XML, Java, and the future of the Web

Table of Contents

Jon Bosak, Sun Microsystems
Last revised 1997.03.10

1. Introduction

The extraordinary growth of the World Wide Web has been fueled by the ability it gives authors to easily and cheaply distribute electronic documents to an international audience. As Web documents have become larger and more complex, however, Web content providers have begun to experience the limitations of a medium that does not provide the extensibility, structure, and data checking needed for large-scale commercial publishing. The ability of Java applets to embed powerful data manipulation capabilities in Web clients makes even clearer the limitations of current methods for the transmittal of document data.

To address the requirements of commercial Web publishing and enable the further expansion of Web technology into new domains of distributed document processing, the World Wide Web Consortium has developed an Extensible Markup Language (XML) for applications that require functionality beyond the current Hypertext Markup Language (HTML). ***This paper*** [0] describes the XML effort and discusses new kinds of Java-based Web applications made possible by XML.

2. Background: HTML and SGML

Most documents on the Web are stored and transmitted in HTML. HTML is a simple language well suited for hypertext, multimedia, and the display of small and reasonably simple documents. HTML is based on SGML (Standard Generalized Markup Language, ISO 8879), a standard system for defining and using document formats.

SGML allows documents to describe their own grammar -- that is, to specify the tag set used in the document and the structural relationships that those tags represent. HTML applications are applications that hardwire a small set of tags in conformance with a single SGML specification. Freezing a small set of tags allows users to leave the language specification out of the document and makes it much easier to build applications, but this ease comes at the cost of severely limiting HTML in several important respects, chief among which are extensibility, structure, and validation.

- *Extensibility.* HTML does not allow users to specify their own tags or attributes in order to parameterize or otherwise semantically qualify their data.
- *Structure.* HTML does not support the specification of deep structures needed to represent database schemas or object-oriented hierarchies.
- *Validation.* HTML does not support the kind of language specification that allows consuming applications to check data for structural validity on importation.

In contrast to HTML stands generic SGML. A generic SGML application is one that supports SGML language specifications of arbitrary complexity and makes possible the qualities of extensibility, structure, and validation missing from HTML. SGML makes it possible to define your own formats for your own documents, to handle large and complex documents, and to manage large information repositories. However, full SGML contains many optional features that are not needed for Web applications and has proven to have a cost/benefit ratio unattractive to current vendors of Web browsers.

3. The XML effort

The World Wide Web Consortium (W3C) has created an SGML Working Group to build a set of specifications to make it easy and straightforward to use the beneficial features of SGML on the Web. See the ***W3C SGML Activity page*** [1] for the current status of this effort. The goal of the W3C SGML activity is to enable the delivery of self-describing data structures of arbitrary depth and complexity to applications that require such structures.

The first phase of this effort is the specification of a simplified subset of SGML specially designed for Web applications. This subset, called XML (Extensible Markup Language), retains the key SGML advantages of extensibility, structure, and validation in a language that is designed to be vastly easier to learn, use, and implement than full SGML.

XML differs from HTML in three major respects:

1. Information providers can define new tag and attribute names at will.
2. Document structures can be nested to any level of complexity.
3. Any XML document can contain an optional description of its grammar for use by applications that need to perform structural validation.

XML has been designed for maximum expressive power, maximum teachability, and maximum ease of implementation. The language is not backward-compatible with existing HTML documents, but documents conforming to the W3C HTML 3.2 specification can easily be converted to XML, as can generic SGML documents and documents generated from databases.

An ***initial working draft for XML 1.0*** [2] has been released for public discussion. A complete specification that includes methods for associating hypertext linking and stylesheet mechanisms with XML documents is scheduled for release at the Sixth World Wide Web Conference in April, 1997.

4. Web applications of XML

The applications that will drive the acceptance of XML are those that cannot be accomplished within the limitations of HTML. These applications can be divided into four broad categories:

1. Applications that require the Web client to mediate between two or more heterogeneous databases.
2. Applications that attempt to distribute a significant proportion of the processing load from the Web server to the Web client.
3. Applications that require the Web client to present different views of the same data to different users.
4. Applications in which intelligent Web agents attempt to tailor information discovery to the needs of individual users.

The alternative to XML for these applications is proprietary code embedded as "script elements" in HTML documents and delivered in conjunction with proprietary browser plug-ins or Java applets. XML derives from a philosophy that data belongs to its creators and that content providers are best served by a data format that does not bind them to particular script languages, authoring tools, and delivery engines but provides a standardized, vendor-independent, level playing field upon which different authoring and delivery tools may freely compete.

4.1 Database interchange: the universal hub

A paradigmatic example of this first category of XML applications is the information tracking system for a home health care agency.

Home health care is a major component of America's multibillion-dollar medical industry that continues to increase in importance as the health care burden is shifted from hospitals to home care settings. Information management is critical to this industry in order to meet the record-keeping requirements of the federal agencies and health maintenance organizations that pay for patient care.

The typical patient entering a home health care agency is represented to the information system by a large collection of paper-based historical materials in the form of patient medical histories and billing data from a variety of doctors, hospitals, pharmacies, and insurance companies. The biggest task in getting the patient into the system is the manual entry of this material into the agency's database.

The coming of the Web has given the medical informatics community the hope that an electronic means can be found to alleviate this burden. Unfortunately, existing Web applications represent fundamentally insufficient models for an adequate solution. Hospitals have begun to offer the agencies a solution that goes something like this:

1. Log into the hospital's Web site.
2. Become an authorized user.
3. Access the patient's medical records using a Web browser.
4. Print out the records from the browser.
5. Manually key in the data from the printouts.

The knowledgeable reader may smile at this "solution," but in fact this is not a joke; this is an actual proposal from a large American hospital known for its early adoption of advanced medical information systems.

A slightly more sophisticated version of this "solution" envisions the operator reading the patient data from the Web browser and keying it directly into the agency's online forms-based interface in a separate window instead of making a printout first. The only difference between this version and the previous one is that it saves the paper that would have been needed for the printout. It does nothing to address the root of the problem. A real solution would look more like this:

1. Log into the hospital's Web site.
2. Become an authorized user.
3. Access the patient's medical records in a Web-based interface that represents the records for that patient with a folder icon.

4. Drag the folder from the Web application over to the internal database application.
5. Drop it into the database.

However, this solution is not possible within the limitations of HTML, for three reasons.

- The HTML tag set is too limited to represent or differentiate between the multitude of database fields in the mixture of documents making up the patient's medical history.
- HTML is incapable of representing the variety of structures in those documents.
- HTML lacks any mechanism for checking the data for structural validity before the receiving application attempts to import it into the target database.

One technically feasible way to implement seamless interchange of patient care records is simply to require all hospitals and health care agencies to use a single standard system dictated by the government (such an approach has actually been suggested). In an environment where hospitals are going out of business on a daily basis and many health care agencies are in deep financial difficulty, however, a scheme that would require them to replace their existing heterogeneous systems with a single new system *en masse* is hardly practical.

The other way to enable interchange between heterogeneous systems is to adopt a single industry-wide interchange format that serves as the single output format for all exporting systems and the single input format for all importing systems. This is, in fact, the purpose for which SGML was initially designed, and XML simply carries on this tradition.

A number of industries, including the aerospace, automotive, telecommunications, and computer software industries, have been using hub languages to perform data interchange for years, and by this time the process is well understood. Typically, the major players in an industry form a standards consortium tasked with defining a Document Type Definition, which is the way in which the tag set and grammar of a markup language are defined. This DTD can then be sent with documents that have been marked up in the industry standard language using off-the-shelf editing tools, and any standard application on the receiving end can validate and process them.

The XML solution is system-independent, vendor-independent, and proven by over a decade of SGML implementation experience. XML merely extends this proven approach to document interchange over the Web. Interestingly, the same day on which the first XML 1.0 draft was released also saw the formal announcement of an SGML initiative within HL7, the standards organization for health care IS vendors, to develop a Health Care Markup Language designed to solve exactly the kind of problem described in this example.

Previous vertical-industry efforts have shown that capturing data in a rich markup often has benefits beyond the immediate requirements of data exchange. In a well-designed standardized patient data system, for example, specific information originally gathered in the course of a routine physical exam and tagged <allergies>, <drug-reactions>, and so on would instantly be available to alert the staff of an emergency room that an unconscious patient from a distant city was allergic to penicillin. The ability of XML to define tags specific to an area of application is critical to this scenario, because the otherwise unqualified word "penicillin" in the thousands of pages of a patient's entire medical history could not trigger the recognition that the same word inside an <allergies> element could trigger.

The health care example is relevant not only because of the scope of the problem and the enormous sums of money involved but also because it is paradigmatic of a very wide range of future Web applications -- any in which Web clients (or Java applications running on those clients) are expected to mediate the lossless exchange of complex data between systems that use different forms of data representation in a way that can be standardized across an industry or other interest group. Some random examples of such applications are:

- Legal publishing
- The government drug approval process
- Collaborative CAD/CAM efforts
- Collaborative calendar management across different systems
- Any corporate network application that works across databases, especially where policies must be enforced: purchase orders, expense requests, etc.
- Exchange of information between players in any broker-organized business: insurance, securities, banking, etc.

4.2 Distributed processing: giving Java something to do

A paradigmatic example of this second category of XML applications is the data delivery system designed by the semiconductor industry.

Each major semiconductor manufacturer maintains several terabytes of technical data on all of the ICs that it produces. To enable interchange of this data, an industry consortium (the Pinnacles Group) was formed several years ago by Intel, National Semiconductor, Philips, Texas Instruments, and Hitachi to design an industry-specific SGML markup language. The consortium finished that specification in 1995, and its member companies are now well into the implementation phase of the process.

One might think that the rise in popularity of HTML would cause the Pinnacles members to reconsider their decision, but in fact the limitations of HTML have convinced them that their original strategy was the correct one. Their initial idea was that the richly parameterized data stream made possible by the industry-specific SGML markup would enable intelligent applications not merely to display semiconductor data sheets as readable documents but actually to drive design processes. It is now recognized that this approach is a perfect fit with the concept of distributed Java applets, and the vision of the near future is one in which engineers can access a manufacturer's Web site and download not only viewable data on particular integrated circuits but also a Java applet that allows them to model those circuits in various combinations.

The semiconductor application is a good demonstration of the advantages of XML because:

1. It requires industry-specific markup that cannot be implemented within the confines of the fixed HTML tag set.
2. It requires that the data representation be platform- and vendor-independent so that data from a variety of sources can be used to drive a variety of distributed applications (some of which may be provided by third parties, generating a subindustry of providers of tools that can work with the standardized data stream).
3. Its utility rests ultimately in the fact that a computation-intensive process (modeling circuits for hours at a time) that would otherwise entail an enormous, extended resource hit on the

server has been changed into a brief interaction with the server followed by an extended interaction with the user's own Web client. This aspect has been summed up in the slogan "XML gives Java something to do."

Note that validation, while sometimes important, does not always play the crucial role in this category of applications that it does in applications where data must be checked for structural integrity before entering a database. To make processing as efficient as possible, XML has been designed so that validation is optional in applications where it is not needed.

As with the health-care example, the semiconductor application is notable not merely for the sheer size of the market it represents but also because it is paradigmatic of an enormous range of future Java-based Web applications -- virtually any application in which standardized data is expected to be manipulated in interesting ways on the client. Perhaps the most obvious examples of such applications are the following:

- Design applications where the designer would otherwise use server cycles to consider various alternatives: electronics, engineering, architecture, menu planning, etc.
- Scheduling applications where a customer would otherwise use server cycles to entertain various possibilities: airlines, trains, buses, and subways; restaurants, movies, plays, and concerts. This is what Easy Sabre and Ticketron will look like a few years from now as the economies of distributed Java-based processing become evident.
- Commercial applications that allow consumers to explore alternatives by supplying different shopping criteria: real estate, automobiles, appliances, etc.
- The entire spectrum of educational applications, a small subset of which are the ones we call "online help".
- The entire spectrum of customer-support applications, ranging from lawn-mower maintenance through technical support for computers.

A harbinger of applications to come in the last category is the Solution Exchange Standard, an SGML markup language announced last June by a consortium of over 60 hardware, software, and communications companies to facilitate the exchange of technical support information among vendors, system integrators, and corporate help desks. In the words of the announcement:

> The standard has been designed to be flexible. It is independent of any platform, vendor or application, so it can be used to exchange solution information without regard to the system it is coming from or going to. [...] Additionally, the standard has been designed to have a long lifetime. SGML offers room for growth and extensibility, so the standard can easily accommodate rapidly changing support environments.

Such applications, which the XML subset is specifically designed to address, will grow in importance as consumers come to expect interoperability among their data-manipulating applets and information providers confront the realities of trying to support computation-intensive tasks directly on their Web servers.

4.3 View selection: letting the user decide

A third variety of XML applications are those in which users may wish to switch between different views of the data without requiring that the data be downloaded again in a different form from the Web server.

One early application in this category will be dynamic tables of contents. It is possible now, using Web servers built on object-oriented databases, to present the user with a table of contents into a large collection of data that can be expanded with a mouse click to "open up" a portion of the TOC and reveal more detailed levels of the document structure. Dynamic TOCs of this kind can be generated at run time directly from the hierarchical structure of the document. Unfortunately, the Web latency built into every expansion or contraction of the TOC makes this process sluggish in many user environments. A much better solution is to download the entire structured TOC to the client rather than just individual server-generated views of the TOC. Then the user can expand, contract, and move about in the TOC supported by a much faster process running directly on the client.

A group at Sun actually implemented a form of this solution as part of a Java-based HTML help browser, but the limitations of HTML required the team to come up with a couple of clever workarounds. In this application, a TOC was constructed by hand (the lack of structure in ordinary HTML makes it impossible to reliably generate a TOC directly from the document) using nonstandard tags invented for the purpose, and then the TOC piece was wrapped in a comment within an HTML page to hide the nonstandard markup from Web browsers. A Java applet downloaded with the HTML document interpreted the hidden markup and provided the client-based TOC behavior.

In practice, this application worked very well and testified both to the ingenuity of its designers and to the validity of the basic concept. But in an XML environment, neither the manual creation of the TOC nor its concealment would have been necessary. Instead, standard XML editors would have been used to create structured content from which a structured TOC could be generated at run time and downloaded to browsers that would automatically create and display the TOC using either a downloaded Java applet or a standard set of JavaHelp class libraries.

The ability to capture and transmit semantic and structural data made possible by XML greatly expands the range of possibilities for client-side manipulation of the way data appears to the user. For example:

- A technical manual that covers both the Sparc and x86 versions of the Solaris operating system can be made to appear like a manual for Sparc only, or a manual for x86 only, just by clicking a preferences switch.
- An installation sheet that carries warnings in multiple languages can be made to show just the ones in the language selected by the user.
- A document containing many annotations can be switched from a mode that shows only the text, to a mode that shows only the annotations, to a mode that shows both, just by making a menu selection.
- A phone book sorted by last name can instantly be changed into a phone book sorted by first name.

This list only hints at the possible uses that creative Web designers will find for richly structured data delivered in a standardized way to Web clients.

4.4 Web agents: data that knows about me

A future domain for XML applications will arise when intelligent Web agents begin to make larger demands for structured data than can easily be conveyed by HTML. Perhaps the earliest applications in this category will be those in which user preferences must be represented in a standard way to mass media providers. The key requirements for such applications have been summed up by Matthew Fuchs of Disney Imagineering: "Information needs to know about itself, and information needs to know about me."

Consider a personalized TV guide for the fabled 500-channel cable TV system. A personalized TV guide that works across the entire spectrum of possible providers requires not only that the user's preferences and other characteristics (educational level, interest, profession, age, visual acuity) be specified in a standard, vendor-independent manner -- obviously a job for an industry-standard markup system -- but also that the programs themselves be described in a way that allows agents to intelligently select the ones most likely to be of interest to the user. This second requirement can be met only by a standardized system that uses many specialized tags to convey specific attributes of a particular program offering (subject category, audience category, leading actors, length, date made, critical rating, specialized content, language, etc.). Exactly the same requirements would apply to customized newspapers and many other applications in which information selection is tailored to the indvidual user.

While such applications still lie over the horizon, it is obvious that they will play an increasingly important role in our lives and that their implementation will require XML-like data in order to function interoperably and thereby allow intelligent Web agents to compete effectively in an open market.

5. Advanced linking and stylesheet mechanisms

Outside XML as such, but an integral part of the W3C SGML effort, are powerful linking and stylesheet mechanisms that go beyond current HTML-based methods just as XML goes beyond HTML.

5.1 Linking

Despite its name and all of the publicity that has surrounded HTML, this so-called "hypertext markup language" actually implements just a tiny amount of the functionality that has historically been associated with the concept of hypertext systems. Only the simplest form of linking is supported -- unidirectional links to hardcoded locations. This is a far cry from the systems that were built and proven during the 1970s and 1980s.

In a true hypertext system of the kind envisioned for the XML effort, there will be standardized syntax for all of the classic hypertext linking mechanisms:

- Location-independent naming
- Bidirectional links

- Links that can be specified and managed outside of documents to which they apply
- N-ary hyperlinks (e.g., rings, multiple windows)
- Aggregate links (multiple sources)
- Transclusion (the link target document appears to be part of the link source document)
- Attributes on links (link types)

The first draft of a specification for basic standardized hypertext mechanisms to be used in conjunction with XML is scheduled for release at the Sixth World Wide Web Conference in April, 1997.

5.2 Stylesheets

The current CSS (cascading style sheets) effort provides a style mechanism well suited to the relatively low-level demands of HTML but incapable of supporting the greatly expanded range of rendering techniques made possible by extensible structured markup. The counterpart to XML is a stylesheet programming language that is:

- Freely extensible so that stylesheet designers can define an unlimited number of treatments for an unlimited variety of tags.
- Turing-complete so that stylesheet designers can arbitrarily extend the available procedures.
- Based on a standard syntax to minimize the learning curve.
- Able to address the entire tree structure of an XML document in structural terms, so that context relationships between elements in a document can be expressed to any level of complexity.
- Completely internationalized so that left-to-right, right-to-left, and top-to-bottom scripts can all be dealt with, even if mixed in a single document.
- Provided with a sophisticated rendering model that allows the specification of professional page layout features such as multiple column sets, rotated text areas, and float zones.
- Defined in a way that allows partial rendering in order to enable efficient delivery of documents over the Web.

Such a language already exists in a new international standard called the Document Style Semantics and Specification Language (DSSSL, ISO/IEC 10179). Published in April, 1996, DSSSL is the stylesheet language of the future for XML documents. An initial specification of a ***DSSSL subset*** [3] for use with XML applications has already been published. This specification will be further developed as part of the XML activity.

6. Conclusion

HTML functions well as a markup for the publication of simple documents and as a transportation envelope for downloadable scripts. However, the need to support the much greater information requirements of standardized Java applications will necessitate the development of a standard, extensible, structured language and similarly expanded linking and stylesheet mechanisms. The W3C SGML effort is actively developing a set of specifications that will allow these objectives to be met within an open standards environment.

7. Acknowledgements

The author would like to thank his colleagues in the Davenport Group for early contributions to the beginnings of this document. The example applications were clarified and expanded with the help of participants in the workshop "Internet Applications of SGML and DSSSL" held at the GCA Information and Technology Week in Seattle on August 23, 1996. Special thanks are due to Tim Bray, Kurt Conrad, Steve DeRose, Matt Fuchs, and Murray Maloney for their outstanding contributions to the workshop.

8. Production note

This paper was written in HTML 3.2 and formatted by the ***Jade DSSSL engine*** [4] for printout. The section numbers, headers, footers, and Table of Contents seen in the printed version are not part of the ***HTML source*** [5] but were generated automatically as specified by a ***DSSSL stylesheet*** [6].

9. References

[0] http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.ps.zip
[1] http://www.w3.org/pub/WWW/MarkUp/SGML/Activity
[2] http://www.w3.org/pub/WWW/TR/WD-xml-961114.html
[3] http://sunsite.unc.edu/pub/sun-info/standards/dsssl/dssslo/dssslo.htm
[4] http://www.jclark.com/jade/
[5] http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm
[6] http://sunsite.unc.edu/pub/sun-info/standards/dsssl/stylesheets/html32/html32hc.dsl