XML and the New Integration Frontier

By JP Morgenthal

Y ears from now, when we exchange information electronically as easily as we communicate today using speech, future textbooks will teach our children the lesson learned over the past 20 years. Buried in those lessons will be stories that they find as appalling as we today find the stories of using punch cards to load software on computers that took entire rooms to house. The story will go something like this …

In the early 1980s, the first personal computers landed on the desktops of business people and offered them the first real power to take control of their data. For years before that, data was owned by a select group of highly technical engineers who wielded and abused their power with delight. These engineers worked in special rooms with raised floors and temperature-controlled atmospheres that housed all the corporate data. This area became known as the “glass house.”

In this corporate computing environment, business people were provided terminals with which they could access small subsets of data, using applications and reports provided by those inside the glass house. Obtaining access to a larger subset of data required the approval of many higher-level managers. This did little to improve efficiency within the department with regard to gathering and analyzing data, so individual departments looked for solutions outside the glass house.

The PC and Local Area Network (LAN) revolution gave individual departments the ability they needed to take control of their own applications and data. This created the first need for application integration between the glass house and the rest of the business world. Previously, the glass house handled all application integration within the company and communicated only with glass houses of other companies. But individual departments could not escape the grasp of the glass house completely. Eventually, this data still needed to make its way back into the glass house for processing at the enterprise level.

The first attempt at this integration — commonly known today as client/server — was a disaster because those in the glass house were threatened by this new technology. So they didn’t provide proper access and interfaces to the existing applications — now commonly called legacy systems. Additionally, the new LAN-based applications were being designed to send and receive data over the network, using tightly coupled functional interfaces. Those interfaces failed to support larger processing loads as
others, outside the department, began to use the system.

Expensive middleware was the first solution to this problem. The middleware could emulate the tightly coupled interface, providing users the pretense that they were talking directly to the server that would process their request. Instead, the middleware would proxy requests on behalf of their clients and wait for their actual turn to speak to the server.

In the years between 1988 and 1997, corporations struggled to gain control of all the departmental systems that had grown significantly. Indeed, a new position was created for the Chief Information Officer (CIO), whose job became to lay claim to departmental data and make it the property of the enterprise once more.

Again, we were faced with another integration dilemma: now the departmental systems were talking with the older glass house legacy systems, but they were not communicating with each other. This time, the problem was significantly more expensive to fix and dramatically more complex. Instead of talking to one or two systems, it became necessary to speak to 10 or 12 systems. Moreover, all these systems were tied together, using expensive middleware and tightly coupled interfaces.

To finish this story would yield little more than a fairy tale, as this is where we are today. However, an entirely different story emerged in parallel between the years 1994 and 1999. In this story, the LAN grows up and becomes a network backbone that connects our machines together within our company as well as connecting us electronically to the entire world. This network is the Internet.

So, here we stand, at the beginning of a new millennium, watching these two parallel stories collide. We’re rethinking our entire approach to computing over the past 20 years so we can efficiently make our departmental systems communicate with each other and to the world at large over the Internet.

One technology has become more heavily relied upon for integration — Message-Oriented Middleware (MOM). MOM has been around as long as networks have existed. It includes the class of software known as electronic mail, but is more often identified as the software that asynchronously connects two disparate systems using pure data messages transmitted in a variety of formats.

Expensive middleware was the first solution to this problem. The middleware could emulate the tightly coupled interface, providing users the pretense that they were talking directly to the server that would process their request. Instead, the middleware would proxy requests on behalf of their clients and wait for their actual turn to speak to the server.

In the years between 1988 and 1997, corporations struggled to gain control of all the departmental systems that had grown significantly. Indeed, a new position was created for the Chief Information Officer (CIO), whose job became to lay claim to departmental data and make it the property of the enterprise once more.

Again, we were faced with another integration dilemma: now the departmental systems were talking with the older glass house legacy systems, but they were not communicating with each other. This time, the problem was significantly more expensive to fix and dramatically more complex. Instead of talking to one or two systems, it became necessary to speak to 10 or 12 systems. Moreover, all these systems were tied together, using expensive middleware and tightly coupled interfaces.

To finish this story would yield little more than a fairy tale, as this is where we are today. However, an entirely different story emerged in parallel between the years 1994 and 1999. In this story, the LAN grows up and becomes a network backbone that connects our machines together within our company as well as connecting us electronically to the entire world. This network is the Internet.

So, here we stand, at the beginning of a new millennium, watching these two parallel stories collide. We’re rethinking our entire approach to computing over the past 20 years so we can efficiently make our departmental systems communicate with each other and to the world at large over the Internet.

One technology has become more heavily relied upon for integration — Message-Oriented Middleware (MOM). MOM has been around as long as networks have existed. It includes the class of software known as electronic mail, but is more often identified as the software that asynchronously connects two disparate systems using pure data messages transmitted in a variety of formats.

Enter XML

The evolution of Extensible Markup Language (XML) is an interesting story. In 1997, a group of individuals led by Jon Bosak of Sun Microsystems decided that the Internet offered far more opportunity than Hypertext Markup Language (HTML) could ever leverage. These individuals had experience using the Standard Generalized Markup Language (SGML), of which HTML was an application. They decided that a subset of features of SGML could be applied to the Internet to create an information pipeline, not just a visual data portal. Together, this group created XML, with an initial emphasis on publishing information in a manner that could be easily recognized and reused by many.

At the same time, two new industries were just beginning to emerge:

- The Enterprise Application Integration (EAI) industry, which had developed to assist companies in linking together all their disparate, departmental applications
- The e-commerce industry, which had developed to help companies use the Internet for communication with trading partners and customers.

Both of these industries share a common requirement — to share information among many parties without the expense associated with point-to-point integration. (Integrating two single systems at once typically requires too many resources and too much time to be cost-effective.)

Not surprisingly, both of these industries recognized that MOM could simplify the problem because it removes the need for all of these systems to be simultaneously available and opens pathways for communications among a large group of systems. The last link in the chain would be establishing a single syntax used by all of these messages across the pipe. This is the role XML was born to fulfill.

What’s Makes XML Special?

Initially, XML was described as the predecessor of HTML — the new language of the visual interface for the Internet. Much has been written about XML since then to significantly clarify the picture, but the key to XML’s strength is often overlooked. As both the EAI and e-commerce industries are now beginning to realize, it’s not enough to have common syntax for messages. Real electronic communications requires a semantic understanding of the data being exchanged. That is, systems need more than to understand that price is being transmitted in a particular segment of a message, it needs a context to apply price against.

In the older Electronic Data Interchange (EDI) world, this context was
supplied, using relative positioning of records and groups of records. One line of a file would indicate that the next two lines represented purchase order detail, of which price would be a component. This approach works well. However, due to technological constraints, such as bandwidth and processor speeds, EDI was forced to implement this data as a series of mnemonic codes that can be misconstrued and are difficult for humans to read. Despite these shortcomings, EDI was successful as a technology for information exchange between trading partners.

We can now overcome the limitations of EDI. Currently, XML is the best syntax to represent that data. It is the perfect fit because of its ability to clearly and accurately represent context — providing semantic benefit, too. That is, XML lets us combine our data and metadata — the semantic meaning — in a single, encapsulated unit that’s vendor- and machine-independent. The example XML document in Figure 1 clearly shows this:

From Figure 1, we can ascertain key relationships about different items of data with regard to the whole entity. This document states that we wish to send this list of product items and total cost due to Dan Cocos. “Bill_To” represents the entity to which this invoice should be delivered. This is sometimes referred to as self-describing data because the tags — called element types in XML — describe the information contained within.

There are also some interesting implications based on the way XML lets us define these element types. For example, the statement of currency on prices ensures we’re discussing U.S. dollars, not French francs. Such information is important for letting machines and humans understand the data inside an element. Prior to XML, many standard data formats had to leave this information out or provide it in an obtuse manner.

XML-Based Application Integration (XAI)

Application integration has been going on for decades, but, in earlier times, integration was far less complex. That’s because our systems were centralized and used a common repository for data and metadata. In those times, integration merely required engineers to add new tables or files to describe how data was linked to other systems’ data.

Since the advent of client/server technology, application integration has become far more complex. Repositories for data and metadata are now on disparate hardware platforms in many data formats. Initial attempts at integration tried to eliminate these differences by creating an abstract interface to the data that was openly published. However, this resulted in specialized software to communicate with the abstract definition and map the results into existing systems. Typically, this occurs between only two systems at a time or point-to-point. Additionally, these interfaces may not be complete with regard to the processing requirements of other systems.

Recently, business has been moving away from the abstract interface concept toward a data-oriented paradigm. In this paradigm, systems communicate through well-established communication protocols and send each other messages in agreed-upon formats. This still requires specialized code to map from the input message format to an existing system, but now the system performing the mapping controls the data. Additionally, by default, these messages can be, and usually are, more data-rich since the sending system cannot assume what data the receiving system actually needs. The only information the sending system has is that a particular party, with certain access restrictions, seeks a set of information.

This change in control is the first major step toward a more sophisticated level of integration. Using XML to represent the data being exchanged is the second major step because it lets the receiv-
The ultimate goal for XAI is Straight-Through Processing (STP) of all requests from manufacturer to consumer. In the future, manufacturers will design products and give Original Equipment Manufacturers (OEMs) and resellers the description for that product as an XML message. The reseller group will add information to the message that includes availability, pricing, and delivery options. This new message will then be used to feed various catalogs read by consumers who will initiate electronic XML-based transactions. Of course, while XML is enabling this scenario, those using and updating the information may never see the raw XML. It will appear to humans through procurement applications, Web browsers, and hand-held devices. This is the true benefit of XML and XAI. Machines will communicate with other machines to provide the information humans need to generate and drive the exchange of goods and services.

Conclusions

There are many technical implementations of XAI, such as RosettaNet, Commerce XML (cXML), and BizTalk. However, understanding the importance of these initiatives requires understanding past initiatives. XAI was not developed to be a new way to sell products over the Web. Its evolution has been a long, bumpy road that required millions of lines of code and man-months of work. It is the apex of current technologies such as increased bandwidth, inexpensive and widespread Internet access, and cooperation of industry to agree on important standards.

The following are a few of the key initiatives XAI has helped to enable.

---

XEDI: This is an openly published specification that describes how to map from legacy EDI formats to XML. The Aerospace Industries Association (AIA) has endorsed it. XEDI enables organizations with large EDI installments to communicate with smaller trading partners that spurn or cannot afford traditional EDI.

BizTalk: This is an industry consortium, led by Microsoft Corp., to define a standard XML grammar for XML-based messaging. BizTalk defines how to include XML and non-XML documents as payloads of an XML document. It's designed so that a BizTalk server that has the ability to take appropriate action on the payloads can process it.

XFRML: This is an initiative, led by the AICPA, to define a standard XML-based grammar for reporting financial information from public companies to multiple agencies.

XML-Schema: This is a working group within the W3C that is tasked with creating a more powerful mechanism than DTDs for describing the structure of XML documents. A recommendation is expected within months from this group.

XML-Query: This is a working group within the W3C tasked with creating an “algebra” (the set of operations), language syntax, and underlying (meta)data model, for accessing and processing XML data. This group has just started its work.

XSLT: Extensible Stylesheet Language Transformation lets users define how to transform one XML document into another XML document. Using these stylesheets, a generic processing engine can perform the transformation. The resulting XML document can include other formats, such as HTML and comma-delimited.

---

About the Author

JP Morgenthal is CTO of XML Solutions, Inc. a supplier of B2B e-commerce software and consulting. He is a leading authority on enterprise application integration and e-business technologies. E-mail: jpm@xmls.com. Website: www.xmls.com.