded in Web home pages could automatically offer information in the same commonly understood format to a variety of programs, Web forms, and scripts. As another example, a flight checker could extract the airline flight status reports from several different Web content provider services, and collate them into a single page formatted according to the readers' requirements.

To perform such automation tasks, Web programmers could use operational hacks such as scripts—which leave room for plenty of errors and manageability problems. The alternative that XML suggests, as exemplified by the airline flight example, would allow the evolution of an airline community ontology for flight data.

There are alternative approaches to imbuing a document with a structured ontology. At one extreme, the meaning of a document could be represented by its behavior alone; that is, its meaning is reflected only by what happens when it's processed. W3C's Document Object Model follows this route by systematically binding programs to parts of an HTML document, animating it like a puppet on a string (this approach is also marketed as "dynamic HTML").

A more robust approach may be to declare, deterministically, what parts of a document mean and what behaviors those parts have. The Web Interface Definition Language (WIDL) was developed by webMethods* specifically to describe the inputs and outputs of programs on the Web.[16] WIDL captures the meaning of a document by extracting relevant output fields and mapping inputs onto Web forms.

WIDL is a metadata syntax implemented in XML that defines APIs to Web data and services, enabling automatic and structured Web access by compatible client programs, including mainstream business applications, desktop applications, applets, Web agents, and server-side Web programs.

WIDL provides well-defined "machine-readable hooks" into Web data and services on the Internet. Most important, WIDL can describe interfaces for Web sites that are not con-

trolled by calling programs. WIDL files can reside on the client or server, or they can be centrally managed by third-party naming services.

This simple example demonstrates how a package tracking service might be described in WIDL:

```
<WIDL NAME="PackageTracking">

<SERVICE NAME=TrackPackage
 INPUT=InputData OUTPUT=OutputData METHOD=POST
 URL="http://www.packages_r_us.com/cgi-
bin/AirbillTrace" />

<BINDING NAME=InputData>
<VAR NAME=trackNum />
</BINDING>

<BINDING NAME=OutputData>
<CONDITION TYPE="success" MATCH="*Airbill
Number:*"
  REF=doc.title />
<CONDITION TYPE="failure" MATCH="*Blank Airbill*"
  REF=doc.p[0].value REASONTEXT="Please provide an
Airbill Number" />
<CONDITION TYPE="failure" MATCH="*should be*"
  REF=doc.p[0].value REASONREF=doc.p[0].value />
<CONDITION TYPE="failure"
  MATCH="*No information available*"
  REF=doc.p[0].value REASONREF=doc.p[0].value />
<CONDITION TYPE="failure"
  MATCH="*is not a valid*"
  REF=doc.p[0].value REASONREF=doc.p[0].value />
<VAR NAME=package
  REF=doc.tables[1].tr[0].td[0].value />
<VAR NAME=deliveredOn
REF=doc.tables[2].tr[3].td[1].value />
<VAR NAME=signedForBy
REF=doc.tables[3].tr[2].td[1].value />
</BINDING>

</WIDL>
```

Bindings between HTML and XML document elements and program variables can be defined using DOM references. Condition statements provide fault tolerance and can initiate alternate binding attempts and other WIDL-defined service invocations. These features provide enhanced fault tolerance and the capability to return meaningful error messages to calling programs.

WIDL provides abstract definitions for services that can be implemented in any language. WebMethods WIDL-based tools generate application-level function calls in C/C++, Java, Javascript, Visual Basic, and ActiveX directly from WIDL files.

Because WIDL is dynamically interpreted at runtime, client applications are insulated from changes in service locations and document structure. Transparency is achieved by changing document object references and service URLs without regenerating client code.

Taken together, these facilities make WIDL arguably analagous to the Interface Definition Languages (IDLs) of the Common Object Request Broker Architecture (CORBA) and the Distributed Computing Environment (DCE). WIDL can define the name, inputs, outputs, data types, and exceptions for any "function" on the Web.

Consider a simple customer-service function already on the Web: tracking the status of a package with an overnight courier service. One webMethods customer wrote an abstract interface to a long list of delivery companies' tracing forms, then collated it into a single "metatracer" that could simultaneously check on the status of several packages at several companies every time the report page is reloaded. Another developer put together a similar demo, which compares current RAM prices from several online vendors.

In summary, there are two aspects of automatable data mining using XML. One is as a metadata file describing interfaces to other files. In this case, converting to XML is a format nicety for handling arbitrary new interface description files. The other aspect is as metadata for adding XML markup to the actual output information, for direct manipulation by an author or program.

Automated Publication

XML can also automate the generation of data from databases and other data stores in a wide variety of publication formats. XML makes it easy to translate a new data schema into a document format for storing it. For example, with a bCard type in hand, we can store entire rolodexes in XML files, extract new records from old HTML home pages, SQL databases from personnel, or inscrutable X.509 certificates. Furthermore, we can compose bCard data with other kinds of structures. A bCard embedded within a link or a page with active content could answer the questions "Who owns this link?" or "Who wrote this applet?" Even beyond the realm of human-readable data, XML could be an ideal way to hold the state of any attribute-value data in a distributed system, because it is trivial to substitute object references with URLs.

Automated Processing

Bosak[10] has demonstrated how XML can enable advanced Web applications, allowing Java applets to embed powerful, automatable data manipulation facilities directly into Web clients. XML may be adopted for the automated conversion of data because it respects the adhoc social agreements behind community ontologies in a way that the creation by

committee of universal data dictionaries cannot.

These next few years will give rise to many such interpreters and translators akin to the combinatorial explosion of graphics-format converters.[17] There is a virtuous cycle to creating a new data type, publishing its syntax freely, and giving away some of the tools that manipulate it. One need only contemplate the positive-feedback growth of RealAudio streams and RealAudio players—and to consider how much further the community could have gone without proprietary audio encodings.

THE EVOLUTION OF THE WEB

The World Wide Web Consortium, the driving force behind XML, sees its mission as leading the evolution of the Web. Given the competitive market of Internet technologies, it is important to consider just how quickly the Web has absorbed competing protocols. Though it shared several adaptations common to Internet protocols ("free software spreads faster," "ASCII systems spread faster than binary ones," and "bad protocols imitate; great protocols steal") it leveraged one unique strategy: self-description. The Web, as we are now beginning to realize, can be built upon itself. Universal resource locators, machine-readable data formats, and machine-readable specifications can be knit together into an extensible system that assimilates any competitors.

The Web stole content-neutrality from MIME: It learned how to adapt to any document type equally. On the other hand, some types were more equal than others: The Web prefers HTML over Portable Document Format (PDF), Microsoft Word, and myriad other formats. That's because of a general trend over the past seven years of Web history from formatting to structural to semantic markup. Each step up in the ascent of formats—from PostScript (opaque, operational, formatting); to troff (readable, operational formatting); to Rich Text Format (RTF) (readable, extensible, formatting); to classic HTML (readable, declarative structure); to HTML 1.x (readable, limited declarative semantics like <ADDRESS>); to XML; and on to intelligent metadata like Platform for Internet Content Selection (PICS) labels and Knowledge Interchange Format (KIF)—adds momentum to Web applications.

As such, the Web is becoming a kind of cyborg intelligence: man and machine, harnessed together to generate and manipulate information. If automatability is to be a human right, then the drudge work involved in exchanging and manipulating knowledge must be eliminated by machine assistance (see the discussion by MIT Laboratory for Computer Science Director Michael Dertouzos in his book *What Will Be*[18]).

In short, the shift from structural HTML markup to semantic XML markup is a critical phase in the struggle to transform the Web from a universal information space into a knowledge network. ■

REFERENCES

1. C.F. Goldfarb, *The SGML Handbook*, Y. Rubinsky, ed. Oxford Univ. Press, 1990. (This volume contains the full annotated text of ISO 8879 (with amendments)).
2. D. Raggett, A. Le Hors, and I. Jacobs, "HTML 4.0 Specification," World Wide Web Consortium Working Draft (Work in Progress), July 1997, http://www.w3.org/TR/WD-html40/.
3. Y. Rubinsky and M. Maloney, *SGML and the Web: Small Steps Beyond HTML*, C.F. Goldfarb series on Open Information Management, Prentice-Hall, Upper Saddle River, N.J., 1997. (An ideal introduction to SGML for HTML users.)
4. D. Connolly and J. Bosak, "Extensible Markup Language (XML)," 1997, http://www.w3.org/XML/.
5. T. Bray and C.M. Sperberg-McQueen, "Extensible Markup Language (XML): Part I. Syntax," World Wide Web Consortium Working Draft (Work in Progress), Mar. 1997, http://www.w3.org/TR/WD-xml-lang.html.
6. T. Bray and S. DeRose, "Extensible Markup Language (XML): Part II. Linking," World Wide Web Consortium Working Draft (Work in Progress), Apr. 1997, http://www.w3.org/TR/WD-xml-link.html.
7. B. Box, "The XML Data Model," 1997, http://www.w3.org/XML/Datamodel.html.
8. B. Bos, "XML Representation of a Relational Database," 1997, http://www.w3.org/XML/RDB.html.
9. T. Bray, "Adding Strong Data Typing to SGML and XML," May 1997, http://www.textuality.com/xml/typing.html.
10. J. Bosak, "XML, Java, and the Future of the Web," http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm, 1997.
11. P. Murray-Rust, "Chemical Markup Language (CML)," Version 1.0, Jan. 1997, http://www.venus.co.uk/omf/cml/.
12. P. Ion and R. Miner, "Mathematical Markup Language," W3C Working Draft, May 1997. http://www.w3.org/pub/WWW/TR/WD-math.
13. C. Ellerman, "Channel Definition Format," W3C submission, Mar.1997, http://www.w3.org/TR/NOTE-CDFsubmit.html.
14. R.V. Guha and T. Bray, "Meta Content Framework Using XML," June 1997, http://www.w3.org/TR/NOTE-MCF-XML/.
15. *Unwired Planet*, "Proposal for a Handheld Device Markup Language," Working Draft, Version 2.0, May 1997, http://www.uplanet.com/pub/hdml_w3c/hdml_proposal.html.
16. webMethods, "Web Interface Description Language Specification," 1997, http://www.webmethods.com/technology/widl.html.
17. J.D.Murray and W. vanRyper, *Encyclopedia of Graphics File Formats*, O'Reilly & Associates, Sebastapol, Calif., 1996.
18. M. Dertouzos, *What Will Be,* HarperEdge, New York, 1997.

Rohit Khare (http://xent.w3.org/) is a member of the MCI Internet Architecture staff in Boston. He was previously on the technical staff of the World Wide Web Consortium at MIT, where he focused on security and electronic commerce issues. He has been involved in the development of cryptographic software tools and Web-related standards development. Khare received a BS in engineering and applied science and in economics from the California

Institute of Technology in 1995. He expects to join the PhD program in computer science at the University of California, Irvine, in Fall 1997.

Adam Rifkin (http://www.cs.caltech.edu/~adam/) is pursuing a PhD in computer science at the California Institute of Technology, where he works with the Caltech Infospheres Project (http://www.infospheres.caltech.edu/) on the composition of distributed active objects. He received BS and MS degrees in computer science from the College of William and Mary. He has done Internet consulting and performed research with several organizations, including Canon, Hewlett-Packard, Griffiss Air Force Base, and the NASA-Langley Research Center.

Readers may contact Khare at khare@alumni.caltech.edu and Rifkin at adam@cs.caltech.edu.

URLs FOR THIS ARTICLE

***XML 1.0 working draft** www.textuality.com/sgml-erb/WD-xml.html
***Unwired Planet** www.uplanet.com/
***webMethods** www.webmethods.com/home.html

Other Useful Links

An Initial Investigation of XML
www.cs.caltech.edu/~adam/local/xml.html
World Wide Web Consortium HTML Page
www.w3.org/MarkUp/Wilbur
SGML Page at the Summer Institute of Linguistics (SIL)
www.sil.org/sgml
World Wide Web Consortium SGML Activity Page
www.w3.org/MarkUp/SGML/Activity