eXtensible Markup Language:
Enabling Intelligent Applications

Developing creative, intelligent custom solutions for our clients has never been easier, yet paradoxically, more challenging. New technologies such as Java, ORBs, and application servers make it easy to design powerful, distributed applications. But developing and installing these systems present some significant challenges that must be overcome. Getting the data from its source to an end user in a reliable, expedient, and functional manner is the subject of many long and tedious design meetings. Granted, technology has made the possibilities seem endless. However, barriers to high-technology implementations, stemming from immature technologies and the rapid pace of development, may be too risky at times.

One simple new technology may be able to mitigate much of the risk and offer increasing returns to those who implement complex distributed systems: eXtensible Markup Language (XML). While it may not be a panacea for existing legacy issues XML offers a design edge for the future. In conjunction with Java, distributed objects, and other evolutionary technologies a true computing revolution may be at hand.

1. XML History

XML is a sub-set of the Standard Generalized Markup Language (SGML). Becoming an international standard in 1986, SGML was a reaction to industry demand for a standardized format in document publishing and storage. SGML has experienced widespread acceptance for certain specialized applications. As a medium for distributed content, SGML suffered from its own strength: complexity.

Too complex for simple applications to handle, SGML spawned the Hypertext Markup Language (HTML). By 1992 HTML gained support in the burgeoning web community and in 1994 became an official standard recommendation from the World Wide Web Consortium (W3C). But programmers were already discovering the shortcomings of HTML.

SGML is the Rolls Royce of markup and gives us the flexibility (and associated maintenance!) that you would expect from a high-end markup language. HTML is more like the Volkswagen Beetle—easy to look at, fun to drive, but not very powerful.

Enter XML.

By 1996 a first draft of XML had been presented at an annual SGML conference. Designers were aiming for that elusive Taurus, Camry, or Accord; a language which was easy to use, cheap, reliable and great at moving objects around. SGML was slimmed down from a specification of over 400 pages to very trim 36 page language specification for XML. In August of 1997 a proposal was made to the W3C and by February of 1998 the official recommendation was made. Today, XML retains most of the OO characteristics of SGML and adds HTML stylesheets and simplicity—making for a compact and powerful markup.

2. XML Vision

XML should be considered very cautiously, and rightly so. In an era of innovation we are faced with only two options: build it now and manage the risks or build it later and be left in the dust. XML makes sense in a variety of applications for many reasons.

2.1 Overcoming Issues in Technology

XML was designed to bridge the gap between SGML and HTML; mitigating some of their weaknesses.[1] In addition, XML helps Java applications overcome some inherent weaknesses.

2.1.1 SGML

The Standard Generalized Markup Language was the first fully extensible markup with rules for validation. Benefits of SGML include: infinite tag set, reuse, and validation. The approach used in specifying SGML hindered its growth for three reasons.

[1] See 'XML for Managers' from ArborText

2.1.1.1 Complexity and Compatibility

The sheer size of the specification for SGML has inhibited the widespread creation of applications that can interpret it. Namely, no major browser supports SGML as a native document type. Thus, formatting and data transfer over the web medium using SGML is all but impossible.

XML is a very small subset of SGML and was derived with the motivation that programs written to parse XML should be easy (and fast) to write. To date, Microsoft Internet Explorer 4.0 can parse XML and Netscape has released XML support in the source code for Mozilla (version 5.0).

2.1.1.2 Stylesheets

SGML lacks the standard way of linking formatting styles—stylesheets. Designers interested in managing the aesthetics of large web sites (or multiple, linked documents) have few choices beyond HTML for programming with stylesheets. HTML has fully supported stylesheets since it's last language specification (4.0). The XML specification has built-in support for the eXtensible Stylesheet Language; a format which is similar to HTML's cascading stylesheet language (CSS).

2.1.2 HTML

If XML were considered to be the sibling of SGML, then HTML would be its child. HTML has a fixed tag set infinitely less complex than SGML. HTML is the markup language that fueled the growth of the web precisely because it was so simple. Programmers today have great difficulty producing complex applications that have an HTML GUI. It works well for formatting display, but HTML manages data very poorly.

2.1.2.1 Reuse

HTML is primarily used for displaying information in web browsers. Unfortunately for most businesses the web is one of many mediums for storing, distributing, or displaying information. Therefore, HTML makes sense to use in only a limited number of applications, which means that businesses can't reuse much of the work creating HTML content.

In contrast, XML has the potential to become a true champion of object reuse in document publishing. Microsoft has committed to including XML as a native data format in its next release of the Office suite. Other publishing software makers, including Arbor Text, DataChannel, Chrystal, Informix, and Sybase are rapidly moving to support XML as well.

2.1.2.2 Structure

HTML primarily supports formatting tags. Information about information (metadata) exists only at the document level, and only minimally supported there. The inability to describe the content (at the data level) creates a superficially rigid and unaware structure for HTML documents.

If an ASCII text file is one-dimensional and an HTML document is two-dimensional, then XML files are multidimensional.

XML structure becomes very OO-like by supporting metadata. With custom tags and elements publishers may now create an inheritance for the data in their documents. A well structured XML definition becomes, in effect, a relational database. It's no wonder that Gartner Group sees demand for dB publishing solutions plummeting relative to demand for XML solutions[2].

2.1.2.3 Interchange

HTML's value as an interchange format is minimal for the same reasons that its utility for reuse is impractical/impossible. Interchange applications that source HTML must be very robust and/or proprietary because HTML cannot tell the application anything about different data within it. Because of this, HTML poses no threat to more traditional Electronic Data Interchange (EDI) formats. XML derivatives, on the other hand, have great potential to manage EDI in ways that could be interpretable

[2] Notation of GG IDOM # here

by everything from word processors to web browsers. Self-describing objects within a simple document structure is arguably the most powerful way to exchange data between applications.

2.1.2.4 Automation

Automating complex processes is dependent upon data that is either very expressive (lots of metadata), very consistent, or both. HTML is neither expressive nor consistent and therefore is not well suited for automated processes.

Imagine a scenario with an application tester in the telecommunications industry. A test analyst might write a test script, hand it off to a programmer who scripts it in an automated test tool, the test is run, and the results available from the tool are either raw (one-dimensional) data or HTML reports. Next, imagine how XML might change that. A test analyst might write the scripts in Word (XML format), source the same file for execution from the test tool. The test tool then formats the results in XML, making them available from the web for use in a browser, Word, Excel, or a reporting tool. XML could greatly enhance automated processes.

2.1.2.5 Signal to Noise Ratio

Go to Yahoo or AltaVista today and perform a search on the word 'pitch'. Your results set will certainly vary from pitch forks, to voice pitch, to sales pitches…etc. No way exists to search key words in a context sensitive manner. The web has been likened to a huge library with its books scattered all over the floor. No Dewey Decimal system exists to categorize data on the web. XML has the potential to show patterns in chaos. In the future, search engines might search on both key words and context cues (XML tags)—greatly decreasing the signal to noise ratio.

2.1.2.6 Shifting Standards

Because HTML is not extensible it cannot grow without new standards being drafted and approved. In addition, major browser makers vie for dominance by extending HTML themselves, thereby decreasing its utility as a standards based markup language. XML does not solve this problem entirely, but it does mitigate it rather well.

Document Type Definitions (DTDs) are like metadata descriptions for a whole series of documents—they tell applications what types of data a document contains and their relationships. Thus, browsers (or any other apps) are not faced with frequent upgrades to support new tags: they will look at the DTDs for descriptions of your tags[3].

2.1.3 Java

Jon Bosak of Sun Microsystems has said that, "XML gives Java something to do." This comment refers to a conceptual point about Java: as a programming language it allows programmers to build applications that do what C++ does faster, better, more distributed, etc.—only it doesn't allow us to do anything fundamentally different. However, by combining the strengths of XML with Java we may be able to build applications that really revolutionize computing.

In fact, this vision is so compelling that many believe that, "XML will be the glue that will integrate EDI, dBs, and even OSs, making the computer itself an extensible linked document and database."[4]

2.1.3.1 Proprietary Formats

Currently, to build Java applications that source rich data, a database for example, we use JDBC or middleware to import the data for processing. Once in the Java application the data itself essentially becomes locked in proprietary format. XML would allow Java to source data in an open and distributed format.

2.1.3.2 Single Sourcing

Java applications will be better able to source data from multiple XML files stored locally or remotely than with a few dBs. By linking beans, Java objects (via ORB technology) and XML

[3] In XML, unlike SGML, DTDs are not required—this will be discussed later in the paper.
[4] John Gage, Sun Microsystems

documents, programmers will be able to source data from a range of object repositories.

2.1.3.3 Processing Power

XML's object characteristics combined with Java's OO strengths make for a dramatic increase in the potential utility of our applications. In short: Java + XML = Processing Power.

Solutions that XML creates for distributed business applications include[5]:

- Extensible object definitions which don't devalue over time
- Distributed data objects available without predetermining and pre-defining relationships
- Identifying actionable objects in one-dimensional files
- Distributing actionable objects across all forms of electronic interchange

2.2 Horizontal Industry Solutions

Horizontal industry solutions are those which cut across vertical industries—solutions that have value across particular niches. Certain technologies and applications have value independent of industry: they are key enablers for functions required by many businesses. XML itself enables those technologies and applications to add more value by simplifying their creation and increasing their power.

2.2.1 Electronic Commerce

Companies are flocking to set up E-Commerce strategies to capitalize on a new era of electronic communications. Software vendors and consultancies are building applications to enable these new forms of business-to-business and consumer-to-business transactions. XML will be pivotal in decreasing time to market and increasing functionality in these applications.

Encapsulating information about transactions, products, and services will allow the next generation of E-Commerce applications to provide that elusive 'intangible value' that so often distinguishes a winning commerce application from an also-ran. XML will allow on-line malls to communicate with several back-end systems in a native format. In turn, the end user is more directly connected to background processes—sharing real-time status is key in successful E-Commerce ventures.

2.2.2 Electronic Data Interchange

EDI technologies have here-to-fore been proprietary formats which carried little, if any, metadata about their content streams. XML enables remote systems to communicate in a more standard way. Remote systems may someday not even have to predetermine or pre-define their inputs. Instead they might reference a DTD and sort out what's valuable and what's not. The process would be entirely loss-less since the XML tags and elements would define the data transmitted via EDI.

2.2.3 Knowledge Management

Knowledge management has become such a misused term it's almost inappropriate to talk about XML in its context. However, if ever there was a technology that could claim to enable knowledge management it would be XML. The very existence of XML presupposes an interest in knowledge management—XML is a language which categorizes data in printed form while utilizing object oriented methodology.

XML will potentially redefine the scope of knowledge management solutions by bringing metadata to the masses. Once a critical mass is reached in self-contained or open systems XML will boost efforts for: data categorization, knowledge categorization, and knowledge discovery.

2.2.3.1 Data Categorization

Utilizing XML, chunks of data in discrete documents can be related to each other, thereby providing a useful means to categorize data.

2.2.3.2 Knowledge Categorization

[5] JP Morganthal article

If data categorization is the means, then knowledge categorization is the end. At a high-level XML enables a superficial structure to be applied to an otherwise chaotic mess of data. Once the structure is in place for chunks of knowledge (composed of data) more complex processes can be applied with great success.

2.2.3.3 Knowledge discovery

XML might enable data mining techniques on traditional print media (far more numerous than traditional dBs). Machine learning algorithms may someday mine millions of web pages looking for patterns.

Imagine a repository containing gigabytes of published documents (they could be pharmaceutical reports, Aeronautical CAD drawings, silicon research, whatever…). Each document might be viewable independently, searchable in an intelligent XML way, or downloadable for input to a distributed Java app/local processor. In addition to all that, intelligent system agents might comb the repository applying complex algorithms to look for useful patterns. The power of XML is that it is self-contained and self-describing.

2.3 Vertical Industry Solutions

Individual industries often have needs that apply only to them. XML will allow the creation of industry-specific data types for information exchange between corporations.

2.3.1 Library Sciences

Library science is a text-book example of how XML may provide value. Every year print publications grow exponentially on the order of millions. Managing, tracking, and categorizing that growth is a phenomenal task. If an XML language was specified, which mirrored existing categories used in the library sciences, annual workloads might be drastically simplified or reduced.

2.3.2 Aerospace

The aerospace industry, as secretive as it is, also shares a great deal of information across companies. Currently much of this data transfer is handled using traditional EDI or hardcopies. XML could be used to embed and display data in CAD drawings. Documents could then be shared, manipulated, and processed in ways that are impossible today.

The list of potential examples could go on and on. The point here is that the sharing of intra-industry knowledge may spur a transformation of industry as XML gains popularity and semantic standards are reached.

2.4 Vertical Business Solutions

In the same way that XML can be leveraged for vertical industry solutions, XML can also be used for specific solutions within an organization. XML formats may be specified by individual business units to accomplish goals that may be extensible in the future. For example: if a testing business unit (BU) needed to store results sets in a format for use later they might define a set of tags and elements that are specific to that testing BU. Once implemented that testing organization would be able to fully leverage the OO characteristics of the XML documents and build applications that exploit it.

2.5 The Object Model

Perhaps the most conceptual point to the XML vision is the extension of the object model to data. Modern object oriented programming languages have their roots in languages that are over twenty years old, including Lisp, Smalltalk, and Simula. Object oriented techniques belong to more than just programming and therefore it's natural to apply them to data.

2.5.1 Data definition—Metadata

With XML it is possible to create documents that use metadata to define objects and classes[6]. This allows XML parsers to treat the document and its contents as objects with attributes and type-specific behavior. An application (Java, C++, etc.) would directly extract data from an XML document as type-specific to application

[6] Class definitions are extensions to XML, the specification does not standardize this approach.

objects. This approach treats XML objects as an extension of the application. Very powerful.

2.5.2 Object Systems

The unison of OO programming and object-aware data comes close to creating a truly ubiquitous object environment. The vision of self-aware data everywhere is mind-boggling. Data published by anyone becomes fair game for incorporation in to an object system—where applications may source themselves from potentially anywhere.

This vision seems far out. It is. A truly ubiquitous object environment is quite a way off, but conceivable. The Gartner Group estimates that in three years only 1/4th of businesses using XML will utilize its most important strengths[7]. Object systems sound great, but the reality of their planning and implementation is very complex.

2.5.3 Sophisticated Hyperlinking

One piece of the puzzle that must fall into place for any of these visions to work is a more sophisticated method of linking documents and data. The XML Linking Language (XLink) is currently in draft at the W3C; it will extend current hyperlinking methods greatly.

The history of hyperlinking actually dates back to the 1945. The concepts of associative indexing put forth since then are far more complex than what we use today with HTML. The Xlink draft is an attempt to bring sophisticated hyperlinking to the web medium. Some of the theoretical varieties of linking that will be explored in the Xlink draft are:

- Location Independent Naming
- Bi-directional Links
- Remotely Managed Links
- N-ary Hyperlinks (rings and multiple windows)
- Aggregate Links
- Transclusion (the link target appears in the link source document)

[7] Gartner Group IDOM T-PUB-422

Two working drafts are under consideration at the W3C—Xlink and Xpointer—which attempt to implement portions of these linking concepts, they will be discussed in the following section.

3. XML Technology

When considering the vision of XML it's important to bear in mind that XML itself is not a language with specific tags and elements. XML is, more accurately, a family of languages which allows a programmer to define a syntax for intelligently describing data as objects.

3.1 The XML Family

The XML family is a loosely combined group of extensions that comply with the XML specification. If you were to define an XML tagset with a DTD you would actually be adding to the XML family of languages. The following is a summary of some key XML family members.

3.1.1 RDF

Resource Description Framework – RDF is a "foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. RDF emphasizes facilities to enable automated processing of Web resources"[8]. XML is complimentary to RDF and is utilized to define a structure, required terms, their order and legal values for declaring metadata.

The working draft for RDF was submitted in October of 1997. Because of issues with semantics—RDF must be scaleable to any industry domain—many analysts expect RDF to spend some time in the approval process.

3.1.2 XSL

eXtensible Stylesheet Language – In note status since late August of 1997, XSL is a specification, in XML, for extending robust stylesheets to XML. XSL is a hybrid between DSSSL and CSS combining the ease of use

[8] RDF Working Draft. http://www.w3.org/TR/WD-rdf-syntax/

offered by simple CSS and the more complex object reordering which is allowed in DSSSL.

3.1.3 Xlink

Xlink and Xpointer are both working drafts that specify ways to link documents and objects together. Xlink is a document to document linking system which may be used with:

- Open systems: Out-of-line-linking (use links maintained somewhere else)
- Closed systems: In-line-linking (use links only you maintain)
- Uni-directional (similar to current HTML applications of hyperlinking)
- Bi-directional and Multi-directional (used in modern hypermedia systems today)

Xlink will be expressed in XML syntax.

3.1.4 XPointer

XPointer is the syntax, XML based, which will be used to 'point' to objects within XML documents: similar to how name anchors function in HTML. XPointers will be robust enough to address into XML documents with many hierarchies of data.

3.1.5 DOM

Document Object Model – A working draft for DOM was opened in October of 1997. DOM will function as a syntax for defining hierarchies within XML documents and how they may be accessed via an application programming interface (API). Applications will be empowered by opening the XML document directly for programmatic calls.

3.1.6 Others

As mentioned previously, defining an XML based syntax creates a unique application of XML. Here are some that exist today:

- OFX – Open Financial Exchange, a standards based format for financial transactions.
- CDF – Channel Definition Format, Microsoft's proposal for web channel format.
- OSD – Open Software Description, a standard format for online software distribution and support.

3.2 XML Semantics

Changes in semantics are not that great for those familiar with SGML. However, if your only markup experience is with HTML you may want to familiarize yourself with the following concepts before diving in to XML.

3.2.1 DTD

XML documents may be validated at parse-time, that is to say, the parser may verify that the document has the correct structure. In order for the parser to know if the document has good structure it must reference a document type definition (DTD).

A DTD is a declaration, that can reside anywhere, which contains a set of rules about the association of objects within a document. A single DTD may contain rules for thousands of documents—each of which is validated at the time of display or processing.

This capability is a huge leap forward from HTML. Powerful applications may be written to access data in XML documents by verifying proper structure and extracting objects at will.

3.2.2 Well Formed

In SGML it is a rule that DTDs must be referenced. This is fine for business applications, but what weekend web site designer will want to write a DTD to validate his or her homespun XML?

For that reason XML may exist as a standalone document. But, as a standalone document, it must still be well-formed.

A well formed document will have:

- balanced tags
- at least one tag pair for all data
- nested elements
- declarations for all entities

At parse-time a document may generate errors if it is not well formed.

3.2.3 Namespace[9]

In a given XML application, what is the ultimate meaning of the content of a given element? For example, in one application, someone might use an element called address to markup the mailing address of a person; in another application, address might instead be used for a network address. How would a machine or even a person looking at the XML markup know what "address" meant in a given instance?

What you need is a method for identifying the conventions governing the use of particular sets of elements. The idea is to use a URL as a globally unique name for the conventions. W3C's work on namespaces is concerned with the implementation of this idea.

3.3 XML and Java

While XML is not specifically designed to be utilized by any particular technology, Java will be able to leverage quite a bit from implementations of XML.

Distributed applications in Java will be able to source distributed XML documents as objects. More importantly, Java will be able to source portions of documents as objects; creating a quasi-database type environment. Data will no longer have to be locked away in proprietary database formats to be effectively used in applications. In addition, these same documents will be accessible in standard, commonly held document publishing platforms—further extending the value of data in XML format.

The chairman of the XML working group at the W3C, Jon Bosak, is a Sun employee. Sun Microsystems as a whole has committed wholeheartedly to the XML effort precisely because of its potential value to the Java language.

[9] The following information is taken directly from the W3C's Activity Statement on XML (http://www.w3.org/XML/Activity).

4. Counterpoint

Certainly XML has a lot of potential. XML is also very, very early in its development as a language. We've seen it before, with the hype surrounding Java for years, where pontificated benefits were sporadic at best.

Thoughtful application architects should not jump on the bandwagon too quickly—a real business case must exist to plan XML into a product.

4.1 Risks

The risks to business and the XML language itself are significant. Fortunately they are also well-known and manageable.

4.1.1 Hyperbole

Early in the release of a new technology much work is spent on visionary hype. Businesses and consumers begin to believe the hype, and anything short of absolute success becomes a total failure.

Early in the Java timeline visionaries saw it as the Microsoft killer. Today we have a language which is great for application web servers and distributed objects. Java may still be the Microsoft killer, but not in the near future.

XML will likely develop into a language which is good at some things and impractical for others. Some have expressed concern that too much hype and early failures will combine to kill momentum that otherwise might transform the computing industry. Maybe so, but project managers will have to balance the hype with reality to mitigate high-profile XML failures.

4.1.2 Immaturity

Of all of the XML family of languages only the parent—XML—is actually a standard recommendation from the W3C. Rules for managing metadata, links, pointers, and objects are all still in the approval process. Even when approved their utility at handling certain tasks will be unproven. Design teams will, to some extent, be taking a leap of faith when designing XML centered solutions.

4.1.3 Shifting Standards

The development of the HTML specification(s) should teach us a few things about proprietary extensions to languages. At best, companies can further develop new functionality; at worst, they can (nearly always) slow the spread of highly compatible formats.

XML will not suffer from shifting standards in the same way that HTML did. For HTML the issue was primarily extending the tagset. Since XML is fully extensible these issues will be non-existent. However, as language standards are reached XML parsers will have to keep up with those changes to effectively enforce the rules and structure of XML documents.

4.1.4 Bandwidth

As XML becomes more attractive for use in E-Commerce and EDI applications careful attention must be paid to how much metadata is contained in XML documents. A page of a given length in HTML might be two to three times more bandwidth expensive in XML. With EDI formats, which typically don't contain any metadata or markup, XML could be much more bandwidth intensive.

5. Development

Development of applications which utilize XML will be very OO focused. Development of an object model which describes the content of XML documents will be employed. This model will map closely with the DTD, describing the metadata rules for the XML documents.

Like traditional OO development methodologies a CASE tool may be used to generate relationship diagrams which use the unified modeling language (UML), interface definition language (IDL) or both. Coding the client, server, and document sources will begin once the models are complete.

5.1 Tools

SGML vendors are quickly realigning themselves as XML vendors. For the most part however, the tools selection is limited to first-generation only. Given the overall immaturity of the XML specification vendors will likely be targeting 3rd or 4th quarter of 1998 as a time frame for releasing powerful XML development solutions.

5.1.1 Datachannel XML Development Environment (DXDEE)

Datachannel is an early pioneer for XML tools. Their development environment includes a parser, XML generator, and channel manager.

5.1.2 Microsoft IE 4.0

Microsoft's early entry in to the XML parser field utilizes a C++ parser, Java validator, and data source objects.

5.1.3 ArborText XML Styler

This is a Java-based XML stylesheet generator which uses a wizard-type interface to create XML documents in a GUI interface. This styler is used in conjunction with the MSIE parser to generate the XML.

5.1.4 XML Parsers

- NXP is a validating XML parser written in Java by Norbert Mikula.
- Lark is a non-validating XML processor written in Java by Tim Bray.
- XP is another non-validating XML processor written in Java, by James Clark.
- MSXML is a validating XML parser written in Java by Microsoft.
- TclXML is a validating XML parser written in Tcl by Steve Ball.
- LT XML is an XML developers' toolkit from the Language Technology Group at the University of Edinburgh.
- JUMBO is a Java-based XML browser designed for the Chemical Markup Language, an XML application developed by Peter Murray-Rust.

6. Testing

XML documents themselves may be validated (as discussed in the Technology section), but no tools exist which test XML-based applications. Given the pace of development in the industry it should be only a matter time before some first-generation testing tools for XML documents become available.

7. Conclusions

XML-based technologies hold much promise. This simple technology may be able to link published content and delivery systems in very powerful ways. For managers and developers considering an XML solution, one word should sum-up their approach: cautious.

No doubt exists that XML's growth and popularity will skyrocket by the end of 1998. With vendor support as diverse as Microsoft, Netscape, Sun, HP, Sybase, Infomix, etc. marketable solutions will surely come of it. Building business applications with XML will partly be a matter of designing the systems and mostly a matter of sorting out the XML hype.

***

This whitepaper is composed by:
Jeff T. Pollock, Ernst & Young MC/SD&I