

Native

XML

vs. XML-enabled



... the difference makes a difference!

Contents

Simple concepts win	1
How things develop	2
It's "I", not "ME"	3
Unleash the power of XML	4
Quick fix or holistic approach?	5
XML -- enabled or native?	6
XML-enabled products	7
Native XML products	8
Database management systems	9
XML	10
RDBMS	10
Document authoring systems	11
Other application areas	12
Conclusion: XML "steel" vs. XML "glue"	12

More detailed information about XML is available at our Web site: www.softwareag.com. You are also invited to view our landmark Web conference at www.xml4e-business.com.



Simple concepts win

Great inventions don't have to look difficult to be great. As a matter of fact, most breakthrough inventions are great because of their extreme simplicity and broad appeal.

And, most great inventions take time to be implemented. However, as a rule, we always overestimate an invention in the short term and always underestimate the impact of such technologies in the long term. I think XML is such a technology. As some of the hype around XML subsides, people will underestimate how fundamental XML will change the way we store, publish and exchange information in the next decade.

Do you believe you are involved in something extremely significant as you are working with XML? I believe XML may prove as significant to the industry at large as any other major invention in the history of humankind.

But, you have to be careful in choosing your route.

How things develop

For example, the ancient Mayas already knew of the wheel but did not have horses to make horse-drawn carriages. The Chinese knew about steam power about 2000 years ago but only used it to raise the throne of the emperor during coronation ceremonies. When the telephone was invented, the then U.S. president said: "I think the telephone is great, I can see a time when every town will have one." Now there are nearly twice as many mobile telephones as there are personal computers.

When Edison invented the modern light bulb, did he think about the Hoover dam? When Stephenson invented the first steam locomotive, could he even imagine a bullet train? When the Wright brothers flew on the beaches of North Carolina, did they ponder frequent flier miles?



As much as gasoline was the killer application for the oil barons of Texas and the Middle East, XML will be the killer application for software developers that want to deliver rich, personalized content to the growing community of online citizens. XML technology will go a similar route to the examples above. Those that fail to act on the XML shockwave now will get steamrolled later.



It's "I" not "ME"

At this point in time, we are still being bombarded with new products that start with the letter "E" and increasingly, with the letter "M" for mobile. Well, these terms still miss the point. Those of us that are dealing with software are not dealing with electrons or with mobile devices as such. These are just technical details. The point is the letter "I".

"I" for *Information*. Information as in data that has context and meaning, which enables normal people to make decisions and to create value in the new economy. All the electrons and mobile devices in the world will not create value unless people can understand the information that is being provided to them. And, XML is the key technology to make information understandable—and to make the Internet truly useful in the economic domain.

Unleash the power of XML

If you want to unleash the power of XML today—whether for B2B, B2C electronic commerce, content management, application integration, or portals—you obviously won't be starting from scratch. You'll need to integrate your core business processes and perhaps those of your business partners. You'll need to store and retrieve XML information in a native, reliable and rapid fashion.



When intelligently incorporated into your strategic plan, the power of XML will amaze you.

Quick fix or holistic approach?

To reap the benefits of data exchange and management via XML, will it be enough for you to make quick fixes that “XML-Enable” the outward-bound interfaces of existing applications, relational and post-relational databases and messaging systems?

While such quick fixes are sometimes useful and necessary, they ignore the big picture and open up a Pandora's box of complex questions. For example: Where to use XML and where to leave it out; When to transform one XML format to another; When to transform XML into another system's proprietary format; Where to do the transformations.



XML-enabled or native?

The market is currently laden with so-called “XML-enabled” products that claim support XML as an input/output format. While these products clearly have advantages over others without XML support, another class of products referred to as “native XML” offer significant additional advantages. These products, which support XML down to their internal architectures, are more scalable, more reliable, and even more truly interoperable than those that merely uses XML as a data exchange format.



XML-enabled products

Many products currently support XML as an input/output format, that is, they can translate back and forth between their internal data formats and APIs and those of XML. Such “XML-enabled” products can exchange data with products running on other platforms, and in some cases they can be programmed to a limited extent by means of code written to XML-related specifications.

This has encouraged some companies to use XML as “glue” to connect existing enterprise systems with others within a single company, with those of suppliers and customers, and to present live data to consumers over the Web. A clear example of this sort of use case for XML is SOAP, an XML-based object serialization format that can be used to perform asynchronous messaging and remote procedure calls between non-XML applications using the Internet infrastructure.





Native XML products

Another class of products, however, supports XML deeply in their internal architectures. Such “native XML” products offer significant advantages over those that are merely XML-enabled. Many of these advantages boil down to *scalability*: as the volume and complexity of e-Business transactions increases, the overhead needed to convert back and forth between XML and other data representation will seriously impinge on the speed, reliability, and functionality of “XML-enabled” systems. Native XML systems, which deliver not only the appearance but the reality of an XML architecture run faster, more reliably, and with less administration. Let’s consider some examples.

Database Management Systems

Perhaps the clearest way to illustrate this is to compare architectures that provide an XML view of an underlying relational database with those that store and index data in a native XML internal format.

The definition of a “Native XML Database” is one whose internal data structures map directly onto the hierarchical format of XML. Users of a native XML database are not required to distinguish between some external “interchange” format and an internal “efficient” format, nor to design applications that distinguish “business data” from “document content”. In a native XML database, such distinctions are meaningless.

Most RDBMS vendors now or will soon provide interfaces and utilities to allow XML data to be stored in their systems with relatively little obvious pain to the developer. But, consider the mismatches between XML data and normalized RDBMS storage that these interfaces must paper over:



XML

- ❖ Nested hierarchies of elements
- ❖ Elements are ordered
- ❖ A formal schema is not necessary
- ❖ Ordinary business documents can be represented, stored, and retrieved as a single object
- ❖ The XPath standard provides a common (if limited) query language for locating data.

RDBMS

- ❖ Data arranged in rows and columns, with atomic cell values, and multiple tables JOINed together must be defined to represent hierarchical relationships
- ❖ Row ordering is not defined
- ❖ A predefined schema is usually necessary to describe the structure of the data
- ❖ JOINS of several tables are usually necessary to retrieve even simple business documents
- ❖ Queries are done with SQL retrofitted with proprietary XML enhancements

In effect, the major database vendors have masked the immediate pain once required to store XML in an RDBMS. These interfaces, however, cause pains of their own as more complex XML documents and messages are stored and as the transaction volume increases.

The complexity of the underlying tables, separate full-text databases, and number of JOINS may be hidden from the developer, but will be a constant burden on the DBAs and system administrators responsible for a large-scale system.

Similarly, as the XML view of data and the standards that support it become more widely understood, end-users will employ more sophisticated queries that will be easy to express in XPath and future XML query languages but difficult to decompose into some combination of SQL and full-text queries.



Document Authoring Systems

A native XML text handling system that is truly built on implementations of standard formats, APIs, and protocols will tend to be easier to use and integrate than one that implements XML interfaces via translation. For example, contrast, using a native XML authoring tool such as SoftQuad XMetaL versus using an ordinary word processor to author content then converting it to XML format for storage and interchange.

Many vendors have devised clever techniques for minimizing developers' immediate pain by translating MS Word and/or RTF data produced by conventional word processing systems into XML, and this does indeed XML-enable these products in a way that can be useful. But, here again native XML tools have capabilities that will make them much more useful as the volume and complexity of your data increases.

XML authoring tools allow the author to be aware of the underlying distinctions in the XML markup that have no obvious equivalent in an ordinary word processor. These distinctions may be crucial, for example, in identifying the “essence” of a document that is to be preserved when it is translated to a format suitable for viewing on PDAs or mobile phones. Techniques that allow such content to be identified in MS Word are often fragile and “break” when new authors are hired, as documents are “round-tripped” between the authoring and storage environment, etc.

Other application areas

As industry standards are developed and vendors refine their native XML products, similar patterns will be seen in other areas, especially:

- ❖ Display and entry of ordinary business data presented in forms
- ❖ Routing and transformation of e-Business messages (B2B, EAI, etc.)
- ❖ Development of workflows, scripts, and software objects that automate the actual handling of data
- ❖ Extraction and cataloging of “metadata” describing the semantics of the information embedded in documents and messages

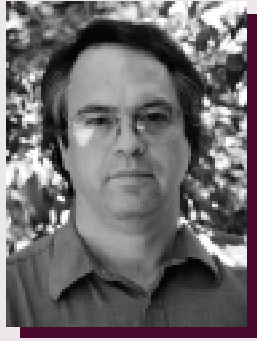
Conclusion—XML as “Steel” vs XML as “Glue”

XML offers developers very significant advantages when it is deeply embedded in the infrastructure of enterprise systems and not just used as a kind of industrial-strength glue to connect them together. Native XML systems are far easier to develop while being more scalable, reliable, and even more truly interoperable than those that merely uses XML as a data exchange format.

XML is the enabler for the information economy. Those of you who are writing applications in native XML will be the leaders of this new stage of software development. When we look back to this time 20 years from now, we will not remember the year 2000 as the year of potential catastrophic computer failures but as the year in which the Internet became truly valuable through the use of XML technologies.



What it comes down to is a “quick fix” versus a long-term strategy. If you’re weighing the pros and cons of Native XML versus XML-enabled, my advice is . . . go native!



Michael Champion is a Research and Development Specialist at Software AG. He is an alumnus of the University of Michigan and did graduate study specializing in data analysis and computer simulation. He has been a software developer in for 20 years, working primarily in the area of middleware for client-server document and image management systems. Mike has been active in the World Wide Consortium's Document Object Model (DOM) Working Group for more than three years and was a principal author of the core XML portion of the DOM Level 1 recommendation.



Software AG, based in Darmstadt, Germany, is one of the largest and most highly respected system software companies in the world and the premier provider of database management technology. With products and services in use globally, our focus is on mission-critical electronic business applications linking heterogeneous platforms, and our commitment to and support for open-standard XML technology is absolute.

Software AG is a founding member and active participant in the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS).

Software AG, Inc.
Bishop Ranch 3
2613 Camino Ramon, Suite 110
San Ramon, CA 94583-4289

©2001 Software AG USA. All rights reserved. Software AG and/or all Software AG products are either trademarks or registered trademarks of Software AG. Other product, company names and logos mentioned or depicted herein are the trademarks of their respective owners.